uxodocs

Fast2 / uxodocs

Author:	Your name
Date:	2026-03-13

Contents

Getting started with Fast2

This first section is your go-to resource for understanding and harnessing the power of Fast2 to build efficient and seamless document migration workflows. Whether you're new to the product or an experienced user, this documentation will provide you with all the necessary information to get started and create effective document migration workflows.

Overall Concepts

The "*Overall Concepts*" section provides a comprehensive overview of the principles and concepts behind Fast2. Gain a deep understanding of the underlying architecture, core functionalities, and key components that make Fast2 an industry-leading solution for document migration workflows. This section will help you grasp the fundamental concepts necessary to effectively use and configure the product.

Installation Guide

In the "*Installation Guide*" section, you'll find detailed instructions on how to install and set up Fast2. This step-by-step guide will walk you through the installation process, including system requirements, software dependencies, and configuration settings. By following the instructions provided, you'll have Fast2 up and running in no time.

Creating Workflows

The "*Creating Workflows*" section is your go-to resource for building powerful document migration workflows using Fast2. Explore the various components, features, and configurations available to create customized workflows tailored to your specific requirements. This section will provide you with detailed instructions, examples, and best practices to guide you through the process of designing, configuring, and executing workflows using Fast2.

Getting started / What you need to know before committing to Fast2

Basic jargon

Source

A source is a Fast2 task whose role is to gather the documents or items to migrate. As they are identified, the source converts them into punnets.

Punnet

The Punnet is the pivot format which is used for data mapping, content conversions and folder management. This is the migration entity, processed and then forwarded by the workflow tasks.

Task

A task is either an extract-, transform- or injection-step that composes a workflow. Each task can be configured to match the user's needs. Once all tasks are completed in the specific order, the migration is over.

Map

A workflow (aka "*Map*") is a succession of tasks, where the output of the ones is the input of the following others. Each task can be considered as a step of the workflow.

Campaign

A campaign is the perimeter where a map is executed (once or several times). Different campaigns can either be cumulative or independent.

Worker

The Worker is the punnet processor, applying the changes onto the punnet, according to how the tasks have been configured by the user.

They are waiting in silence to do their job. When a punnet needs to be processed by a task, the broker triggers the assigned worker.

If the workload is too important, you can manually add workers to speed up processing.

Broker

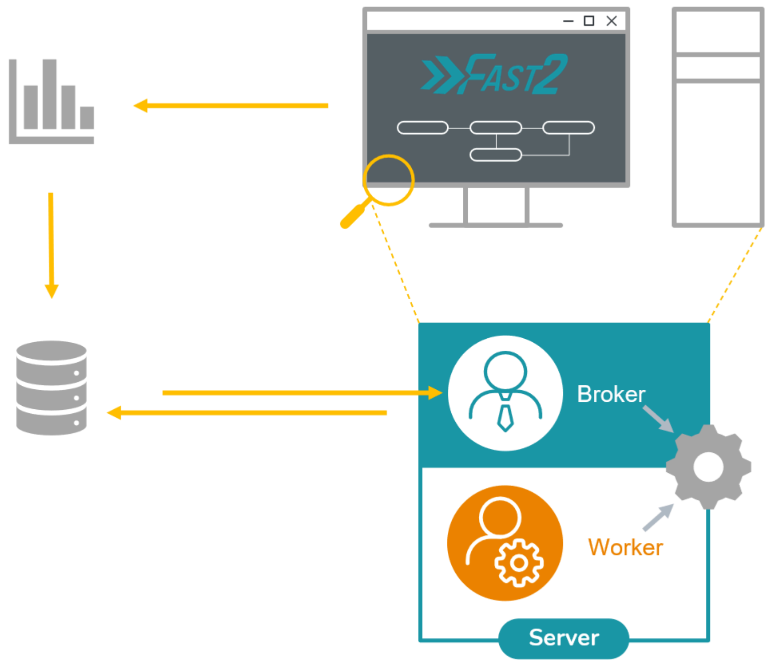
The broker is the trump card of the migration. It is basically the workflow orchestrator, in charge of database communication, sending punnets to the worker(s) for them to process the operations.

Scheduling, orchestrating or even managing queues : the broker is everywhere.

His first job is to handle the workers. Worker coordination is a key point in terms of performance, knowing that there may be a multitude of them.

In addition, the broker ensures the persistence and traceability of the data carried out by the punnets into the database, where logs, data and errors and more are stored.

Architecture



Fast2 objects

Folder

Folder object represents a folder in the ECM or file system sense, and can have metadata as well as links to documents

INFO

This type can be included into punnets and documents, and folders themselves.

```
▫ folder
        ▫ name
        ▫ path
        ▫ parent folder
                ▫ name
                ▫ path
                ▫ parent folder
                        ▫ ...
```

Dataset

Data object represents metadata in the ECM sense. It contains a name, a type, and one or more values.

INFO

This type can be included into punnets, documents and workflows.

```
▫ dataset
        ▫ metadata A (ex/ key: value)
                ▫ properties
                        ▫ property
                        ▫ property
                        ▫ ...
        ▫ metadata B (ex/ key: [value A, value B])
        ▫ ...
```

Content (aka 'ContentContainer')

Content object materializes document content that can be simple or made up of several pages. It can be materialized by a relative or absolute path to its storage location or stored directly in memory / in an XML file.

INFO

This type can be included into documents and annotations.

```
▫ content
        ▫ URL
        ▫ mime-type
        ▫ properties
                ▫ property
                ▫ property
                ▫ ...
        ▫ subcontents
                ▫ content
                ▫ content
                ▫ ...
```

Annotations

Annotation object represents an annotation (post-its, arrow…) affixed to the content of a document. This object is not conceptualized in all ECM systems.

```
▫ annotation
        ▫ ID
        ▫ content
```

Document

A document can also contains its own data, its content with annotations and the folder where it is stored.

INFO

This type can be included into punnets and workflows.

```
▫ document
        ▫ documentId
        ▫ dataset
        ▫ contents
        ▫ mime-type
        ▫ folders
        ▫ annotations
```

Workflow

```
▫ workflows
        ▫ dataset
        ▫ associated documents
                ▫ document
                ▫ document
                ▫ ...
```

Punnet

As introduced above, the punnet gathers all the different assets to migrate.

```
▫ punnet
        ▫ punnetId
        ▫ documents
                ▫ document
                ▫ document
                ▫ ...
        ▫ dataset
        ▫ workflows
        ▫ folders
```

When serialized in XML format, it will look roughly like :

```
<?xml version='1.0' encoding='UTF-8'?>
<ns:punnet xmlns:ns="http://www.arondor.com/xml/document"
punnetId="34c5434c-4234-4fa2-9f91-7882a899a994#1">
        <ns:documentset>
                <ns:document documentId="34c5434c-4234-4fa2-9f91-
7882a899a994">
                        <ns:contentset>

<com.arondor.fast2p8.model.punnet.ContentContainer
contentStorage="URL">

<ns:url>C:/samples/file.pdf</ns:url>

</com.arondor.fast2p8.model.punnet.ContentContainer>
                        </ns:contentset>
                        <ns:dataset>
                                <ns:data name="name" type="String">
                                        <ns:value>sample</ns:value>
                                </ns:data>
                        </ns:dataset>
                        <ns:folderset>
                <ns:folder parent-path="/primary-folder/subfolder"
name="sample">
                    <ns:dataset />
                </ns:folder>
            </ns:folderset>
                        <ns:annotationset />
                </ns:document>
        </ns:documentset>
        <ns:dataset />
        <folderSet />
</ns:punnet>
```

Lifecycle

The punnet will iterate through the follwing lifecycle until the last step is reached.

Task

Task can be represented as a processing unit to be applied to a punnet. A punnet comes at the entry of the task, as an input. The task performs operations and then outputs the modified punnet.

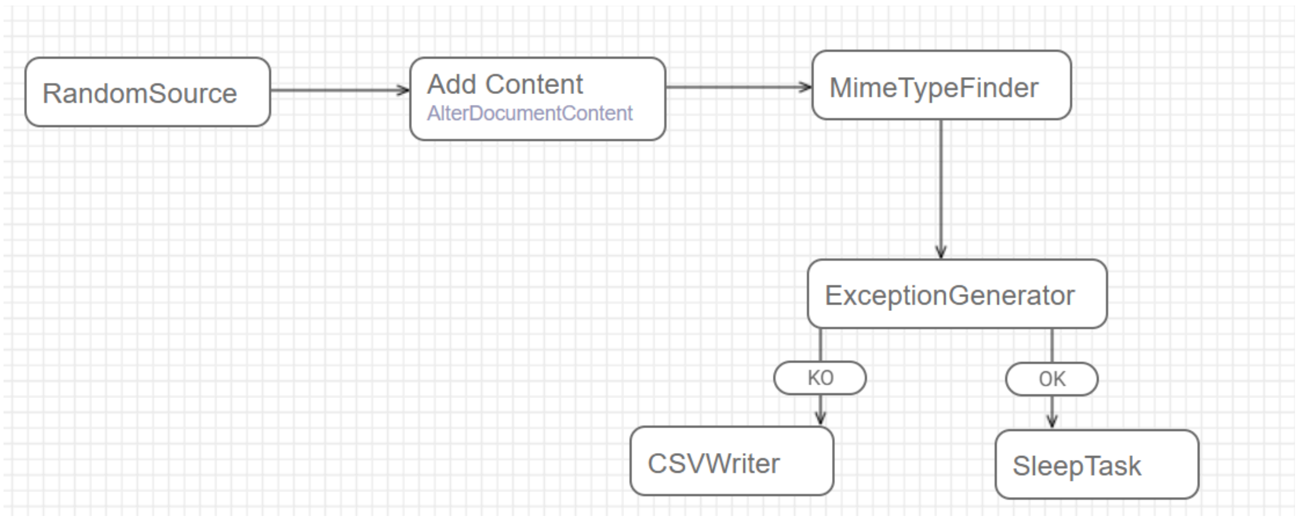
During the processing of each task, statistics are collected allowing to know the number of punnets processed per second. This is the actual throughput of the task and it is of course dependent on the environment (neighboring tasks, multi-thread…) From this speed, Fast2 tries to estimate the average time left for all the tasks.

One of the benefits of these statistics is the ability to visualize bottlenecks. Sometimes some tasks have a longer processing time than others. Thanks to the visualization of the queues, it is quite easy to know which task is greedy.

When multiple tasks are linked together it represents a processing chain or a workflow where each punnet will be processed task by task. We will call this object a campaign.

Map — workflow

Map is the Fast2 word for the workflow. It is a collection of tasks.



Campaign — workflow instance

As we have just seen, a campaign is made up of several tasks. In other words, it represents an instance of a map.

WARNING

A campaign has a unique name

Despite this uniqueness, each campaign can be ran multiple times. The statistical data of the new run will be added to the previous run(s).

Other runs can be added on top of this one.

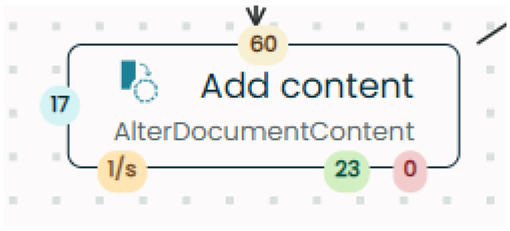
A user has the opportunity to stop any campaign when he wants. He can always resume the campaign later, or start a fresh one.

A retry feature is also available after each campaign. This makes possible to filter certain punnets and to replay them directly in the campaign. For instance, retry each punnet in exception. You can even select which type of exception you want.

Lifecycle

Punnet status

For each task, the color bubble indicates the status of punnet processing.



Each colored bubble shows a specific metric:

🟡 Yellow (Top) – Number of punnets waiting to be processed

🔵 Blue (Left) – Number of punnets currently being processed

🔴 Red (Right) – Number of failures

🟠 Orange (Bottom Center) – Processing speed, in punnets per second

🟢 Green (Right-Center) – Number of successfully processed punnets

Operating

Getting started / Installation

Requirements

The installation of Fast2 requires a few environment specifications to run properly :

What		Description
RAM	8GB+	We highly recommend having at least 8GB. When switching to production environments, 16GB or 32GB will be required since more documents will be handled at once, and heavy tasks (*e.g.* conversion, extraction) might get short on resources.
Processor	8 CPUs	Processor capabilities need to be aligned with migration requirements, such as data mapping, content conversion and heavy I/O.
Storage	500GB+	Although the contents dealt by Fast2 will be temporarily stored (and deleted afterwards if asked), the server needs enough storage for the files/contents alongside the database tracking all the migration information.

What		Description
		For production environments, we strongly recommend **500GB or more** to handle large-scale migrations and prevent storage-related issues. While 128GB may work for development or testing, it is insufficient for production workloads and can lead to increased technical operations during migrations.
Java	JDK-11	Any provider will fit (Oracle, OpenJDK, etc). If you have multiple JDK/JRE already installed, specify the correct one in the `./config/env.properties` file.

What		Description
OS	Windows 7+, Linux	All versions of Windows 7+ are supported. All common distros of Linux are supported (Ubuntu, RedHat, CentOS, etc) Power architecture are supported as well (except the ones running in AIX), but only Java parts will work seamlessly whereas third-party software (*e.g.* imagemagick, libreoffice, etc) might not, as they have not all have been developed for such platforms. Although the broker will not run correctly on an Windows 2003, a worker can still run on it, remotely, and communicate with a broker installed on a more recent version.
Bandwidth	1GB	The more calls, payloads, and contents Fast2 will have to deal with, the bigger the network bandwidth must be to reduce latency. If 250-500MB might do for lower environments, we recommend 1GB for Production environments.

While setting up the production server for Fast2, make sure to scale the Fast2 machine accordingly. You may need to increase the allocated memory for both the broker and the background database. If you planned to deal with campaigns of a few millions of documents, setting **8GB** of memory for the broker and **8GB** for the database as well is a good starting point.

WARNING

If you decide to go for a custom Elasticsearch database, make sure to confirm the compatibility with your environment at Elasticsearch Support Matrix.

Fast2 packages

The Fast2 distribution you need depends on your target environment. It exists three way to deploy a Fast2 :

On premise: regular package, as an all-in-one zip file
AWS: Standard AMIs
K8S: Docker Images

Each distribution ships the following

- A broker with one embedded worker and a user interface
- An additional worker with all tasks catalog
- A template to create workers with custom tasks

Root folder anatomy

Item	Purpose
config	Configuration files, broker endpoint, Java home

Item	Purpose
logs	Logging files for both broker and worker(s)
maps	XML files of all maps accessible from the UI
opensearch-X.Y.Z or elasticsearch-X.Y.Z	Either Elasticsearch or OpenSearch
service	All files required to start Fast2 as a service
worker-libs/*	All libraries and dependencies for tasks executions
fast2-broker-package-X.Y.Z.jar	Broker unit
fast2-worker-package-X.Y.Z.jar	Worker main unit
startup-broker.bat	Binary file for Windows
startup-broker.sh	Binary file for Linux
startup-worker.bat	Binary file for Windows
startup-worker.sh	Binary file for Linux

Start-up sequence

When Fast2 is started, either as a standalone application or a service, its different modules share a precisely defined roll-out schedule:

- First, the broker and its internal databse are started. The connection between these 2 components has to be effective, otherwise Fast2 will automatically shut down after a couple of attempts to reach the database;
- The worker is then triggered, and has to register itself to the broker.
- Finally, the dashboard will be started if asked so, and if the binaries have been detected. First, Fast2 will try to connect to any dashboard instance running on the configured port.

There is no direct connection between the broker and the dashboard. The only exchange area is the Elasticsearch database, as explained in the architecture section.

Start Fast2 Broker

Once the regular Fast2 package is unzipped, Fast2 can be launched right away.

Whether Fast2 is launched from the batch file or as a service on your environment, the UI will be available at http://localhost:1789/.

By default, Fast2 Broker starts an embedded Elastic Search and an embedded Fast2 Worker.

All commands below are to be run under the Fast2 install path (where the Zip has been unzipped).

From command line

Windows Linux

Go into the Fast2 install folder, and run :

```
C:\path-to-fast2\> startup-broker.bat
```

Administrator rights might be required since Fast2 will handle some port communications.

To end the Fast2 process, just hit `Ctrl+C` in the command line the startup file opened.

As service

Windows Linux

Go into the Fast2 installation folder, and open the Windows Command Prompt.

To install the service :

```
C:\path-to-fast2\service\windows> Fast2_broker_service.exe install
```

Your machine may prompt a message asking to download .NET components. Please click [OK] and proceed.

Once this command is complete, you should see in your services registory a newly installed Fast2 service. You can start/stop/restart it as any other service, or via the Command Prompt (just replace `install` in the previous command by *start/stop/uninstall/restart/status*).

The logs of the service will be available from the `path-to-fast2\service\log` folder.

Start Fast2 Worker

The Broker starts an embedded worker by default.

Windows **Linux**

If you wish to start multiple workers, just hit :

```
C:\path-to-fast2\> startup-worker.bat
```

If the worker and broker are not booted up on the same machine, you need to setup the Broker host name in the worker configuration file. Edit the file `config/application.properties` and modify `broker.host` accordingly.

You can setup Fast2 Worker as a service the same way you did for the Fast2 Broker. For a worker installed on a Windows machine, you need to edit 2 files :

- The `Fast2_broker_service.exe` file in the `service/windows` folder has to be renamed to `Fast2_worker_service.exe`
- The `Fast2_broker_service.xml` file in the `service/windows` folder has to be renamed to `Fast2_worker_service.xml` and requires some changes :
    - The `id` tag has to be changed to something different than `Fast2` in case the broker has already been installed as a service on this machine. For instance,`Fast2-worker`.
    - The `name` tag has to be changed to something different than `Fast2 Broker` to avoid any confusion in case the broker has already been installed as a service on this machine. For instance, `Fast2 Worker`.

  - The `description` tag has to be changed to something different than `Fast2 Broker vX.Y.Z` to avoid any confusion in case the broker has already been installed as a service on this machine. For instance, `Fast2 Worker vX.Y.Z`.
  - The `executable` tag has to be changed to point to the `startup-worker.bat` file which is at Fast2 root level.

The `Fast2_worker_service.xml` file will look like this :

```
<service>
    <id>Fast2WorkerDCTM</id>
    <name>Fast2 Worker DCTM</name>
    <description>Fast2 Worker DCTM v-2.12.1</description>
    <env name="FAST2_HOME" value="%BASE%\..\.." />
    <executable>%BASE%\..\..\startup-worker.bat</executable>
    <logpath>%BASE%\..\log</logpath>
    <startmode>Manual</startmode>
    <log mode="roll-by-size">
        <sizeThreshold>10240</sizeThreshold>
        <keepFiles>8</keepFiles>
    </log>
</service>
```

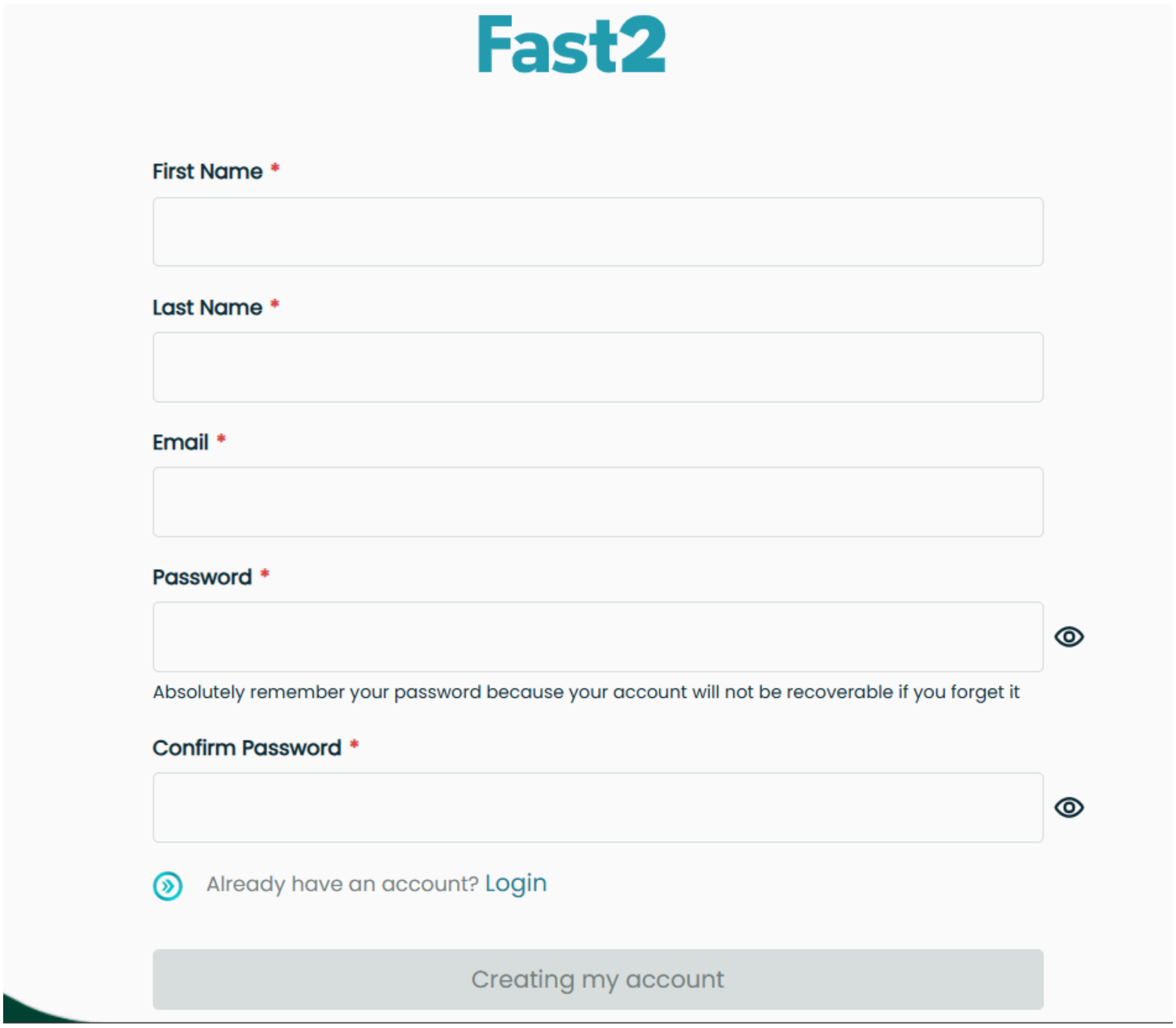
And then, to install the worker as a service :

```
C:\path-to-fast2\service\windows> Fast2_worker_service.exe install
```

Getting started / Authentication & Team management

Account registration

When reaching the Fast2 UI for the first time, you will be prompted to create an account.


Fast2

First Name *

Last Name *

Email *

Password *

Absolutely remember your password because your account will not be recoverable if you forget it

Confirm Password *

Already have an account? Login

Creating my account

To create a new account in Fast2, follow these steps:

1. **Fill in the Required Information**: Enter the following details: : **First Name**: Enter your first name. : **Last Name**: Enter your last name. : **Email**: Provide a valid email address. This will be used for login. : **Password**: Choose a strong password. Remember that your password is crucial for securing your account. : **Confirm Password**: Re-enter the same password to confirm it matches.

2. **Review the Password Guidelines**: : **Ensure your password meets the security requirements**: At least 8 characters long and maximum 16 characters long.

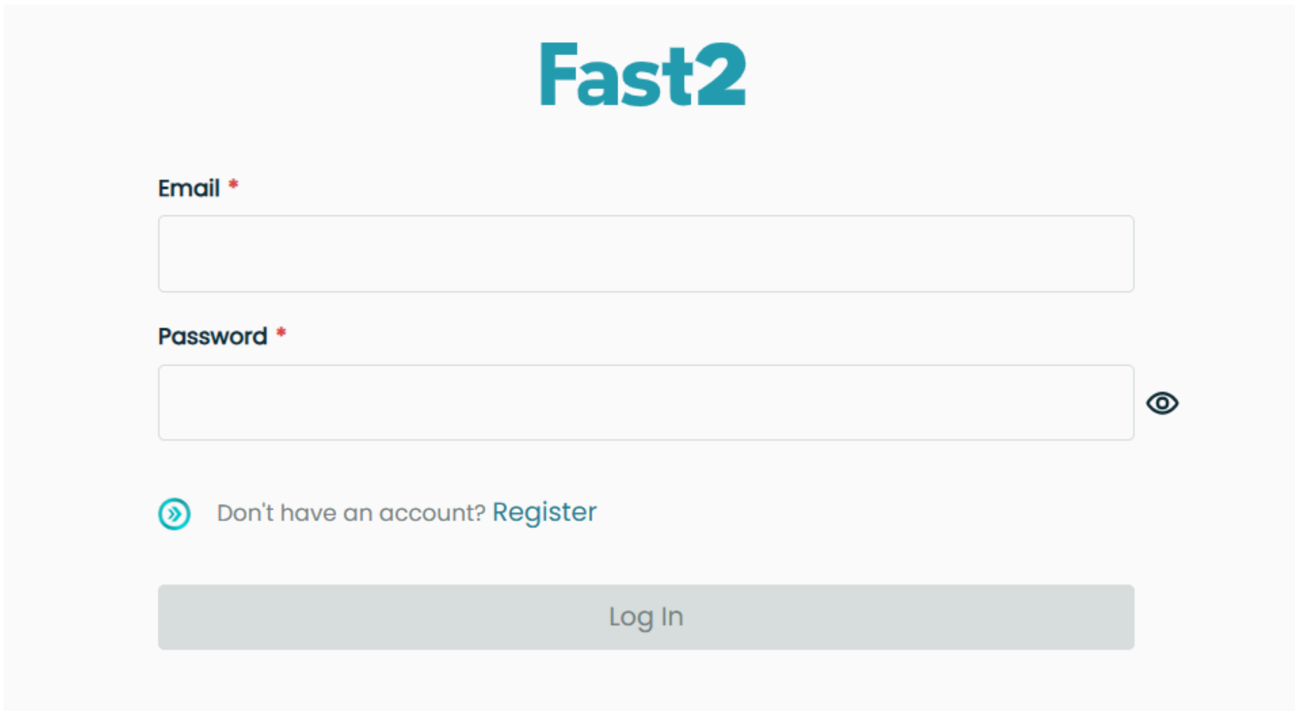
3. **Submit Your Information**: : Click on the "Creating my account" button to complete the registration.

WARNING

It is essential to remember your password. Fast2 does not offer password recovery for forgotten passwords.

Login

Once you have registered, you can log in to the Fast2 UI using your email and password.



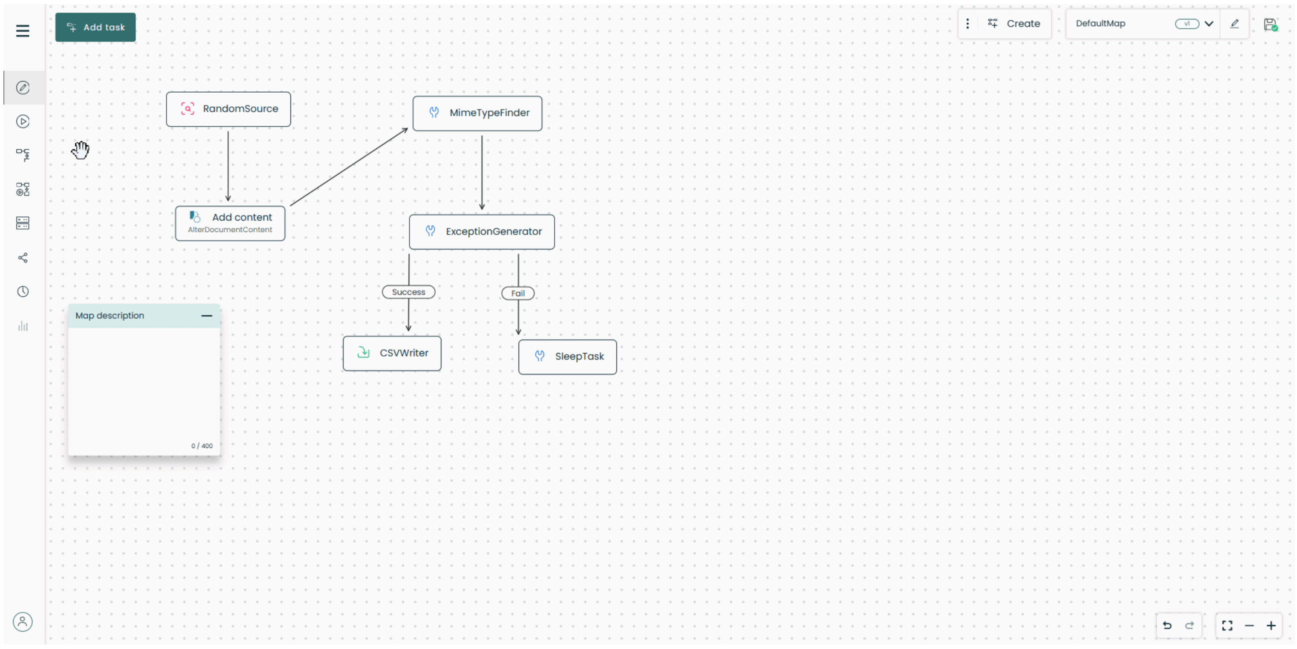
Team management

Fast2 allows you to manage multiple users. **When starting Fast2 for the first time, the created user will be the super admin.** The Team Place allows authorized users (Super Admin and Admins) to manage team members in a Fast2 Place. It includes:

- Viewing all members with details: Name, Email, Role
- Adding, deleting, and updating member roles
- Resetting member passwords

Navigating to Team Place

To access Team Place: Click on your account avatar in the bottom left corner of the screen and then select **Manage Team** from the popup menu.



User Access Restrictions

- **Super Admin/Admins**: Full access to Team Place.

- **Users**: If a user tries to access the Team Place, he will be redirected to EditPlace with error toast:
  > `You do not have permission to access this Place`

Adding a new member

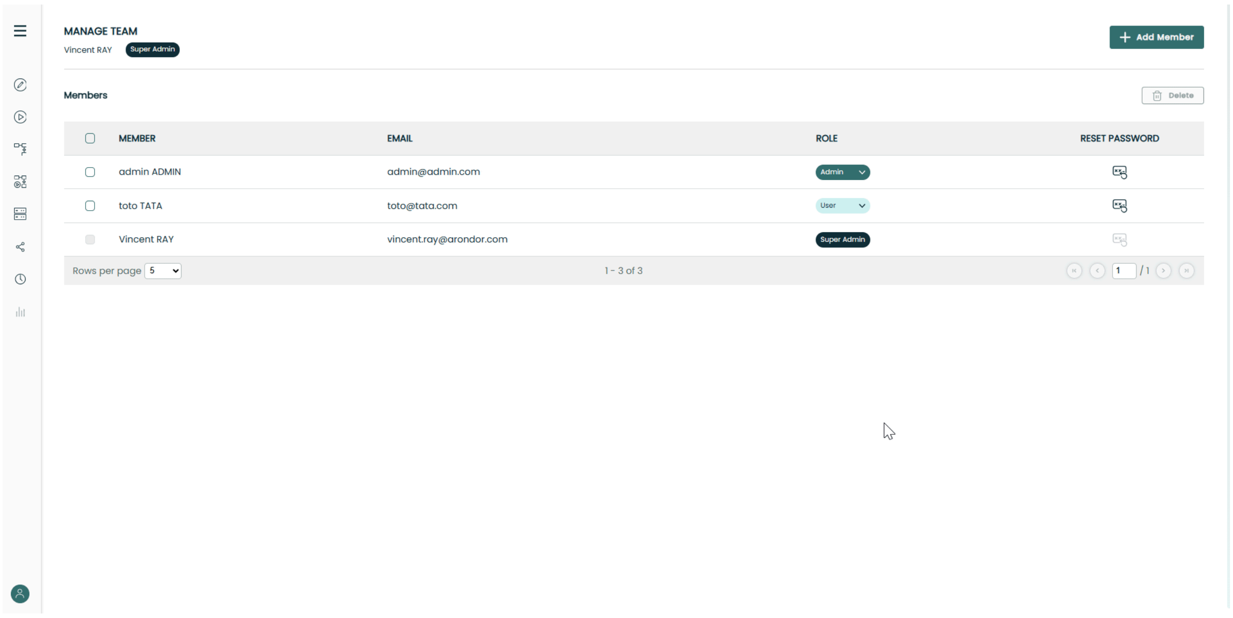
Adding a new member as Super Admin

1. Click **Add Member**.
2. A modal appears with the following fields:
   - Role *(default: User, editable)*
   - First Name
   - Last Name
   - Email
   - Password
   - Confirm Password

Validations

- Email format validation and uniqueness check.
- Password requirements displayed when focusing on the field.

- Save button enabled only when all fields are valid.



MANAGE TEAM

Vincent RAY Super Admin

Members

	MEMBER	EMAIL	ROLE	RESET PASSWORD
☐	admin ADMIN	admin@admin.com	Admin	
☐	toto TATA	toto@tata.com	User	
☐	Vincent RAY	vincent.ray@arondor.com	Super Admin	

Rows per page 5 1 - 3 of 3 1 / 1

Adding a member as Admin

- Button labeled **Add User**.
- Role field preset to *User* and **read-only**.

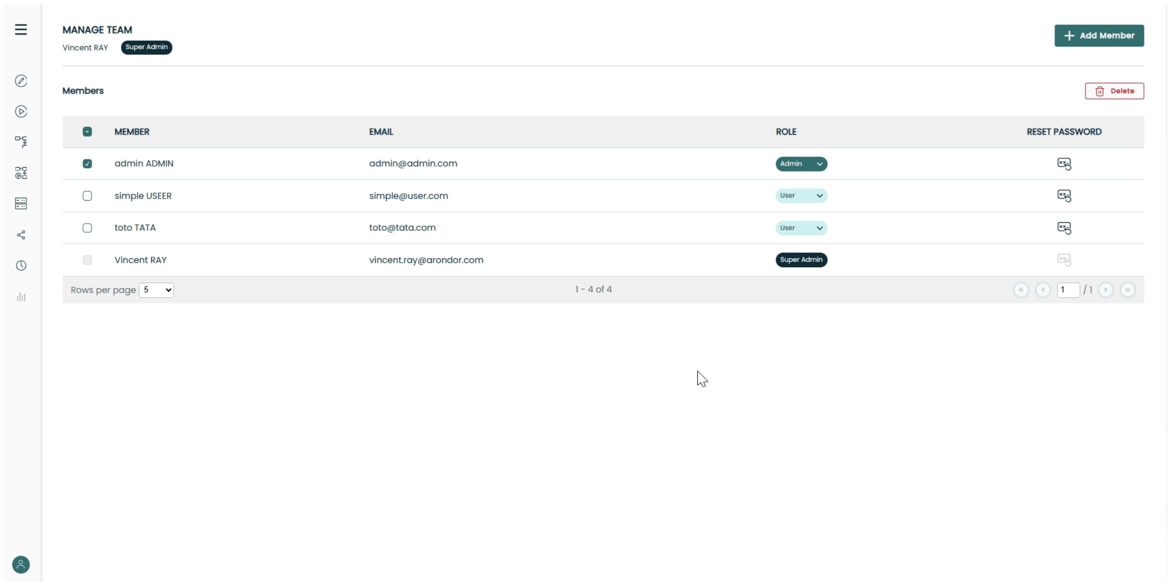
Only Super Admin can add new members as Admins.

Deleting a member

Deleting a member as Super Admin

1. Select one or more members (Users or Admins) via checkboxes.
2. Click **Delete**.
3. Confirmation popup appears with:
   - Member details
   - Warning: `This action cannot be undone`

- Buttons: **Cancel**, **Delete**



	MEMBER	EMAIL	ROLE	RESET PASSWORD
✓	admin ADMIN	admin@admin.com	Admin	
☐	simple USEER	simple@user.com	User	
☐	toto TATA	toto@tata.com	User	
☐	Vincent RAY	vincent.ray@arondor.com	Super Admin	

Rows per page 5 1 - 4 of 4 1 /1

Deleting a member as Admin

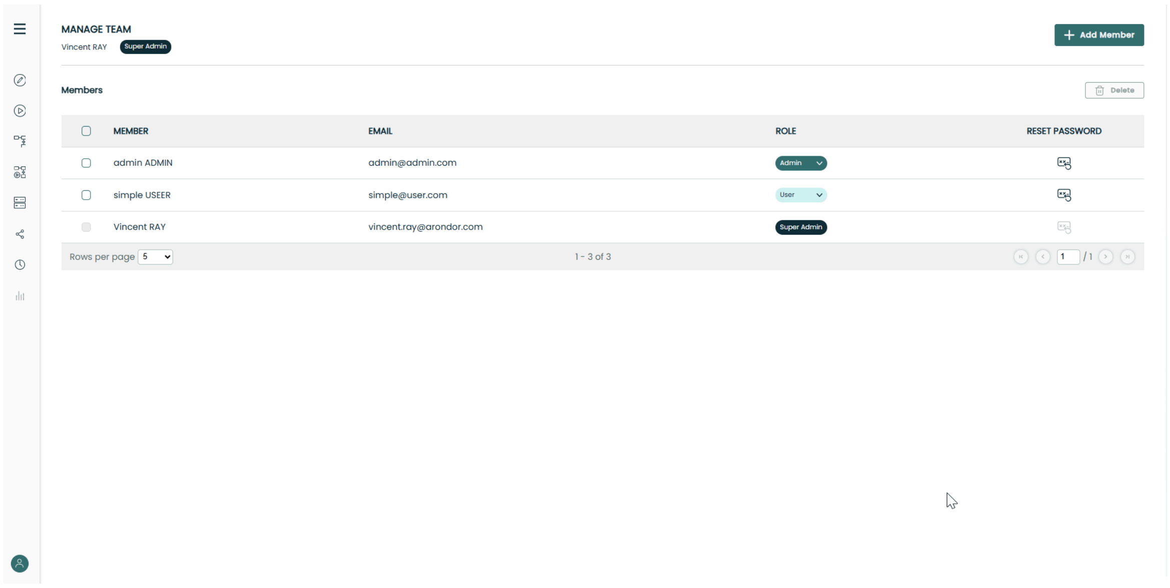
- Can delete **Users only**.
- Checkboxes for Admins and Super Admins are disabled.

Changing a member's role

Changing role as Super Admin

1. Click on the role tile (User/Admin) for a member.
2. Mini popup appears:
   - *“Promote member → Admin”* or *“Demote member → User”*
3. Confirm in the modal popup:
   - Member details

  - Warning about role change permissions



MEMBER	EMAIL	ROLE	RESET PASSWORD
admin ADMIN	admin@admin.com	Admin	
simple USEER	simple@user.com	User	
Vincent RAY	vincent.ray@arondor.com	Super Admin	

Rows per page 5 1 - 3 of 3 1 / 1

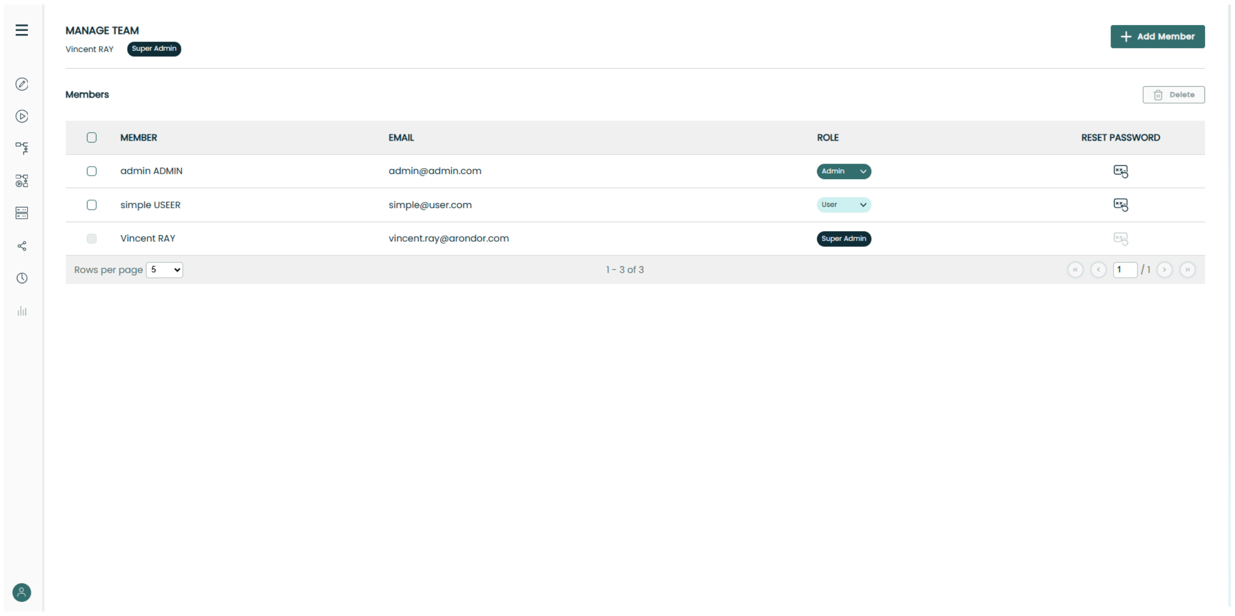
Changing role as Admin

- Can only promote or demote **Users**.
- Role change for other Admins or Super Admins is **not allowed**.

Resetting a member's password

- Available in the Members table via **Reset Password** button.
- Confirmation popup appears before action.
- Success notification confirms the password reset.

- Do not forget to send the new password to the user.


MANAGE TEAM
Vincent RAY Super Admin

Members

	MEMBER	EMAIL	ROLE	RESET PASSWORD
	admin ADMIN	admin@admin.com	Admin	
	simple USEER	simple@user.com	User	
	Vincent RAY	vincent.ray@arondor.com	Super Admin	

Rows per page 5 1 - 3 of 3 1 / 1

Getting started / Create a map

WARNING

Fast2 does not check the integrity of workflows built by the user. It is the responsibility of the latter not to build incoherent maps.

The very first step of every migration is to start building the workflow. Since Fast2 is an ETL, the sequence of all tasks needs to meet this philosophy :

1. first you extract/retrieve the content,
2. then you perform any required transformation to get the content compliant with the target system,
3. finally you load the content and/or its metadata.

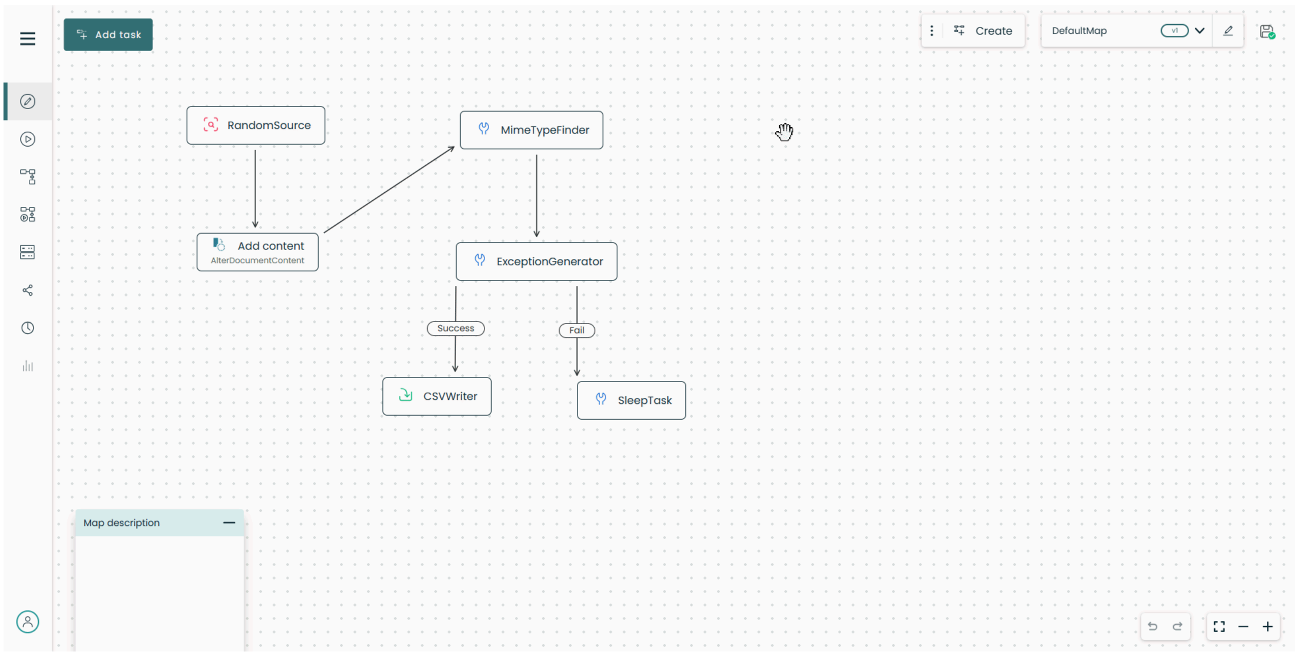
With Fast2 started, go to the UI (default address http://localhost:1789). From here, you can create a new map browse your machine to import an existing one.

WARNING

A map must start with a ***Source*** task, picked up from the list in the task configuration section.

Any change about the map configuration/structure needs to happen in one place only : the design place. Changing the map name, adding or configuring either tasks or links, all these operations will take place in the Design Place.

To add any task to the map, toggle the right panel and search for the task. The dynamic search bar filters out all matching task based either on their name or description. To select a task, click on its name and it will appear in the design area. You will be able to configure it straight afterwards, since the right panel now displays the configuration fields of the task you just added.



For task and link configuration, please head towards the task configuration and link configuration sections.

Set and edit map name

The map name can either be set from the pop-up when creating/duplicating a map, or changed later on.

NOTE

1. A map **name** cannot be empty.
2. Every map must have a single and unique **name**, whatever its version.

When the map is displayed, the name can be updated by directly from the top banner. Once its name edited, the map needs to be saved, otherwise the name will remain as was.

Download a map

Any map can be downloaded directly through the Fast2 UI. Select the map you want then click on download icon located at the top banner.

You can also reach maps from the installation folder of Fast2, in the `/maps` folder.

Fast2 also stores maps into the database instance.

Upload a map

NOTE

When uploading a map locally using the `/maps` folder, only files ending with `.map.xml` will be taken into account. Other files will be ignored.

V1 maps are compatible with the V2, but not the other way around. V2 maps are compatible with the V2025.

Click on the browse icon at the top banner and select the map to upload. Fast2 will automatically switch to the freshly uploaded map.

If you upload multiple times the same map, Fast2 will create a new copy of the map by putting at the end the next suffix `_V-n`, where *n* is the version number of your map.

Delete a map

INFO

Maps can only be deleted from the Configuration Place

To go on the configuration place, click on the gear icon at the top right banner. Use the checkboxes to select the map(s) you want to delete then click on the bin icon.

However, a backup copy is still saved in the database in case you want to restore it later.

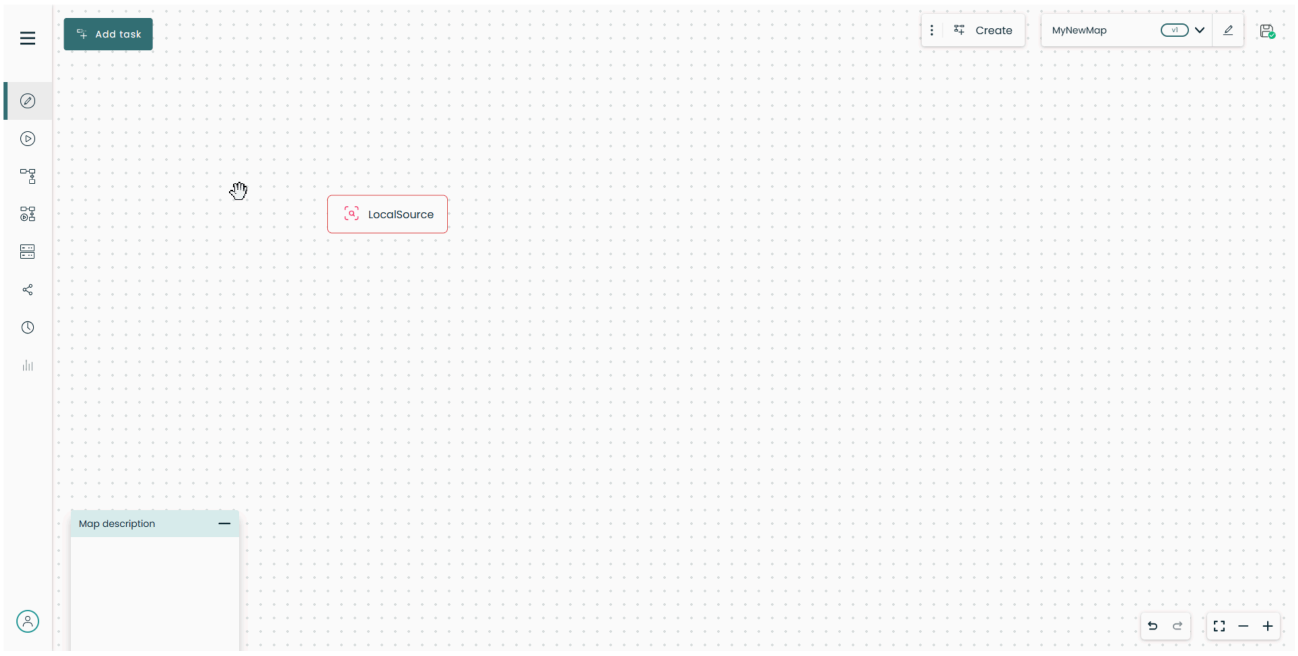
Tasks

INFO

Tasks can only by added, configured and removed from the Design Place.

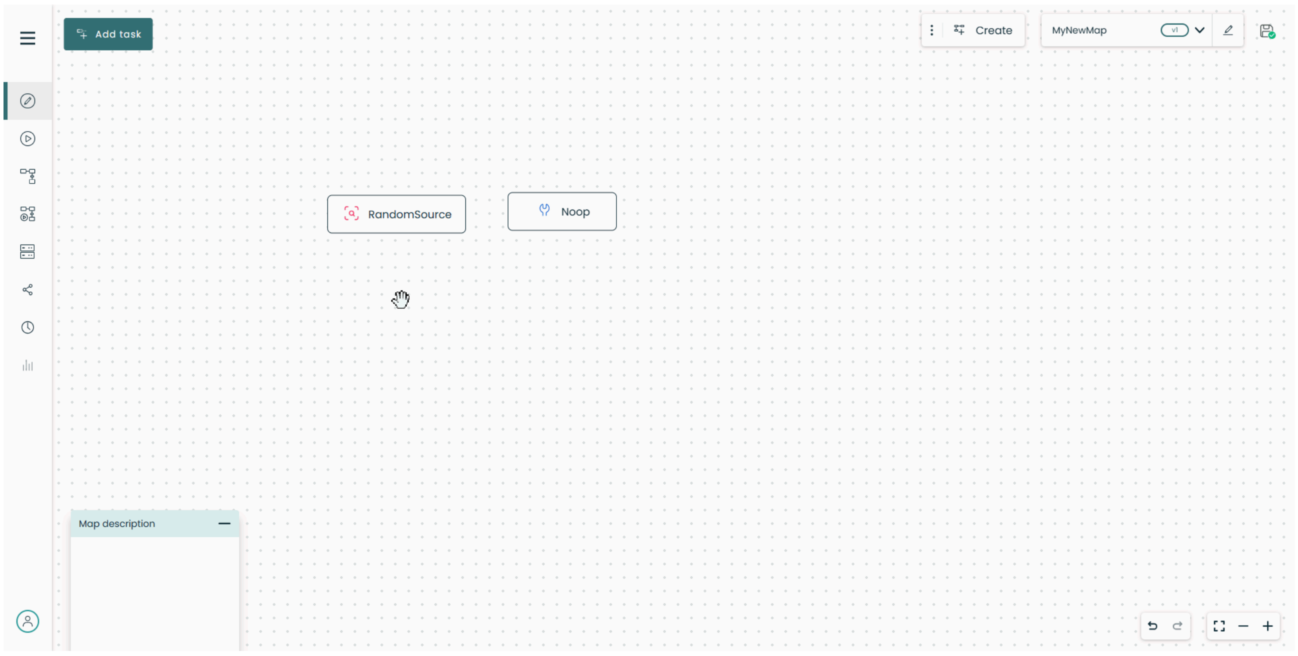
Add task

To add any task to the map, toggle the right panel and search for the task. The dynamic search bar filter all task matching based either on their name or description. Then select a task by clicking on its name. It now appears in the design area. You will be able to configure it straight afterwards, since the right panel now displays the configuration fields of the task you just added.



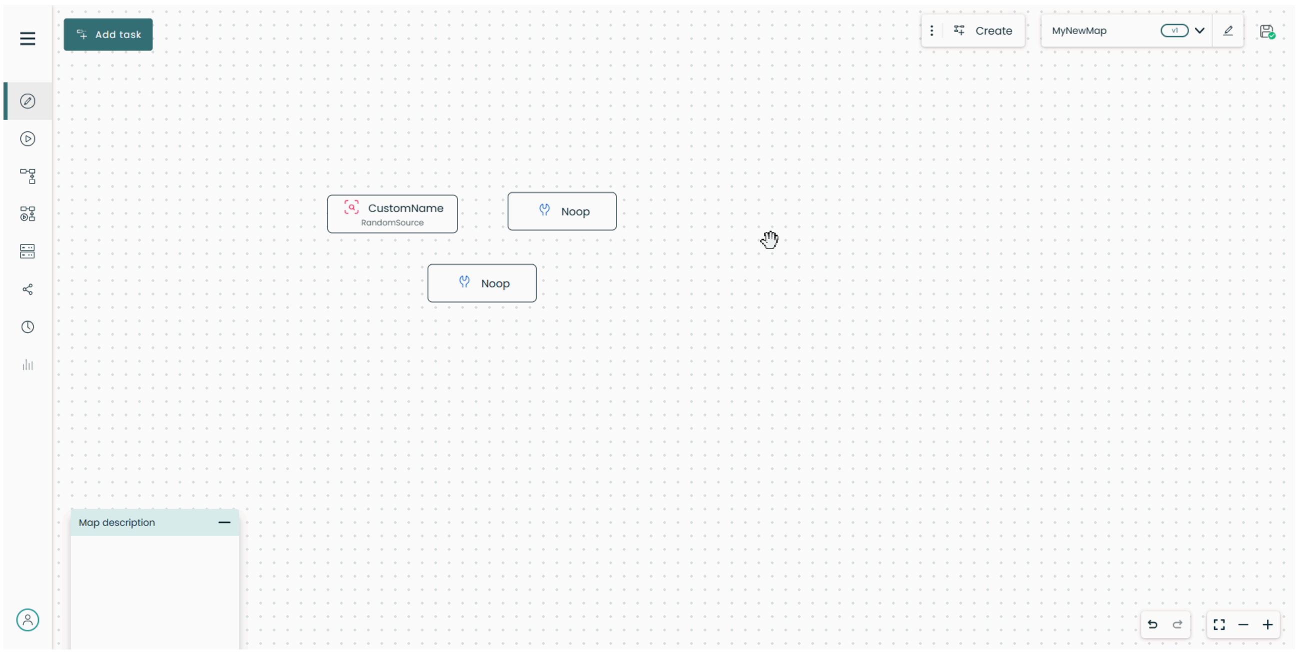
Configure task

To configure a task, hover it in the design area. When it gets dark grey, hit the gear icon : the task is now highlighted and the right panel displays the configuration fields for you to fill.



Delete task

The deletion of a task is a 2-step long procedure. To delete a task, hover it in the design area. When it gets dark grey, click on the trash icon. On the confirmation pop-up, make sure you perform the operation on the task you really intended to delete.



Links

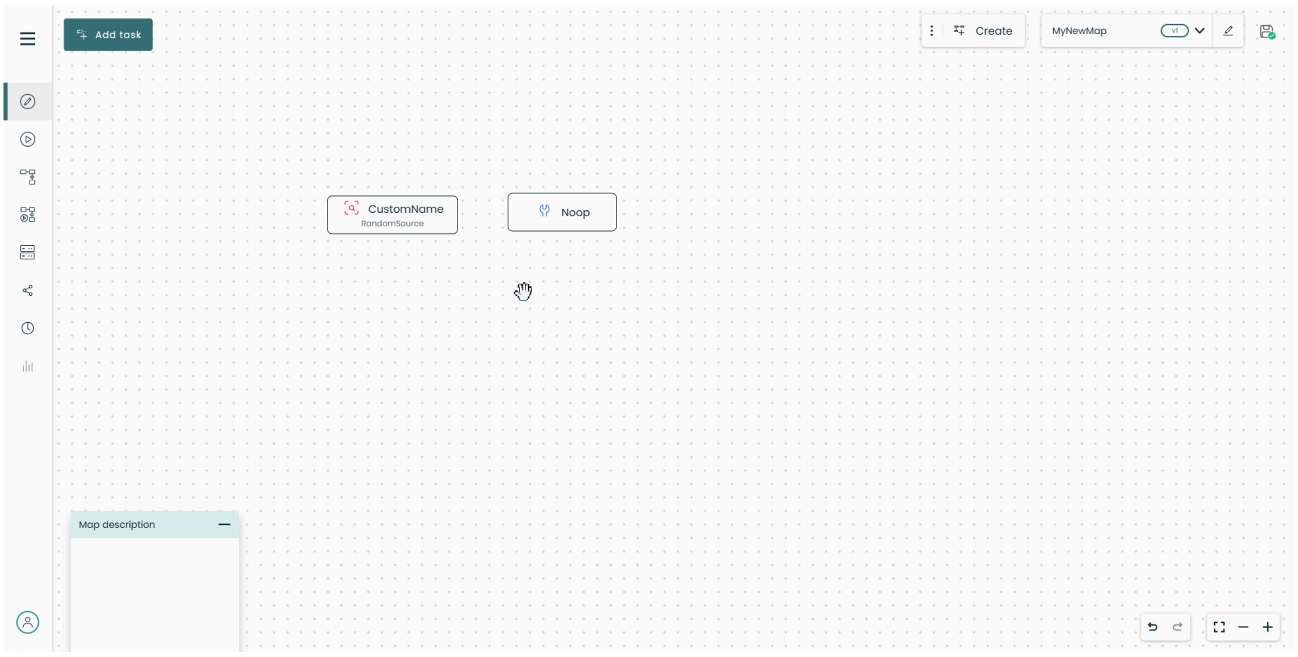
INFO

Links can only be added, configured and removed from the Design Place.

Add link

To add a link between two tasks, click and hold the orange link of the originator, and drag it onto the receiver task widget. One task can have multiple input and output links.

A link can only be added between two tasks. That implies the deletion of any link attached to a specific task once this one is deleted.



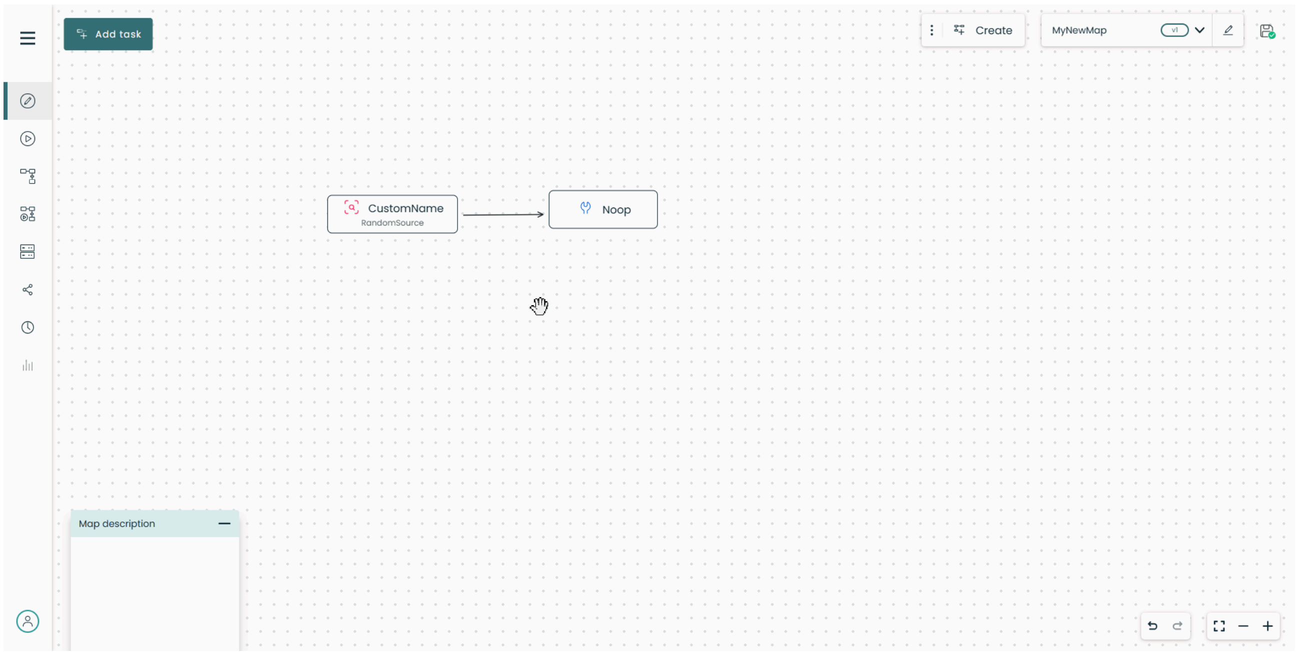
WARNING

Two tasks cannot be linked both ways.

Configure link

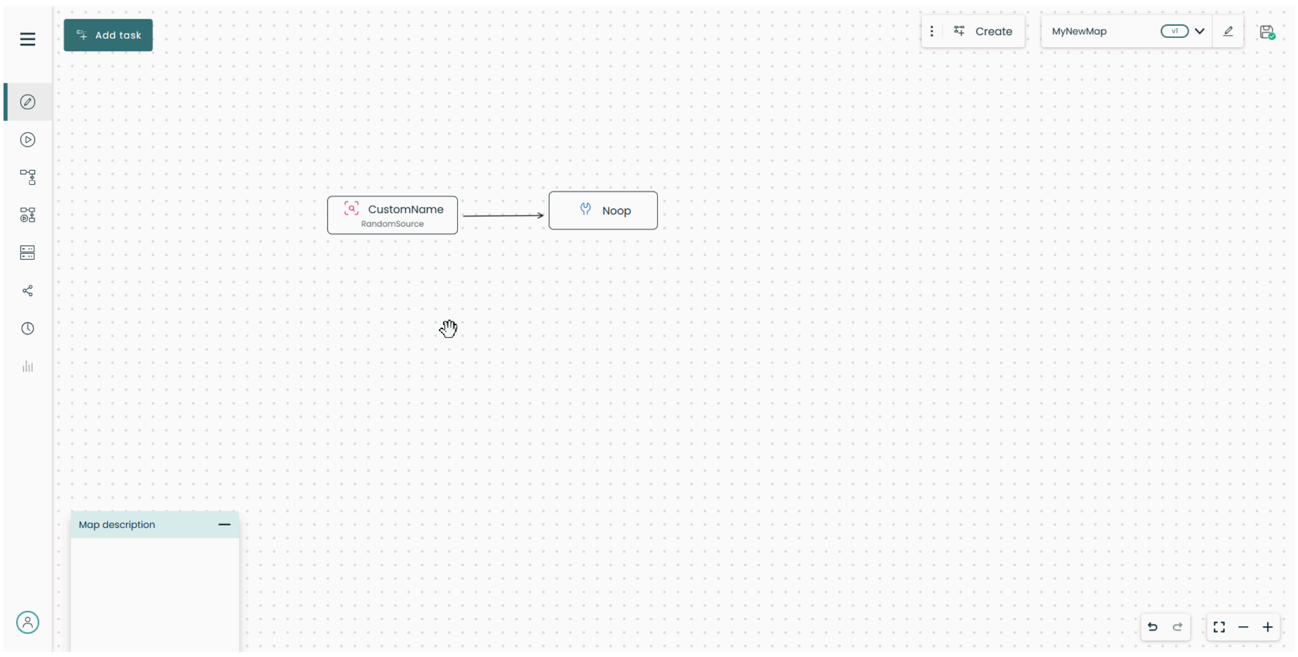
Link condition	Details
PatternCondition	Set specific condition with java language
Otherwise	Punnet which doesn't match other conditions will pass

Link condition	Details
	Consider using when a task has at least 2 multiple output links
AlwaysTrue	All punnets will pass, no matter what
AlwaysFalse	All punnets will be blocked, no matter what
PunnetInException	All punnets in exception will pass
ContentMimeTypeMatches	Filter depending on document mimetype
NumberOfDocuments	Filter depending on the number of document carried by the punnet
PunnetHasData	If an expected data exists the punnet will pass (punnet level)
DocumentHasData	If an expected data exists the punnet will pass (document level)
Or	Use multiple link conditions and if one of them is ok, the punnet will pass
And	Use multiple link conditions and if all of them are ok, the punnet will pass
Not	Use any link condition but with negatively



Delete link

Deleting a link is similar to deleting a task, 2-step long procedure. To delete a link, hover it in the design area. When it gets dark grey, click on the trash icon. On the confirmation pop-up, make sure you perform the operation on the link you really intended to delete.



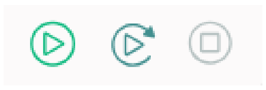
Run a map

INFO

A map can only be executed from the Run Place

From the toggle button you are able to switch between the Design and the Run places.

Next to this toggle button, you'll find the control buttons. It's with these three buttons that you will be able to interact with the campaign.



From left to right :

- *Run as new*:
  When a campaign is run for the first time , a pop-up will ask you to put a

name. Otherwise, a new campaign is created keeping the original campaign name and incrementing the `_Try` number.

- *Rerun*:
  Fast2 will run the same campaign again. No any campaign will be created.
- *Stop*:
  For a running campaign, you have the opportunity to stop its process. Notice that the stop button becomes a *Resume* button once the campaign is stopped.

INFO

A map must be run as new for the first time

When the map has at least been run once as new it is possible to replay a new run over it without necessarily doing as new. To do so, click on the blue play button surrounded by an arrow. Results for all replayed campaigns will be aggregated.

You can see in the table below the allowed actons depending on the campaign status.

Campaign Status	Action(s) allowed
Started	Stop
Stopped	Start as new Resume
Undefined	Start as new Rerun

Campaign Status	Action(s) allowed
Finished	Start as new Rerun

New campaign name

You can create a new series of campaigns with a new name for a given campaign by modifying the campaign name textBox. If the new campaign name does not already exist you are automatically forwarded to the new one and ready to run it. It's mainly a creation than a modification.

You cannot change the campaign name for an already existing one. The modification is aborted and the original name is putted back.

Example

Supposing you have a map named *Production*. You've decided to run this map and click on the `Run as new` button. Put the name of your campaign in the pop-up textBox, for example *prodCampaign*, and click on `Start`.

Fast2 will run your campaign but renaming it in *prodCampaign_Try1*. By hitting the `Run as new` again the pop-up will not show again since the map owns already few campaigns, one actually. Instead, a new campaign will be run and called *prodCampaign_Try2* and so on.

Now, imagine that you want to rename your campaign with the name *newCampaignName*. Click on the campaign textBox, put your new name. You're gonna be redirected to a run place with your freshly created campaign. Hitting the `Run as new` and you will find you're campaign *newCampaignName_Try1* running.

At this moment you have two series of campaigns related to your map *Production*.

Map versioning

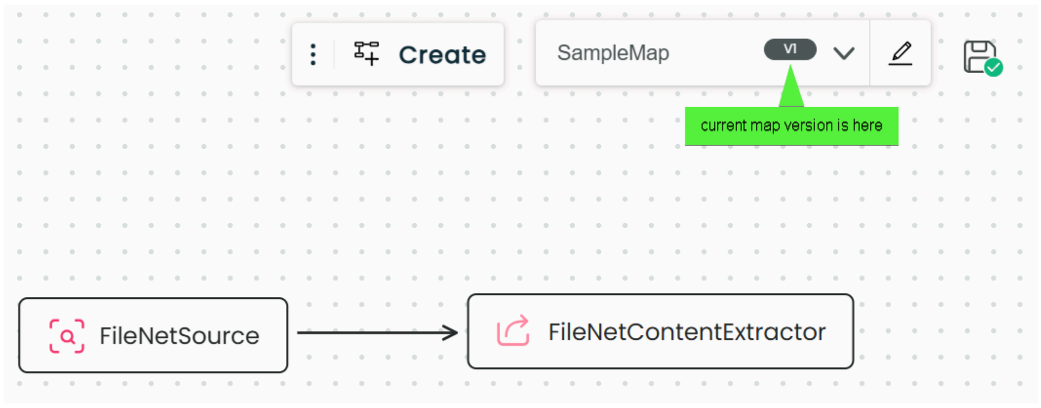
Benefits of versioning

Fast2 allows you to create multiple versions of the same map. This feature provides several significant benefits:

1. **Change Management and Tracking**
   - **Historical Record**: Versioning allows you to maintain a record of all changes made to a map. This is critical for understanding how workflows have evolved over time.
   - **Auditability**: Regulatory or internal compliance often requires a clear audit trail. Versioned maps make it easy to demonstrate changes and decisions.
2. **Flexibility for Iteration**
   - **Testing New Versions**: You can test new map configurations while maintaining the stability of the current version in production. This reduces risk and allows for experimentation.
   - **Rollback Capabilities**: If a new version introduces issues, you can quickly revert to a previous version.
3. **Support for Continuous Improvement**

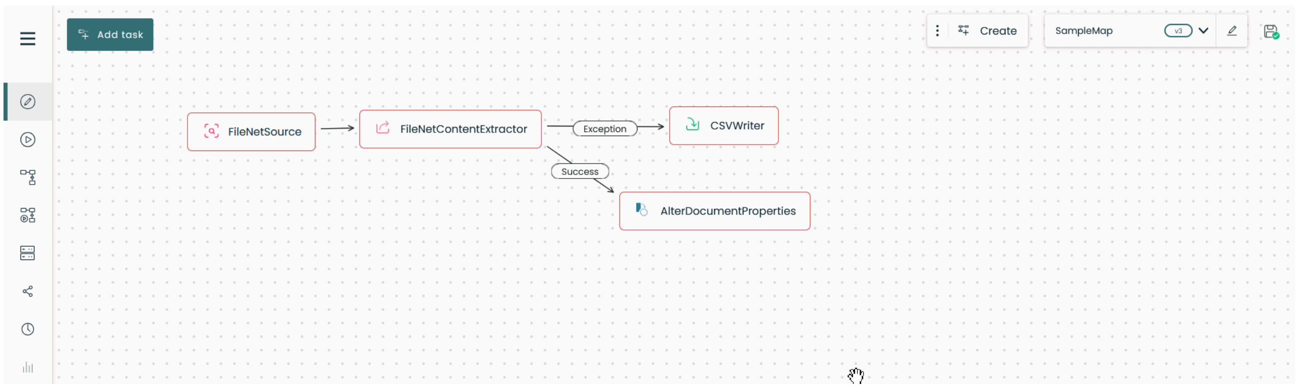
- **Incremental Optimization**: Maps can be improved incrementally while keeping a reliable baseline version in production.
- **Data-Driven Updates**: Analyze performance data from different versions to identify which version works best.

Seamless version creation for the user: they have nothing to do and cannot create a version themselves (there is no manual version creation). The version number is incremented automatically. Current version you are working on is always available in the top right corner.

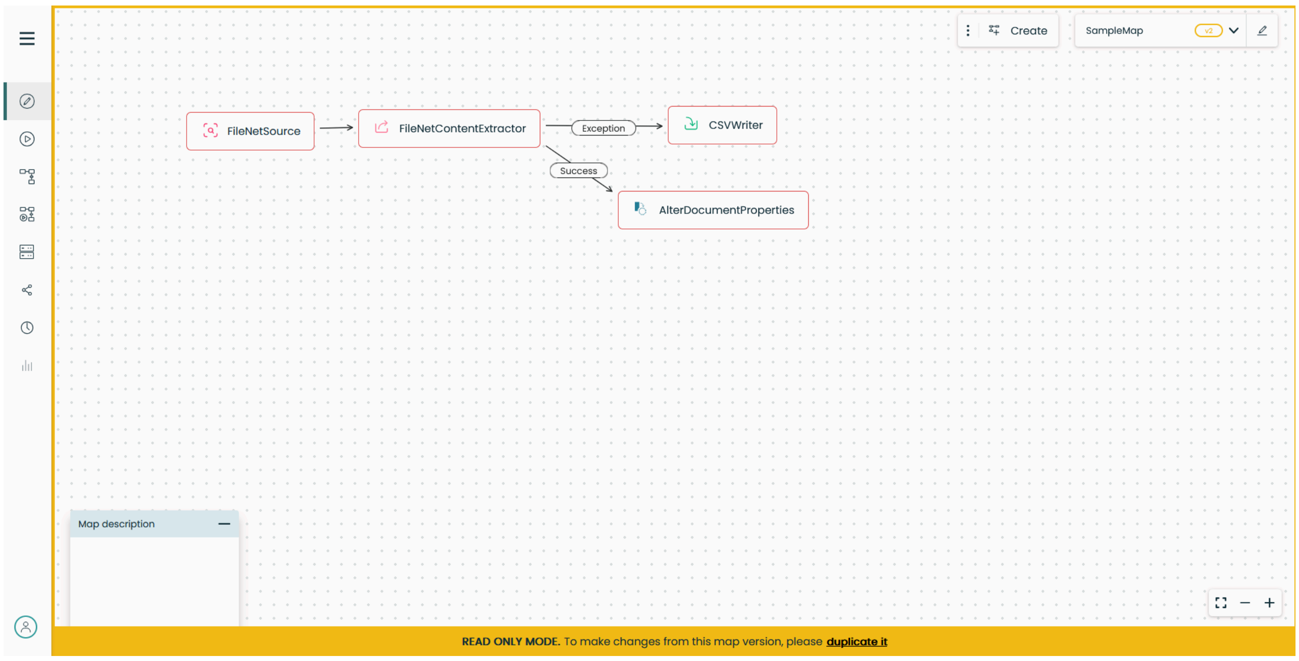


Map versions history

You can access the map versions history by clicking on the `Maps Overview` button in the left navigation menu. If a map has several versions, you can expand and see all the versions by clicking on the last version. Note that current version and previous versions are highlighted differently (green and orange).

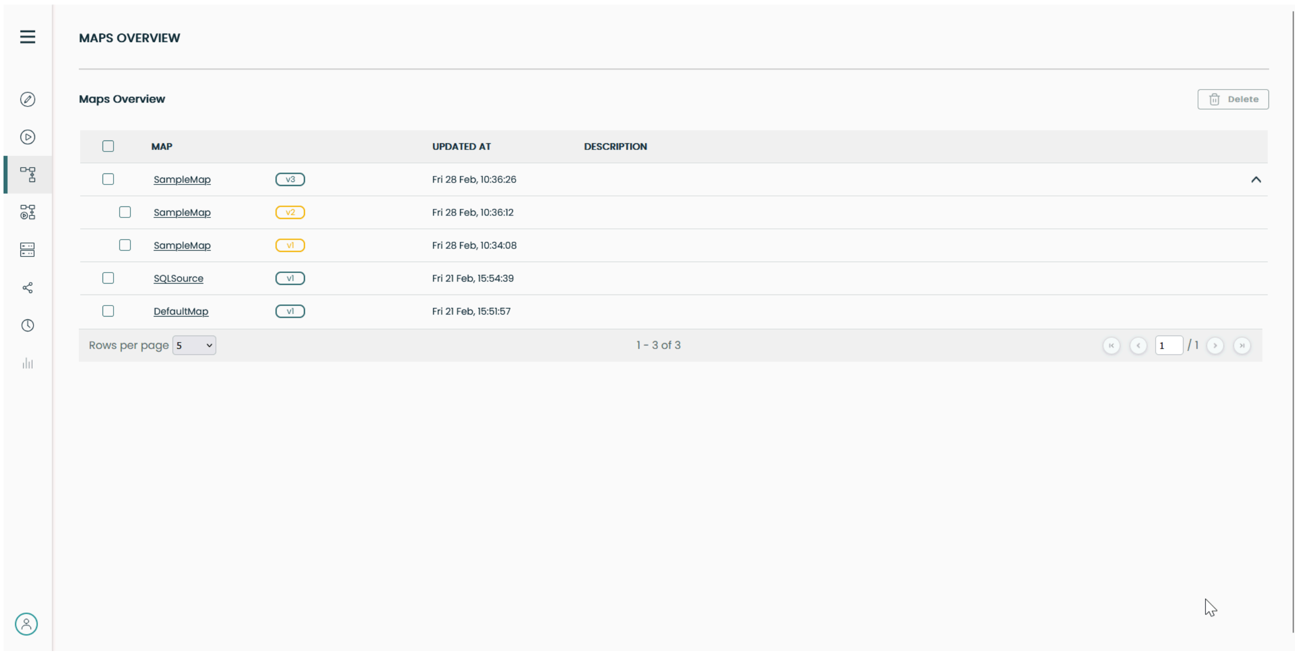


Orange color for previous versions means that they cannot be edited. If you decide to view a previous version, it will be opened in read-only mode.



READ ONLY MODE. To make changes from this map version, please duplicate it

For any reason, if you need to work and make changes to a previous version, you can duplicate it and create a new map from it.



MAPS OVERVIEW

Maps Overview

Delete

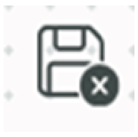
MAP		UPDATED AT	DESCRIPTION
SampleMap	v3	Fri 28 Feb, 10:36:26	
SampleMap	v2	Fri 28 Feb, 10:36:12	
SampleMap	v1	Fri 28 Feb, 10:34:08	
SQLSource	v1	Fri 21 Feb, 15:54:39	
DefaultMap	v1	Fri 21 Feb, 15:51:57	

Rows per page 5 1 - 3 of 3 1 / 1

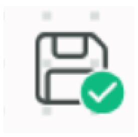
Automatic Save Feature

The automatic save functionality ensures that changes made to the workflow and its configurations are saved seamlessly, enhancing reliability and reducing the risk of data loss. Below are the differents statuses of the save button:

- **Not saved**: The map has not been saved yet (Opensearch database is not reachable).



- **Saved**: The map has been saved successfully.



- **Saving…**: The map is currently being saved.

When Does Auto Save Trigger?

The auto save is triggered under the following conditions:

1. **Configuration Changes:**
   - Any changes made to tasks or links configuration fields are saved as soon as the focus is lost from the edited field.

2. **Tasks and Links:**

   - Adding or deleting a task triggers an automatic save. Adding or deleting a link between tasks also triggers an automatic save.

3. **Task Movements:**

- Moving tasks within the map also triggers an automatic save.

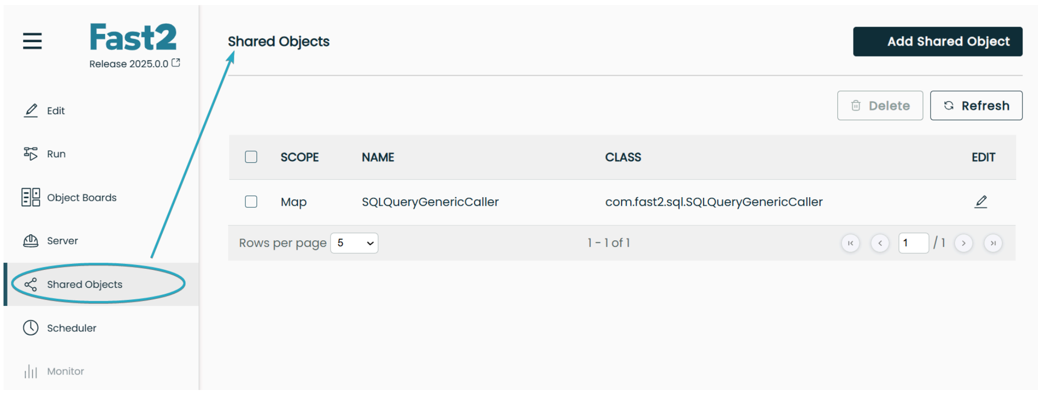
What Gets Saved?

The following elements are included in the automatic save process:

- The **tasks** within the map, including their **configuration settings** and **positions**.
- The **map name**.
- The **shared objects** within the map.
- The **links** between tasks.

Specific Case: Map Shared Objects

Modifications to a shared object within the Shared Objects place also trigger an automatic save, not limited to edits made in the main Edit place.



SCOPE	NAME	CLASS	EDIT
Map	SQLQueryGenericCaller	com.fast2.sql.SQLQueryGenericCaller	

Rows per page 5 1 - 1 of 1 1 / 1

This ensures that all updates, regardless of where or how they are performed, are reliably captured.

Components

Here's a quick intro explaining the purpose and role of the four main components in Fast2 :

👨‍🍳 Broker

The Broker in Fast2 serves as the migration orchestrator, managing the entire document migration process. It delegates unit tasks to the embedded worker or additional workers for scalability. By coordinating the workflow, the Broker ensures efficient and reliable migration operations.

Worker(s)

The workers in Fast2 are responsible for executing the delegated migration tasks. They handle the actual processing of documents, applying transformations, conversions, and other necessary operations. Additional workers can be added to distribute the workload, enhancing performance and enabling scalability.

Database

Fast2 utilizes a NoSQL database to store all migration-related information. This database ensures crash-proof functionality, as well as providing traceability and persistence throughout the migration process. By storing the data in a structured manner, the database facilitates efficient retrieval and management of migration artifacts.

Dashboards

Fast2 offers a powerful dashboards feature that provides users with comprehensive visualizations and graphs. These dashboards give an overview of the overall migration progress, even when dealing with large-scale migrations encompassing millions of documents. Users can monitor key metrics, track performance, and gain valuable insights into the migration status, facilitating effective decision-making.

With the combined functionality of the Broker, Worker(s), Database, and Dashboards, Fast2 offers a robust and scalable platform for streamlined document migration, ensuring a smooth and efficient migration experience.

Components / The broker

TIP

The broker is the workflow orchestrator, in charge of database communication, sending punnets to the worker(s) for them to process the operations.

Configure the broker

Depending on the amount of documents you are dealing with, you may want to control max memory usage allowed (Xmx) for broker.

By default, only 1GB is allocated for this resource :

/config/env.properties

```
...
# Broker Maximum memory allowed (Xmx)
BROKER_MAX_MEMORY=1G
```

If the campaign are involving a couple of millions of documents, increasing this value to 8GB or 16GB will definitely help increasing the performance rate of the migration.

Configure the UI port

The UI port is also subject to configuration.

Fast2 application run on the 1789 port by default. To change this, add or update the parameters below:

/config/application.properties

```
...
# Remote broker port to use by the worker
# broker.port=1789
server.port=1789
```

Components / The worker

TIP

The Worker is the punnet processor, applying the changes onto the punnet, according to how the tasks have been configured by the user.

The workers! Corner stones of Fast2, these guys can litterally add up and constitute a real digitized hive working to migrate your documents, your contents, your rules, your metadata, all synchronously, exactly where you expect them (or asked them to be), never stepping on each other. No migration project could be overcome if it was not by them!

If their role can seem quite important, they are paradoxically as easy and straight forward to get up and running. Just the right files to gather, as mentioned here, and a new worker just enrolled!

One of the major aspects of a promising migration project is what all project managers will ask you to vouch for: performance metrics. Let’s suppose you need to migrate documents from a source system into a second one, the latter having a much higher input flow tolerance. No need for empirical statistics to know that the old ECM will be the bottleneck. An architecture similar to a hybrid deployment variant (topic we presented here) could easily be envisioned. But let’s complicate things a little bit here: in-between the extraction and the injection phase, the metadata have to be updated, with new date formatting and heavy mapping of document related properties. Can still a hybrid-like architecture save you now ?

Configure the worker(s)

The required files for the worker to run properly are the following:

Item	Purpose
`config/*`	Configuration files, broker endpoint etc
`worker-libs/*`	All libraries and dependencies for tasks executions
`fast2-worker-package-X.Y.Z.jar`	Worker main unit
`startup-worker.bat`	Binary file for Windows
`startup-worker.sh`	Binary file for Linux

Change JDK

If you want to use a different jdk version than the one referenced in JAVA_HOME environment variable, you can update the `JAVA_HOME` value in `./config/env.properties`:

./config/env.properties

```
...
# Override JAVA_HOME environment variable
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Memory allocation

Depending on the amount of documents and the number of tasks you are dealing with, you may want to control max memory usage allowed (`Xmx`) for worker.

The default setting is 1GB for this resource:

./config/env.properties

```
...
# Worker Maximum memory allowed (Xmx)
WORKER_MAX_MEMORY=1G
```

Keep in mind that this property is designed for workers started from the binary `start-worker.sh|.bat`. If you intend to target the embedded worker, go to `./config/application.properties` instead:

```
...
# Broker embedded worker max memory
broker.embeddedworker.max.memory=1G
```

Queues management

Queues have to be declared to the workers for them to process the punnets stored in these sames *queues*.

The queues names will also be declared in the tasks configuration panel, so the only worker in charge of executing a task with a defined queue will be the worker whose queue regex matched this very queue.

In order to have specific worker tied to particular queues, the configuration needs to be updated here:

./config/application.properties

```
# Worker queue regex filter
# worker.queue.regex=.*
```

Disabled the embedded worker

In case serveral workers are required for specific queues and tasks, there might be no more need of the embedded worker itself. To make sure not to have it running pointlessly, this worker can be disactivated from the `./config/application.properties` files, as so :

./config/application.properties

```
# Disable auto-launch of embedded worker
broker.embedded.worker.autostart=false
```

Advanced use

One of the major aspects of a promising migration project is what all project managers will ask you to vouch for: performance metrics. Let’s suppose you need to migrate documents from a source system into a second one, the latter having a much higher input flow tolerance. No need for empirical statistics to know that the old ECM will be the bottleneck. An architecture similar to a hybrid deployment variant (topic we presented here) could easily be envisioned. But let’s complicate things a little bit here: in-between the extraction and the injection phase, the metadata have to be updated, with new date formatting and heavy mapping of document related properties. Can still a hybrid-like architecture save you now ?

Several workers

Context

One of the major aspects of a promising migration project is what all project managers will ask you to vouch for: performance metrics. Let’s suppose you need to migrate documents from a source system into a second one, the latter having a much higher input flow tolerance. No need for empirical statistics to know that the old ECM will be the bottleneck. An architecture similar to a hybrid deployment variant (topic we presented here) could easily be envisioned. But let’s complicate things a little bit here: in-between the extraction and the injection phase, the metadata have to be updated, with new date formatting and heavy mapping of document related properties. Can still a hybrid-like architecture save you now ?

How to set up

Checkout in the official documentation the required Fast2 files and folders to set up a new worker. Leave a copy of the required files and folders on the machine hosting the source environment. This worker -- let’s label it as worker-S for “source” -- will be assigned to the extraction part. As indicated in the installation section, starting Fast2 will launch an embedded worker, assigned by default to all tasks composing the migration workflow. This worker here will be our worker-D (for “Destination”, or “Default”).

Plug the worker-S onto the Fast2 broker (yes, the workers -- as illustrated here -- manifest themselves to the broker, and not the other way around) : to do so, open the `config/application.properties` of the worker-S :

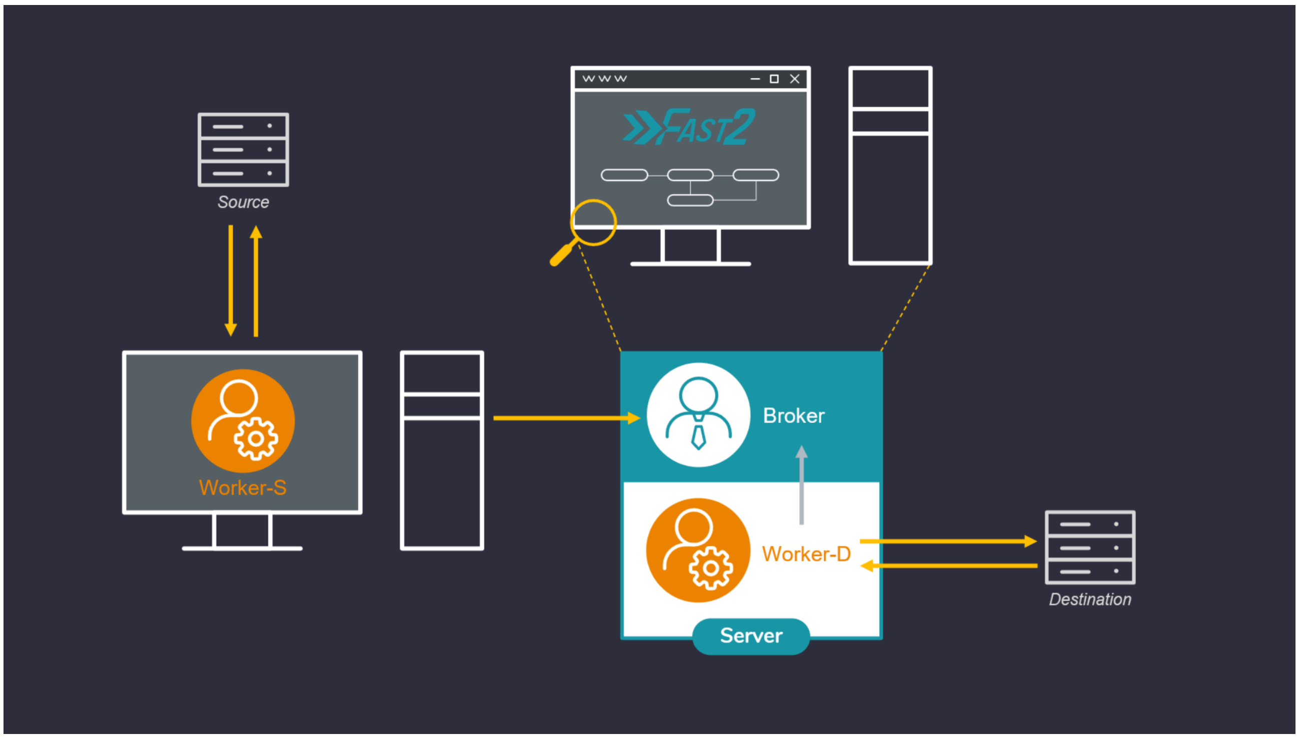
v2.4- v2.5+

./config/application.properties

```
# Fast2 2.1.0 configuration

# Remote broker host to use by the worker
broker.host=localhost

# Remote broker port to use by the worker
# broker.port=1789
...
```

Update the name (or IP address) of the machine where Fast2 is running (`broker.host`), and the name of the queue which the worker will be assigned to (ex/ “extraction”).

1. Start now the Fast2 server (documentation here) to have it up and running alongside the worker-D. This latter will be assigned to both the *mapping* of the metadata and the injection of the documents in the destination environment.
2. Then start the worker-S (documentation here).

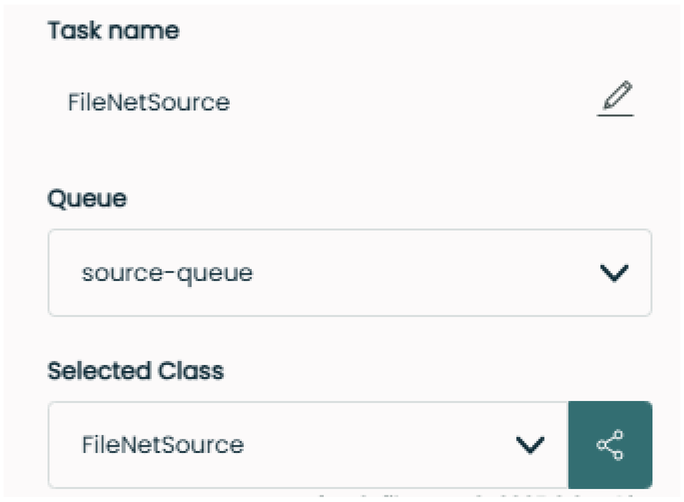


Open your browser to reach the Fast2 UI, and then build up your migration workflow. For the sake of this example, 3 tasks only will suit our needs of extraction, metadata transformation and load.

> 3 tasks, 3 queues, 2 workers: lock and load !!

The extraction task will be linked to the same queue we mentioned in the `config/application.properties` of the lone worker (ex/ source-queue).

No need to set a queue for the last task, as it will be handled by default by the last worker started with the Fast2 server.


Task name

FileNetSource

Queue

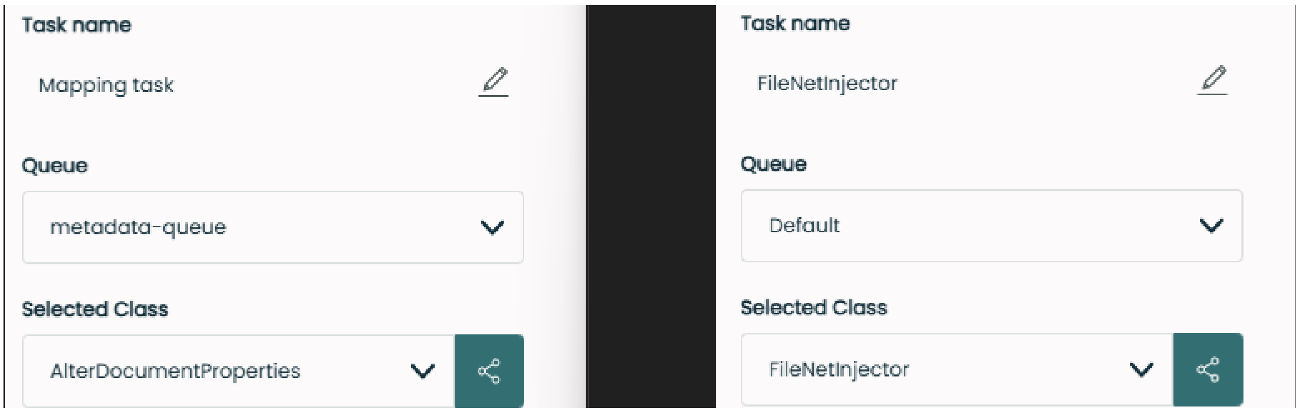
source-queue

Selected Class

FileNetSource

For this worker, the `config/application.properties` will have the queue details set as follows:

```
...
# Worker queue regex filter
worker.queue.regex=source-queue
```


Task name

Mapping task

Queue

metadata-queue

Selected Class

AlterDocumentProperties

Task name

FileNetInjector

Queue

Default

Selected Class

FileNetInjector

For this worker, the `config/application.properties` will have the queue details set as follows:

```
...
# Worker queue regex filter
worker.queue.regex=metadata-queue,default
```

Limits

Just like any architectural decisions, such model comes with its drawbacks and benefits. If the benefits can sound quite obvious given past explanations, the downsides are worth mentioning. We will shortly discuss here about the two most current:

- Resource sharing: the more workers you’ll start on the same machine, the less they’ll have individually available resources.
- Connections and sessions: duplication of workers induces duplications of server calls, therefore opened sessions.

Resource sharing

Let’s consider a migration server with 8GB of RAM (which is a pretty good start, don’t get me wrong): with a running database in the background — the embedded Elasticsearch instance which Fast2 relies on in terms of traceability — needing roughly 3GB, the operating system using 3GB as well, you’ll end up with only 2GB for your worker to open around 100 documents per second and performing content conversion, metadata transformation etc. Needless to say, adding a second worker won’t help you much here !

Increase the threads amount (which you can do in the server configuration, straight from the Fast2 UI) of the queues on which you worker will get the punnets to process will surely be the go-to way for increasing your current performances.

The most recommended scale-up here would be to start another worker on a different machine, using totally independent physical resources and combining them to the ones already solicited by the Fast2 server.

Connections and sessions maxout

A second non-negligible aspect is the number of connections and sessions opened by the workers to communicate with both the source and destination environments. Adding worker will consequently increase these numbers, especially if several threads have been allocated to their processing queues.

And what about...

Several workers on the same machine?

One easy application of multi-worker architecture could be the need of having several source system to extract documents from, via dedicated maps for each. Booting up several workers associated with the right task queues will provide sufficent segmentation to have your migration happen simultaneously.

Sessions conflicts can be prevented as well but such choices of architecture. At the end of the day, only one Fast2 server will have been managing all your different workflows, all you data will be stored in the very same Elasticsearch database, all with significantly better performance rates!

Several workers on the same queue?

In case of seeking for more physical resources for your Fast2 server which, let’s say, is not a scalable machine, you could envision to “plug” a second server to the first one: start another worker on the second machine, and have it aim to the initial Fast2 server where the broker is running. This separate worker will be able to process any task of your workflow, any queue as well, just like the embedded one.

However there would be absolutely no point in starting another worker assigned to the same queues as the embedded one on the Fast2 server. That won’t positively affect you performance rates. If that was you goal before scrolling this page, the secret relies in adding more threads to your queues (as mentioned earlier)!

Remote worker: Configuration Guide

This guide explains how to configure a remote worker to your broker. It covers both scenarios: when both applications are on the same network and when they are on different networks.

Prerequisites

- Java Development Environment: Both the worker and broker applications should have at least a jdk8+ available on their environment. We highly recommend a jdk11.
- Network Connectivity: Both systems should be able to connect to each other through the network (whether local or remote).

Remote worker config

```
server.host=<broker_ip_address>

# Remote = docs ends broker side
# Local = docs ends worker side
worker.content.factory=<remote|local>
```

Network

Same Network Scenario

- Local IP Address: The worker and broker should be able to communicate over their local network using their local IP addresses.
- Network Configuration: The local network should not have strict firewall rules that block communication on the required ports.

Different Networks Scenario

- Public IP Address of Broker: The broker should have a public IP address, or at least a static public IP from the router.
- Port Forwarding on Router: The router connected to the broker must have port forwarding configured to forward incoming traffic on specific ports to the broker’s local IP address.
- Firewall Configuration: The firewall on both systems (worker and broker) should allow incoming and outgoing traffic on the required ports.
- Dynamic DNS (Optional): If the public IP address of the broker is dynamic, you may want to use Dynamic DNS (DDNS) to avoid manually changing the address every time it changes.

Configure Worker and Broker on same network

Step 1: Ensure Network Connectivity On the worker machine, ensure that you can ping the public IP address of the broker. You may need to test it by pinging broker_public_ip_address.

```
ping <broker_public_ip_address>
```

If the ping works, proceed to the next step. If the ping does not work, there may be an issue with the router, firewall, or routing configuration.

Step 2: Update the broker.url in the Worker Configuration On the worker machine, update the **server.host** property in your *config/application.properties* file to the local IP address of the broker.

If needed, you can change the protocol and port information as well. The **broker.url** variable is automatically updated. Do not change it.

```
server.protocol=http
server.host=<broker_local_ip_address>
server.port=1789

broker.url=${server.protocol}://${server.host}:${server.port}/broker
```

Make sure the port is open and the broker is listening on the specified port.

Step 3: Verify broker is listening on specified port On the broker machine, verify that the broker application is listening on the port you specified by using the following command:

```
# Linux
sudo netstat -tuln | grep <port>

# Windows
`netstat -ano | findstr <port>`
```

The output should show something like:

```
tcp6       0      0 :::<port>                 :::*
LISTEN
```

Step 4: Test the connection On the worker machine, test the connection to the broker using nc (netcat) to check if the port is open and accessible:

```
# Linux
nc -zv <broker_local_ip_address> <port>

# Windows
telnet <broker_local_ip_address> <port>
```

If the connection is successful, the worker and broker can communicate.

Configure Worker and Broker on different networks

Step 1: Configure Port Forwarding on the Broker’s Router On the router connected to the broker, you need to configure port forwarding to forward incoming traffic on a specific port to the broker's local IP address and port.

1. Log into the router's web interface (usually at 192.168.1.1 or 192.168.0.1).
2. Navigate to the Port Forwarding or NAT settings section.
3. Add a rule to forward traffic coming on port to the internal IP address of the broker (broker_local_ip_address).
4. Save the settings.

Step 2: Verify Firewall Configuration Ensure that both the broker’s firewall and the worker’s firewall allow communication on the specified port. If necessary, open the required port in the firewall:

On Ubuntu, to open a port in the firewall (if using ufw):

```
sudo ufw allow <port>/tcp
```

Step 3: Repeat steps explained for same network

Remote worker configuration

You have multiple options through the application.properties file to configure your remote worker.

File storage : broker or worker ?

You can either store the files processed from the broker or at the worker side. To choose one or the other you simply have to modify this property :

```
worker.content.factory=<remote|local>
```

- Select **remote** to send back documents to the broker.
- Select **local** (default value) to keep documents processed by the worker from its side

Example

This is an example to understand what happens for both scenarios. Imagine that we are extracting some documents from a Documentum environment and we need to convert tiff files to a pdf format.

Local

Remote

File storage architecture

By default, documents processed by the worker will be stored under the folder **files/**. Then documents will follow a strict hierarchy as mentioned in the property **worker.files.pattern**

```
worker.files.dir=files/
worker.files.pattern=@{campaign?:'shared'}/@{step?:'shared'}/@{docume
```

Values shown above are used by default. Feel free to change it to match your requirements in term of folder organization.

Troubleshooting

Common Issues

Ping does not work

Ensure that the devices can actually communicate over the network. Double-check the network cables, Wi-Fi connection, and make sure there are no misconfigured network settings or firewalls blocking ICMP packets.

Connection times out

If using public IP addresses, check the router’s port forwarding configuration and verify that the firewall on both the broker and worker machines allows traffic on the relevant port.

Port is closed

Verify that the broker application is actually listening on the specified port, and ensure the port is not blocked by a firewall.

Public IP changes

If the public IP of the broker changes frequently, consider using a Dynamic DNS (DDNS) service to map a domain name to the changing IP address, so the worker can use the domain name instead of an IP address.

Components / Internal database

> **WARNING**
>
> Prior to the v2.5, Fast2 was relying on an Elasticsearch database. This component has been dropped in favor of OpenSearch.
>
> However the configuration of these two databases are very close (if not identical).

Every object passing through Fast2 is stored into an internal database. Whether carried by the document or the punnet, all metadata are recorded in the warehouse. The major benefit of such architecture is the opportunity to check whether everything is going well by making counters about documents/data processed during your migrations.

In addition, we can easily rollback or resume operations in case of server crash. Nothing will be lost as Fast2 will precisely know where it all stopped.

There is the logic behind real-time backups in ES.

Indexes

Each database index referenced by Fast2 will be registered with a `f2_` prefix. An index is always written in lower case even if the campaign name in Fast2 contains characters in upper cases.

For example, the campaign `MyCampaign_Try10` will be stored in the index f2_mycampaign_try15.

During the step of broker intantiation at Fast2 startup, some indices are automatically created:

Index key	Description
f2_campaigns	List of existing campaigns with processing dates and status
f2_campaigns_sources	Links between campaigns and workers having performed this campaign
f2_queue_settings	Reference information about source and task threads
f2_jobs_settings	Gather the configuration of save jobs
f2_jobs_info	Information about jobs past execution details

For each new campaign of Fast2, an index will be created: if you decided to run a new campaign named `EcmInjection`, the new index will be `f2_ecminjection_try1`.

Configuration

With or without

For an optimal migration setup, this third-party software can be easily configured at different levels to match you needs at most ! If required, it can even be disabled at will.

v2.4- **v2.5+**

./config/applications.properties

```
broker.elasticsearch.embedded.enabled=true
```

Port

By default, Fast2 sends it data via the embedded database API made available on port `1790`.

However, in the case where this port is already used by either another Fast2 instance or any other process, the port number can be changed from the configuration files.

Since the embedded database has to be reach from both Fast2 broker and Kibana module — if the latter is enabled — there is exactly 3 places where to mention this change:

v2.4- **v2.5+**

File	Specification
./config/application.properties	`opensearch.port=1790`
./elasticsearch-X.Y.Z/config/elasticsearch.yml	`http.port: <es-port>`

File	Specification
./kibana-X.Y.Z/config/kibana.yml	`elasticsearch.hosts: ["http://localhost:<es-port>"]`

If the dashboard component is installed, the database port also needs to be updated on this front as the dashboard needs to access the DB in order to read the data :

v2.4- v2.6+

./kibana-X.Y.Z/config/kibana.yml

```
elasticsearch.hosts: ["http://<DB-server:DB-port>"]
```

Memory

The more documents, the more data. The more data, the more the database will need resources to digest, store, process data and respond to the broker.

Head out to the `./opensearch-X.Y.Z/config/jvm.options` file.

The configuration required are the following:

Configuration	Purpose
`-Xms8g`	This setting will allocate 8GB of RAM to the database JVM heap, directly on startup.
`-Xmx8g`	Here, you specify the maximum memory which can be used, if available, by the database.

WARNING

As specified in the `./opensearch-X.Y.Z/config/jvm.options` file, you should always set the min and max JVM heap size to **the same value**.

See the Official OpenSearch documentation for more information.

For further comprehension of these parameters, check out the Elasticsearch official documentation on the topic or OpenSearch official documenation. Upgrading the metrics will prevent `java.lang.OutOfMemoryError` to pop up during heavy migration executions.

Remote access to the database

The next operations need to happen when the database is shut down. To make sure of that, the `jcmd` command might be of great help.

WARNING

The database port needs to be opened from the Fast2 server, and accessible by your remote machine.

1. To check the database port is accessible from your machine, run the following command (from your work station):

   ```
   curl <fast2.server>:<database-port>
   ```

2. Head out to the database configuration YAML file `opensearch.yml` and add the following lines:

   ```
   network.host: 0.0.0.0
   node.name: node-1
   cluster.initial_master_nodes: node-1
   ```

3. If Fast2 is installed on a Linux server, you may also need to increase the memory usage for your cluster (on the server where the database is running), as stipulated in the Official OpenSearch documentation.

   /etc/sysctl.conf

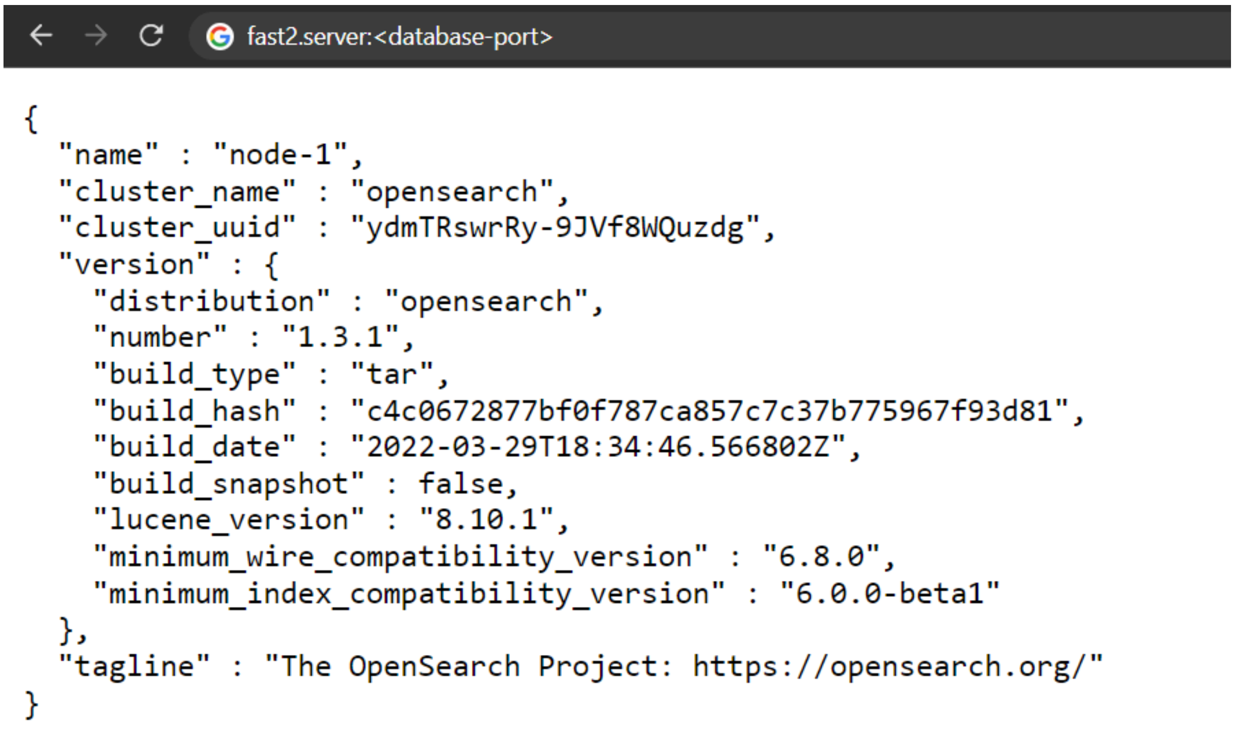
   ```
   # for remote access to Fast2 embedded database
   vm.max_map_count=262144
   ```

4. Save the file, and run the following command to *refresh* your server configuration :

   ```
   sudo sysctl -p
   ```

5. Restart the broker (which will induce the bootup of the database), and go check from your machine the access to the Fast2 database.

   The webpage (at the same URL we `curl`-ed in the 1st step) should display something like so :



fast2.server:<database-port>

```
{
  "name" : "node-1",
  "cluster_name" : "opensearch",
  "cluster_uuid" : "ydmTRswrRy-9JVf8WQuzdg",
  "version" : {
    "distribution" : "opensearch",
    "number" : "1.3.1",
    "build_type" : "tar",
    "build_hash" : "c4c0672877bf0f787ca857c7c37b775967f93d81",
    "build_date" : "2022-03-29T18:34:46.566802Z",
    "build_snapshot" : false,
    "lucene_version" : "8.10.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "The OpenSearch Project: https://opensearch.org/"
}
```

Troubleshooting

In reason of the tight commmunication between the broker and the database, chances are you will soon be reported 500 server error generated by unsuccessful exchanges of the two entities.

Server error 500 when starting a campaign

After running quite a bunch of campaigns, you might end up not being able to start anymore of them due to the limit of shards of the embedded database (for more in-depth details about the shards, checkout the Official Elasticsearch documentation.

The symptom of this limitations comes as a regular 500 server error toast in the UI, but is is by checking the logs/broker.log file that its raw nature is exposed:

```
17:24:45.017 [http-nio-1789-exec-17] ERROR org.apache.catalina.core.C
[Tomcat].[localhost].[/]:175 - Exception while dispatching incoming R
com.google.gwt.user.server.rpc.UnexpectedException: Service method 'p
com.fast2.model.taskflow.Campaign
com.fast2.hmi.gwt.client.service.GWTCampaignManager.startProcessing(

com.fast2.model.taskflow.Campaign,com.fast2.model.taskflow.design.Tas
threw an unexpected exception:
    java.lang.RuntimeException: Caught exception OpenSearch exception
[type=validation_exception, reason=Validation Failed: 1: this action 
shards, but this cluster currently has [1000]/[1000] maximum shards o
        ...
Caused by: org.opensearch.OpenSearchStatusException: OpenSearch excep
[type=validation_exception, reason=Validation Failed: 1: this action 
shards, but this cluster currently has [1000]/[1000] maximum shards o
```

As mentioned in the database technicalities, Fast2 records data under indices prefixed with `f2_`. Thus it implies to begin each index to delete with this prefix.

Although a drastic cleanup induced by a `curl -X DELETE -i "http://<database-server>:<database-port>/f2_*"` would resolve our issue, you might be interested in keeping some campaigns or indices. As any *curl* query allows, exceptions can be added to the deletion operation to prevent them from being removed of the backup database. The syntax goes as follows:

```
curl -X DELETE -i "http://<Fast2-server>:<database-port>/f2_*,-f2_campaigns,-f2_campaigns_sources[,-f2_<campaign-name>]"
```

Let us now study this query:

Section	Purpose
`http://server:port/f2_*`	Here we ask to aim at all indices starting with the `f2_` prefix. This will prevent the deletion of additional indices which could be related to parallel work on the same database instance, such as Kibana reports, charts or data analysis.
`-f2_campaigns,-f2_campaigns_sources`	These 2 indices are needed if you decide to keep any other campaign. The `-` sign declares them as exception from the delete action.

Section	Purpose
`[,-f2_<campaign-name>]`	Then you can list all the campaigns you are willing to preserve, • without space, • separated by commas (`,`) • mentioning the `-` exclusion character Do not forget to begin each name with the indice prefix.

Wildcards are supported, therefore an exception written `-f2_mycampaign*` will protect all the campaign with this *myCampaign* radical (ex/ myCampaign_Try1, myCampaign_Try2...).

Impossible to get results from Explorer place

Documents concerned by a migration can quickly add up, especially on cumulative campaigns. When heading to the Explorer to check punnets results, you may end up with too much information to retrieve from the database. An "error 500" message will then pop up, and if yu head out to the broker.log, the following stack will be found:

```
Suppressed: org.opensearch.client.ResponseException: method [POST], h
[http://localhost:1790], URI [/f2_campaign_try1/_search?pre_filter_sh
status line [HTTP/1.1 503 Service Unavailable]
{"error":{"root_cause":[{"type":"too_many_buckets_exception","reason"
too many buckets. Must be less than or equal to: [10000] but was [100
can be set by changing the [search.max_buckets] cluster level
setting.","max_buckets":10000}],"type":"search_phase_execution_except
shards failed","phase":"query","grouped":true,"failed_shards":
[{"shard":0,"index":"f2_stg-100k_try1","node":"4XM6VD2wTfeOSpr2brIs7g
{"type":"too_many_buckets_exception","reason":"Trying to create too m
be less than or equal to: [10000] but was [10001]. This limit can be
the [search.max_buckets] cluster level setting.","max_buckets":10000}
```

To fix this, you need to stop Fast2 (and the database), and add the following line:

./opensearch-X.Y.Z/config/opensearch.yml

```
search.max_buckets: 1000000
```

Let's quickly wrap up, here

This integrated database guarantees data persistence to Fast2. If required, an embedded database can be shared among several Fast2 servers.

Allocated resources should be increased in order to resist charge of production environments.

Components / Dashboards

Since Fast2 stores every single byte of migration information into its internal database, using dashboard capabilities for intelligible and functional reports just comes in naturally.

WARNING

Prior to the v2.5, Fast2 was relying on Kibana for data vizualisation. This component has been dropped in favor of OpenSearch dashboards.

However the configuration of these tools are very close (if not identical).

The dashboard only communicates with the database (as illustrated in the architecture section.

All the chart visualizations which can be built up with this add-on and integrated to the most advanced dashboards, solely serve one purpose: data digestion for tracking progress, by making now possible to follow edge-cases of a handful of documents lost in a week-long non-stopping flood, and building reports out of it.

Configure the dashboards

Fast2 does not embed any dashboard by default. However, you can get the add-on through the same portal you downloaded the Fast2 binaries. Unzip the package at the root of Fast2 installation folder.

This hierarchy will make Fast2 automatically start your dashboard.

With or without

When Fast2 is booted up, it will by default look for the dashboard folder at the root of the installation folder. If they are unzipped in the right location, they will be started after the broker triggered the database startup.

In case no dashboard folder is found, this step will be skipped after a given number of retries (which you can find the in the `./config/application.properties` file).

The dashboards can still be disabled, even if they are available in the root folder of Fast2:

v2.4- v2.6+

./config/application.properties

```
broker.kibana.embedded.enabled=true
```

Ports

By default, the dashboards are serve from the port **1791**.

However they can be accessed on a different port, which you will have to highlight in 2 different places:

1. This is required to start the add-on on another port:

v2.4- v2.6+

./config/application.properties

```
broker.kibana.embedded.port=8888
```

2. Fast2 should know where to send the user for the visualizations:

v2.4- v2.6+

./kibana-X.Y.Z/config/kibana.yml

```
server.port=8888
```

Remote access to the dashboards

If this port needs to be exposed and accessible from a different machine, the dashboard configuration must be configured to allow connections from remote users:

./opensearch-dashboards-X.Y.Z/config/opensearch_dashboards.yml

```
server.host: "0.0.0.0"
```

What if the database port is changed ?

To make sure the dashboards still reach the database, make sure the port is still up-to-date in the dashboards config file :

```
./opensearch-dashboards-X.Y.Z/config/opensearch_dashboards.yml
```

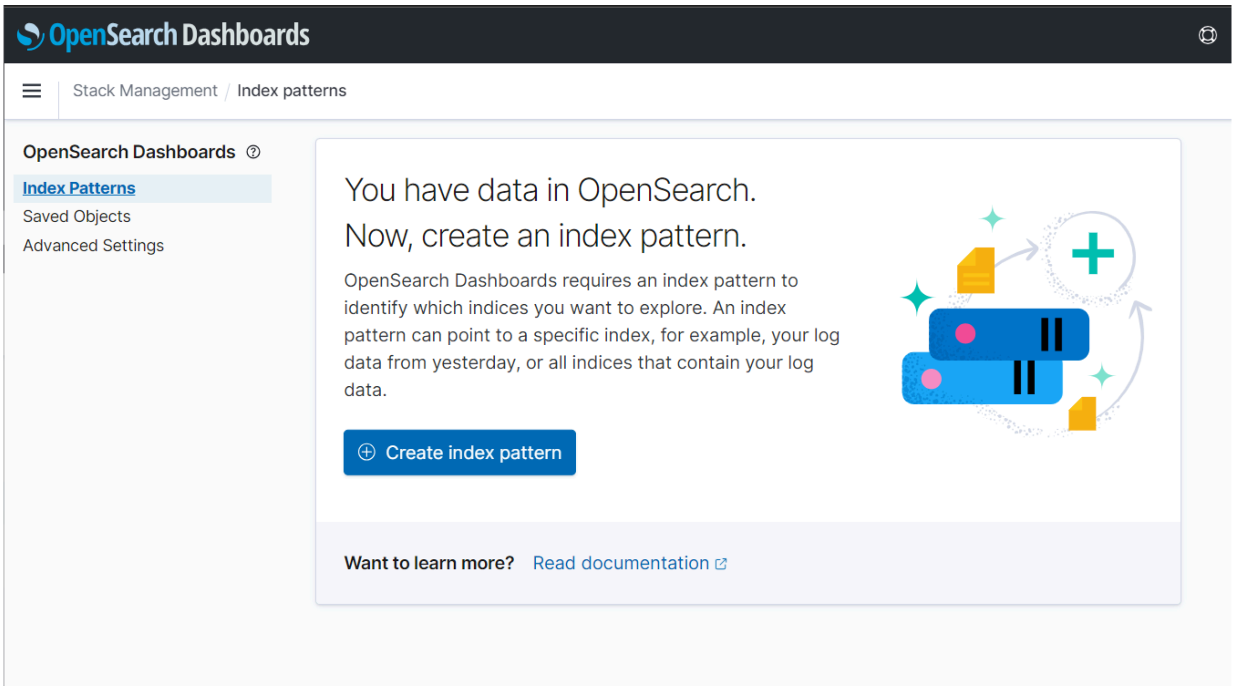
```
opensearch.hosts: ["http://localhost:1790"]
```

Advanced use

Create the index pattern

The visualizations created from the dashboards add-on needs to be attached to the `f2_*` index. If this index pattern is not existing, create it via the add-on management tools:

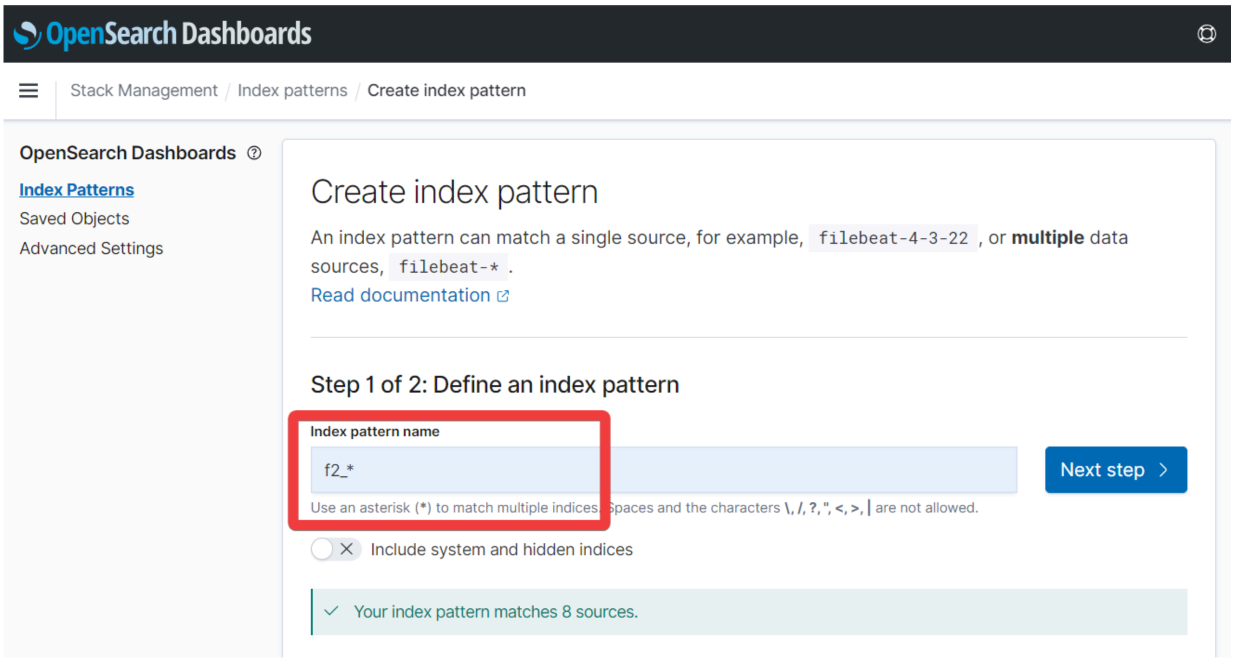
1. Stack management > Index pattern



You have data in OpenSearch.
Now, create an index pattern.

OpenSearch Dashboards requires an index pattern to identify which indices you want to explore. An index pattern can point to a specific index, for example, your log data from yesterday, or all indices that contain your log data.

2. Create a new index



OpenSearch Dashboards

Stack Management / Index patterns / Create index pattern

Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22` , or **multiple** data sources, `filebeat-*` .

Read documentation

Step 1 of 2: Define an index pattern

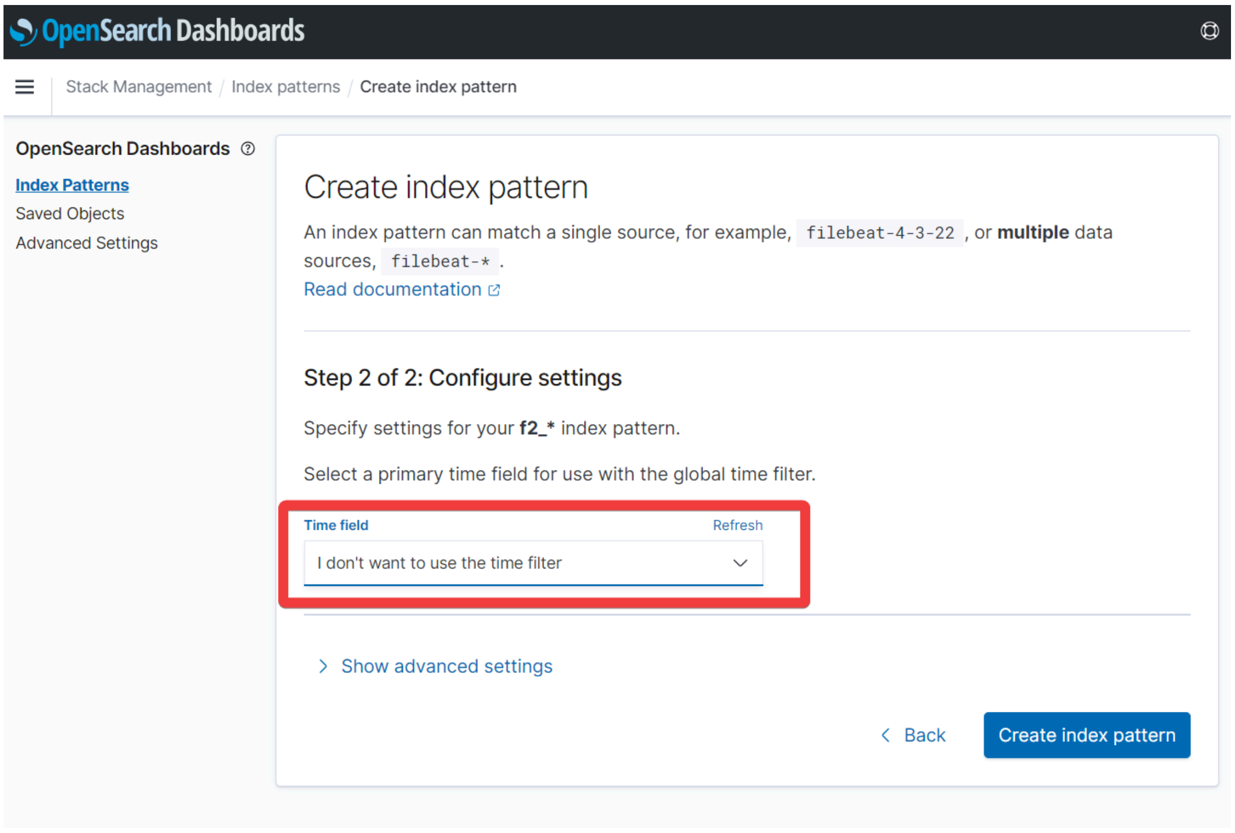
Index pattern name: f2_*

Use an asterisk (*) to match multiple indices. Spaces and the characters \, /, ?, ", <, >, | are not allowed.

Include system and hidden indices

Your index pattern matches 8 sources.

3. Set *time filter* to **none**



OpenSearch Dashboards

Stack Management / Index patterns / Create index pattern

Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22` , or **multiple** data sources, `filebeat-*` .

Read documentation

Step 2 of 2: Configure settings

Specify settings for your **f2_*** index pattern.

Select a primary time field for use with the global time filter.

Time field: I don't want to use the time filter

WARNING

Make sure to set the *time filter* to "*I don't want to use the time filter*" to prevent any issue when trying to pull the data out from the database.

NOTE

If the metadata you are looking for is not available and cannot be found in the dropdown options, refresh the `f2_*` index (which can be manually triggered from the list of saved objects).

The dashboards add-on provided with Fast2 is the go-to tool for migration report, project advancement insights, and deeper data analysis.

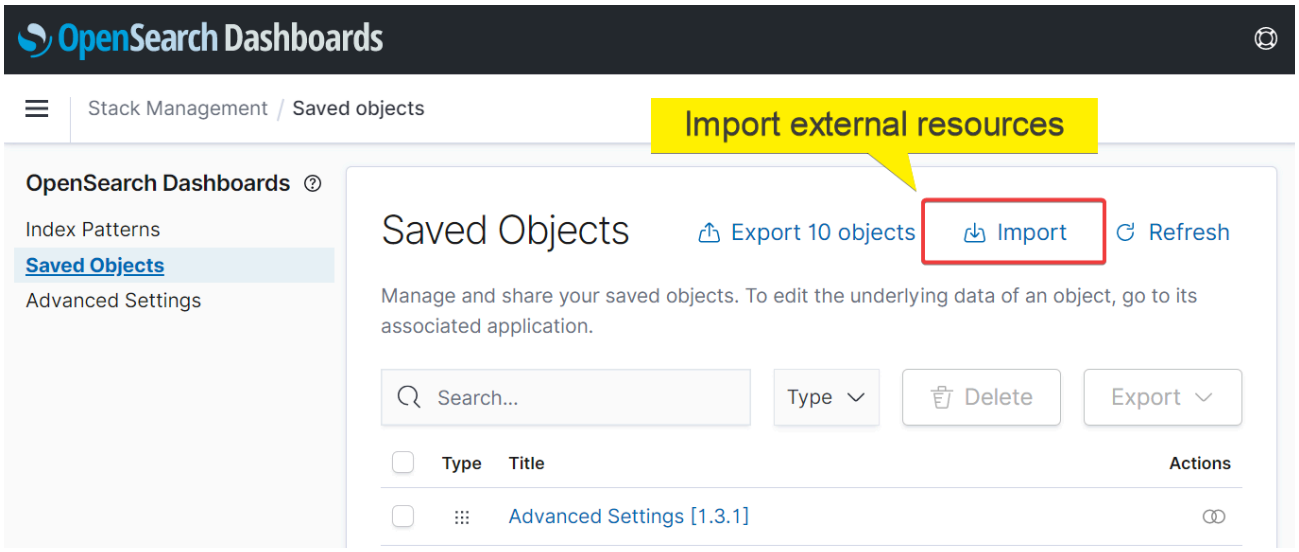
However data manipulations in this tool are not always intuitive nor straight forward, although they do open new dimensions regarding in-depth studies by the compound aggregation, data conversion and other operations now at the tips of your fingers.

Several use-cases can be envisioned, we will only relate here the data conversion steps to go through given the widespread necessity of such a basic task.

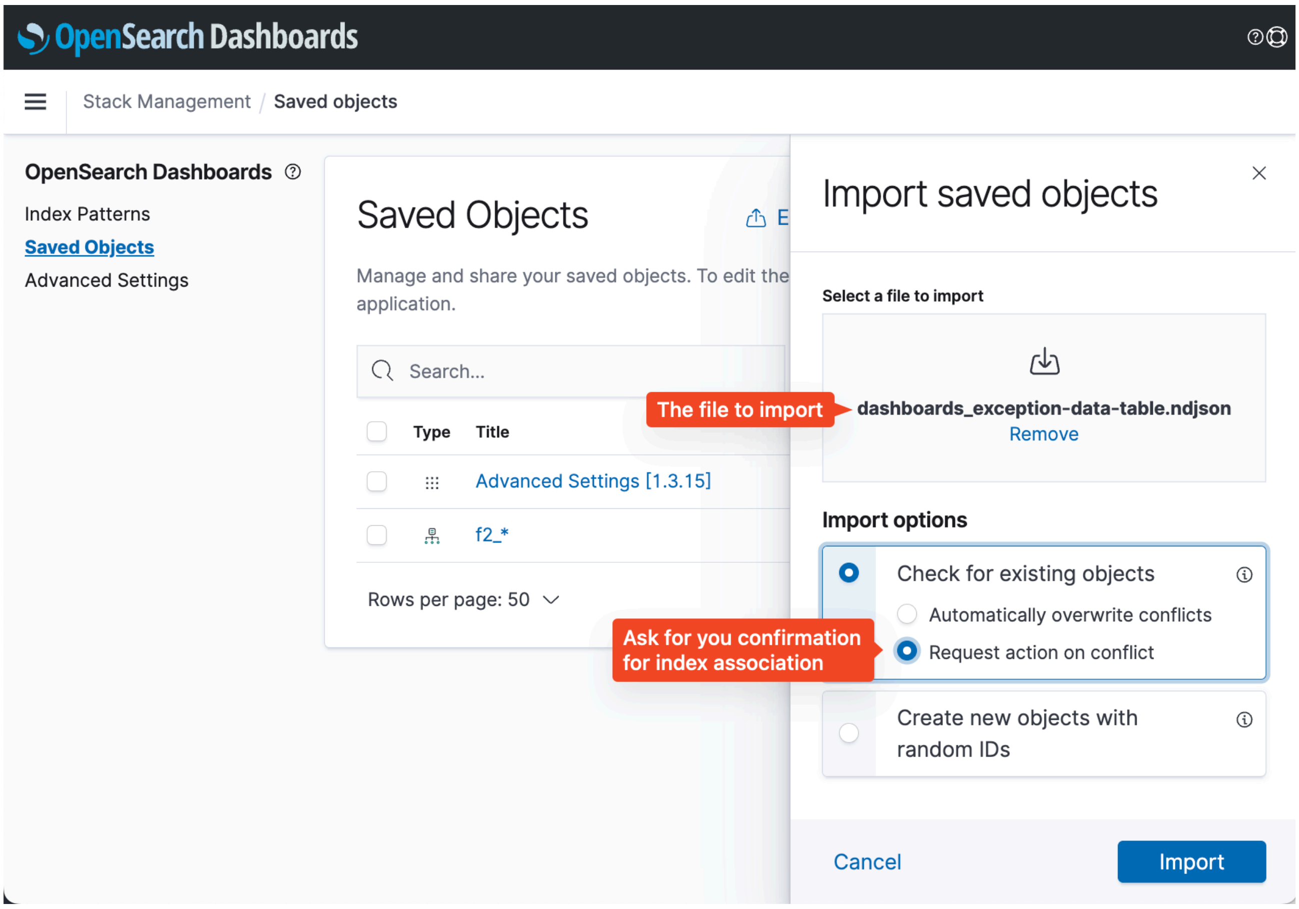
Imports objects into the dashboards feature

This section will guide you through the import process of resources (such as indices, visualization as `.ndjson` files and others).

This resource can be imported into your dashboard add-on from the right-side menu > "Stack management" > "Saved objects" > "Import" as shown on the screen-capture below :

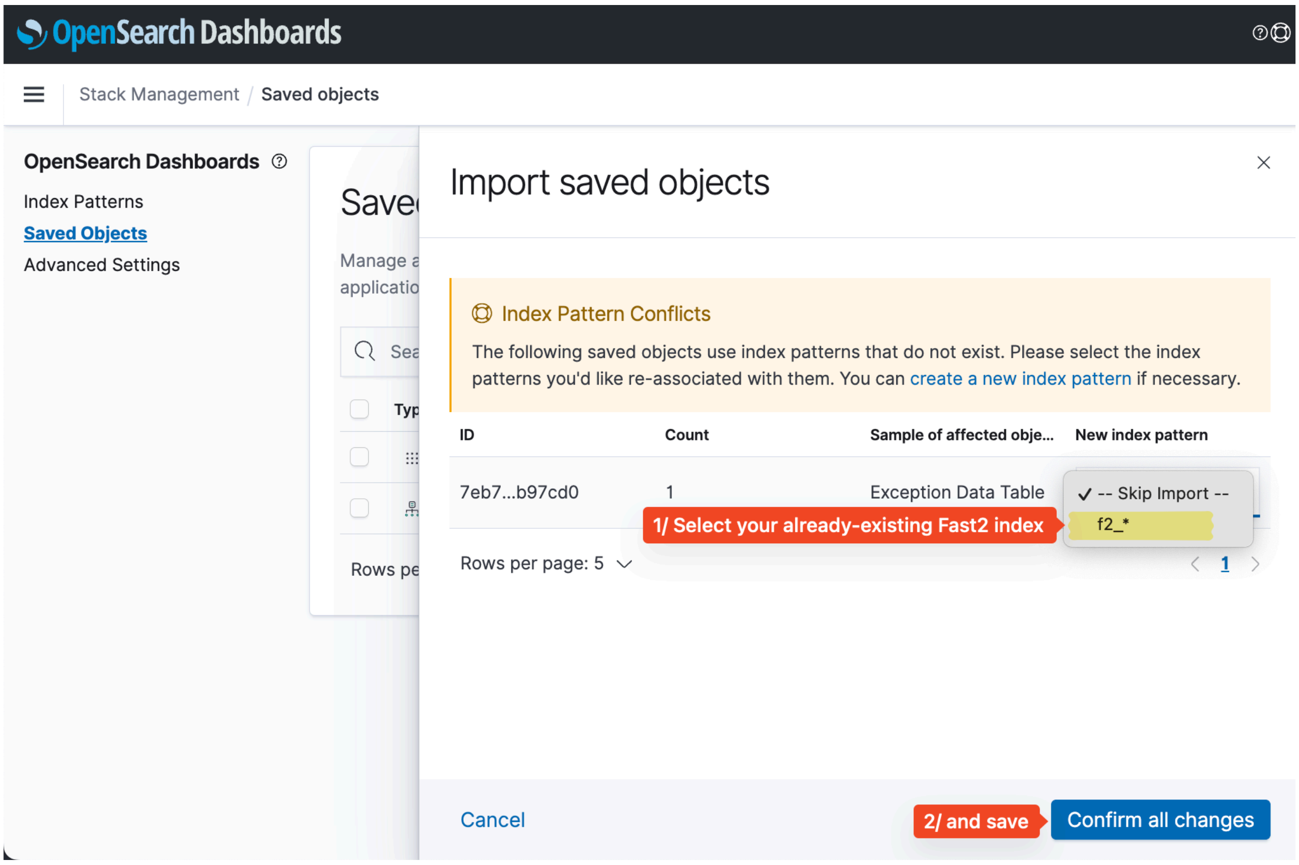


Manage and share your saved objects. To edit the underlying data of an object, go to its associated application.



Manage and share your saved objects. To edit the application.

- [✓ Check for existing objects]
  - Automatically overwrite conflicts
  - [✓ Request action on conflict]
- Create new objects with random IDs



The following saved objects use index patterns that do not exist. Please select the index patterns you'd like re-associated with them. You can create a new index pattern if necessary.

ID	Count	Sample of affected obje...	New index pattern
7eb7...b97cd0	1	Exception Data Table	✓ -- Skip Import -- / f2_*

Resource #1 : Exception table

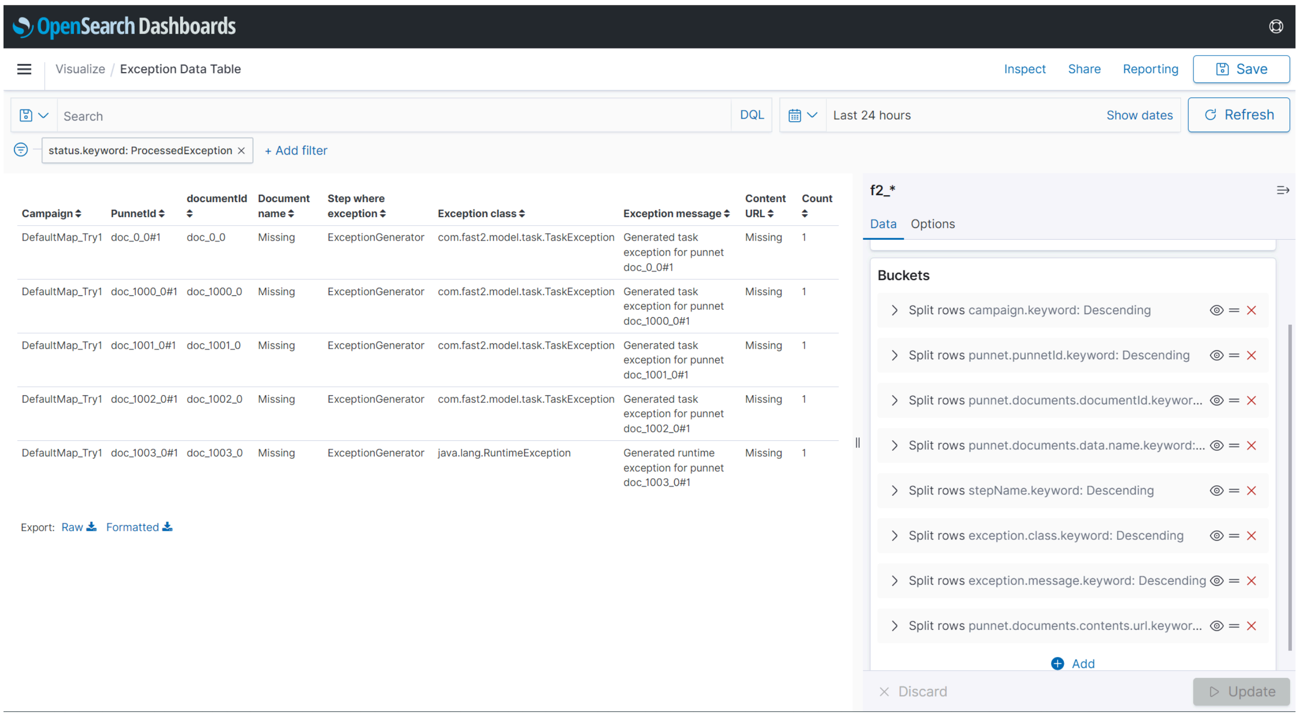
INFO

This resource has been generated from OpenSearch dashboards but can be imported into either Kibana and OpenSearch dashboards.

For a list of exceptions, since the error messages and steps in error are tracked in the database but not attached to the punnet itself, it is possible to rely on the dashboards add-on to generate a table of exceptions, accross multiple campaigns. With all the punnet details required for both investigations and error resolution, but for delta migrations afterwards, this table can be exported in CSV for externalisation of this asset.

Here is an example of a table gathering :

- **Campaign**: campaign where the exception is thrown
- **PunnetId**: ID of the punnet
- **documentId**: ID of the document (useful when the punnets store several documents)
- **Document name**
- **Step** where the exception occured
- **Exception class**
- **Exception message**
- **Content** URL (if any)
- **Count** of the number of documents in the punnet



status.keyword: ProcessedException

Campaign	PunnetId	documentId	Document name	Step where exception	Exception class	Exception message	Content URL	Count
DefaultMap_Try1	doc_0_0#1	doc_0_0	Missing	ExceptionGenerator	com.fast2.model.task.TaskException	Generated task exception for punnet doc_0_0#1	Missing	1
DefaultMap_Try1	doc_1000_0#1	doc_1000_0	Missing	ExceptionGenerator	com.fast2.model.task.TaskException	Generated task exception for punnet doc_1000_0#1	Missing	1
DefaultMap_Try1	doc_1001_0#1	doc_1001_0	Missing	ExceptionGenerator	com.fast2.model.task.TaskException	Generated task exception for punnet doc_1001_0#1	Missing	1
DefaultMap_Try1	doc_1002_0#1	doc_1002_0	Missing	ExceptionGenerator	com.fast2.model.task.TaskException	Generated task exception for punnet doc_1002_0#1	Missing	1
DefaultMap_Try1	doc_1003_0#1	doc_1003_0	Missing	ExceptionGenerator	java.lang.RuntimeException	Generated runtime exception for punnet doc_1003_0#1	Missing	1

f2_*

Buckets

- Split rows campaign.keyword: Descending
- Split rows punnet.punnetId.keyword: Descending
- Split rows punnet.documents.documentId.keywor...
- Split rows punnet.documents.data.name.keyword:...
- Split rows stepName.keyword: Descending
- Split rows exception.class.keyword: Descending
- Split rows exception.message.keyword: Descending
- Split rows punnet.documents.contents.url.keywor...

To get started with this visualisation, or to add it to your existing dashboard, click down below :

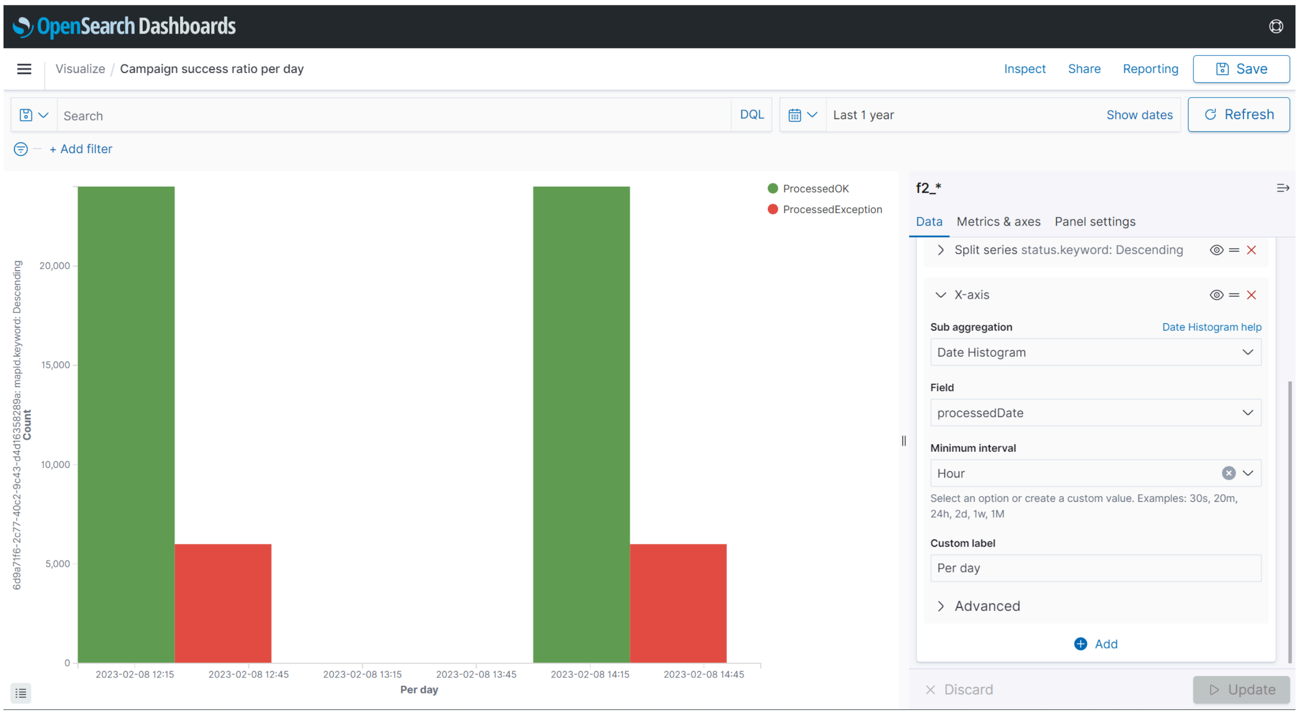
Download this resource

This resources can be imported as explained previously

Resource #2 : Campaign success ratio

> **INFO**
>
> This resource has been generated from OpenSearch dashboards but can be imported into either Kibana and OpenSearch dashboards.

For a graph visualization of the success ratio per map/campaign, the following resource can be imported for a per-day granularity of the results, where exceptions are summed up (no task differenciation), for comparison with the successfully processed documents within this same campaign.



Split series status.keyword: Descending

X-axis

Sub aggregation: Date Histogram (Date Histogram help)

Field: processedDate

Minimum interval: Hour

Select an option or create a custom value. Examples: 30s, 20m, 24h, 2d, 1w, 1M

Custom label: Per day

Advanced

To get started with this visualisation, or to add it to your existing dashboard, click down below :

Download this visualization

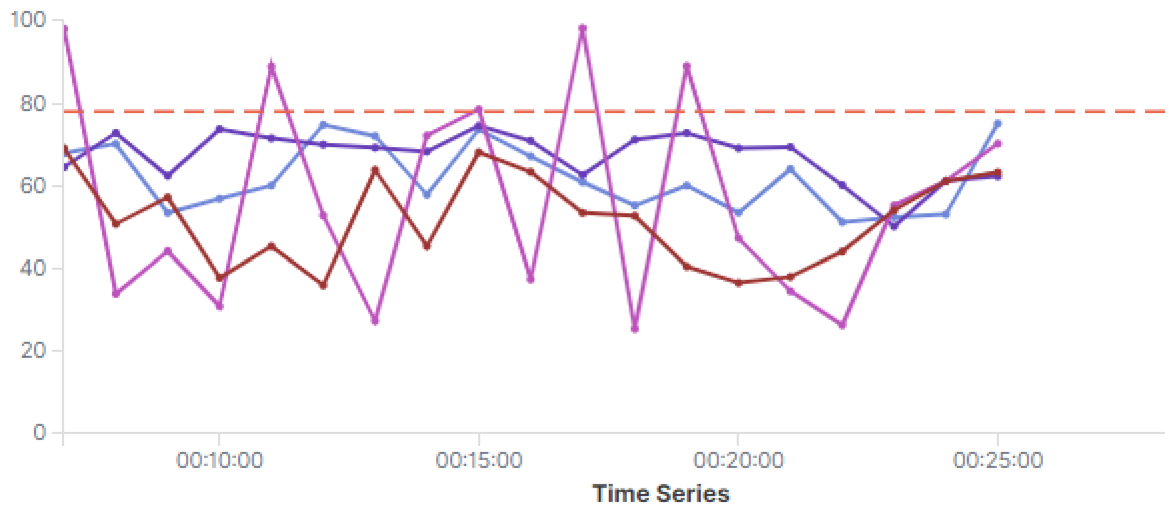
This resources can be imported as explained previously.

Resource #3 : Processing speed per task

INFO

This resource has been generated from OpenSearch dashboards but can be imported into either Kibana and OpenSearch dashboards.

For a graph visualization of the success ratio per map/campaign, the following resource can be imported for a per-day granularity of the results, where exceptions are summed up (no task differenciation), for comparison with the successfully processed documents within this same campaign.



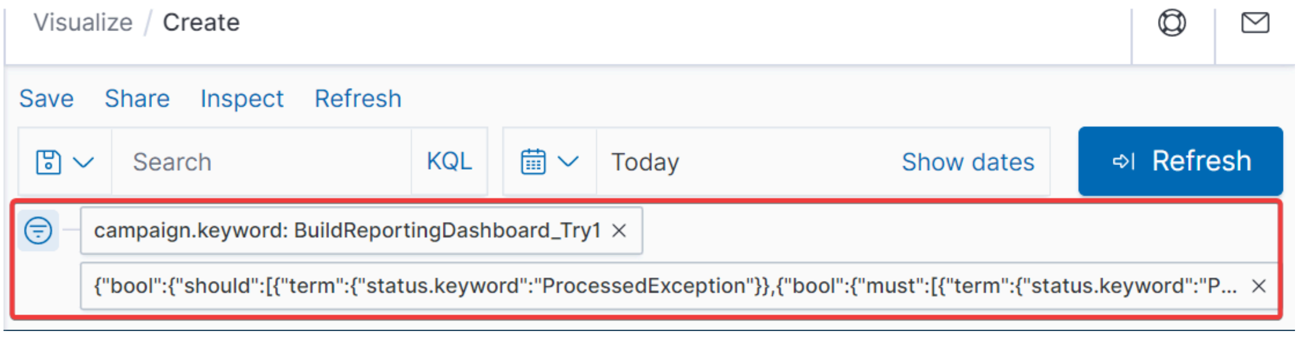
To get started with this visualisation, or to add it to your existing dashboard, click down below :

Download this visualization

This resources can be imported as explained previously.

Advanced filtering capabilities

Since the visualisations can pull out vast amounts of data from the database, most of the results might need to be narrowed down using the filter function :



campaign.keyword: BuildReportingDashboard_Try1

{"bool":{"should":[{"term":{"status.keyword":"ProcessedException"}},{"bool":{"must":[{"term":{"status.keyword":"P...

Head out to the matching documentation (Kibana or OpenSearch dashboards) for basic rules and help on how to build such filter.

We will here just focus on one main filter, which would help to only get the relevant data for either a ratio or datatable of success or failure along the migration.

Our need is to only the the documents/punnets, whose status are `ProcessedException` (to gather all failed documents, no matter the task where the exception got thrown) or the documents/punnets being both `ProcessedOK` from the injection task (which will be called here: **Last task**).

In short, we only want to select :

- the OK's of the injector, which induces the success of the migration for this document
- the KO's of all the tasks

Code-wise, since our expression would looks like this :

```
status == KO || (status == OK && step == "Last task");
```

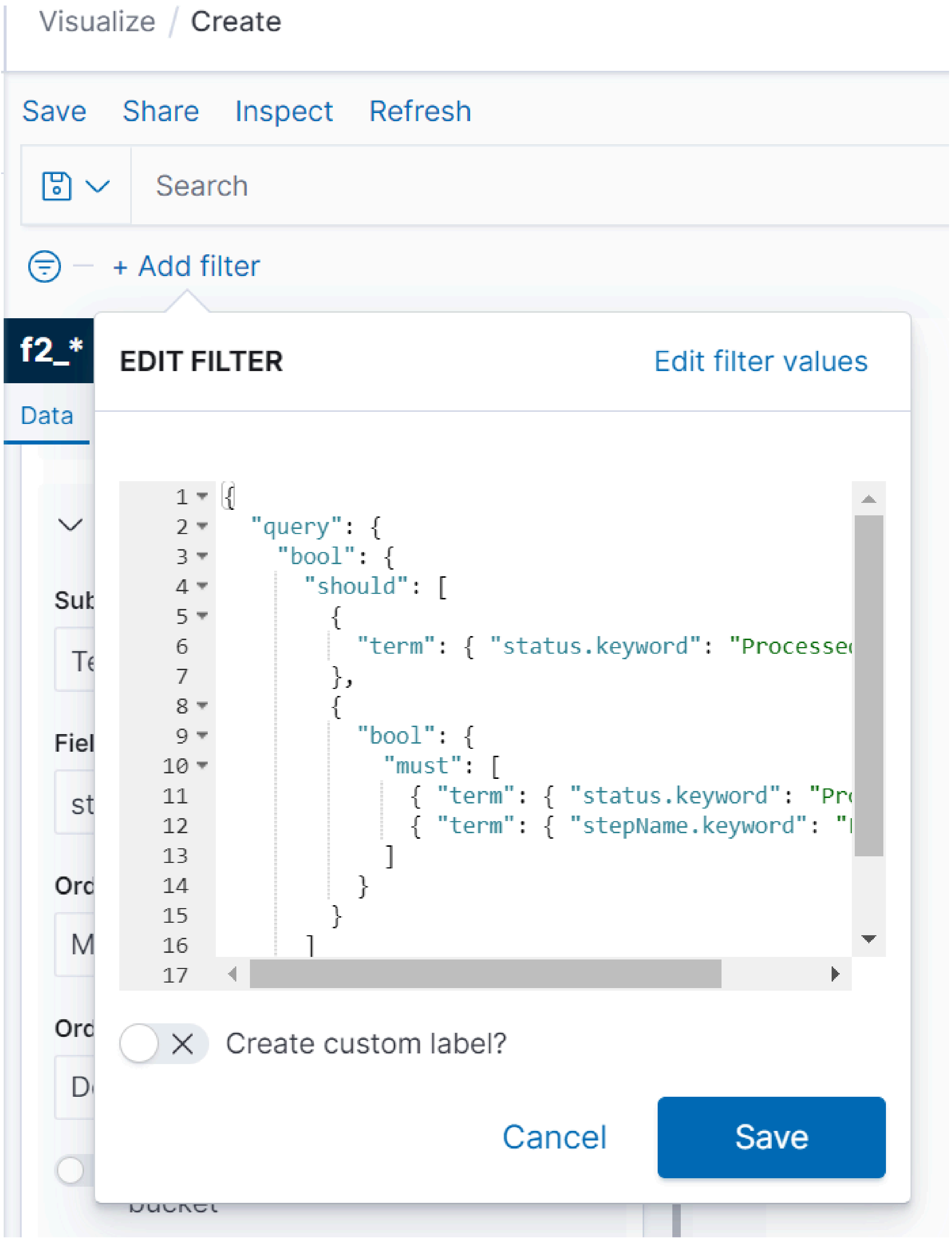
Since

- `||` is *should*
- `&&` is *must*

the final syntax is (for DSL -- *Dashboards Query Language* -- or KQL -- *Kibana query language*) :

```
{
    "query": {
        "bool": {
            "should": [
                {
                    "term": { "status.keyword": "ProcessedException" }
                },
                {
                    "bool": {
                        "must": [
                            { "term": { "status.keyword": "ProcessedOK" } },
                            { "term": { "stepName.keyword": "Last task" } }
                        ]
                    }
                }
            ]
        }
    }
}
```

This code is to be used in the filter function, as advanced filter (instead of the default fields-prepared option).



```
{
  "query": {
    "bool": {
      "should": [
        {
          "term": { "status.keyword": "Processe
        },
        {
          "bool": {
            "must": [
              { "term": { "status.keyword": "Pr
              { "term": { "stepName.keyword": "
            ]
          }
        }
      ]
```

The dashboards add-on provided with Fast2 is the go-to tool for migration report, project advancement insights, and deeper data analysis.

However data manipulations in this tool are not always intuitive nor straight forward, although they do open new dimensions regarding in-depth studies by the compound aggregation, data conversion and other operations now at the tips of your fingers.

Several use-cases can be envisioned, we will only relate here the data conversion steps to go through given the widespread necessity of such a basic task.

Datatype conversions

Let’s consider a metadata processed by Fast2 as a String instead of a float. One frequent use-case could be reporting the sum of all content size processed during a campaign. Adding up String values never ended up well so far, Kibana will have to parse these values beforehand, to have the user access the newly created value with the correct type.

We will base our example on the following punnet structure:

punnet.xml

```xml
<?xml version='1.0' encoding='UTF-8'?>
<ns:punnet xmlns:ns="http://www.arondor.com/xml/document"
punnetId="FileNetSource#page_0#pageIndex_0">
        <ns:documentset>
                <ns:document documentId="{1B62F7C4-8E75-4D99-B84C-
0AAD14B13A4E}">
                        <ns:contentset>
                                <ns:content
mimeType="application/pdf">

<ns:url>path/to/file/content</ns:url>
                                </ns:content>
                        </ns:contentset>
                        <ns:dataset>
                                <ns:data name="MimeType"
type="String">

<ns:value>application/pdf</ns:value>
                                </ns:data>
                                <ns:data name="ContentSize"
type="String">

<ns:value>43315.0</ns:value>
                                </ns:data>
                                <ns:data name="name" type="String">

<ns:value>file_name</ns:value>
                                </ns:data>
                                ...
                        </ns:dataset>
                        <ns:folderset />
                        <ns:annotationset />
                </ns:document>
        </ns:documentset>
        <folderSet />
</ns:punnet>
```

Here, the data type of the *ContentSize* property is `String`, as the type attribute states. Our job will be to parse this `String` value to `Float`, since we have a decimal.

This operation happens in 2 steps:

1. Making the original field (with the wrong type) accessible from a script,
2. Writing the correct parsing script.

Step 1: Making the field accessible

Open the dashboards retrieving data from the database where Fast2 is sending the data.

Access the Dev Tools of the dashboards to execute the query, which has to be built a specific way in order to change the mapping of the field to convert.

The index prefix of Fast2 data is `f2_`, which is why the request has to specify the index to apply the `PUT` operation on.

Next you have to write in the request body the whole way to the property to convert, first by accessing the punnet, then the properties of the punnet where the documents are stored, then the properties of the documents object where the metadata are stored, and so on.

Finally specify the current type (`text` in our case), with `fielddata:true`.

The final query will look like such:

```
PUT f2_*/_mapping
{
  "properties": {
    "punnet": {
      "properties": {
        "documents": {
          "properties": {
            "data": {
              "properties": {
                "ContentSize": {
                  "type": "text",
                  "fielddata": true
                }
              }
            }
          }
        }
      }
    }
  }
}
```

The query is successfully executed once the `acknowledged:true` message is returned.

Step 2: The parsing script

Head now to the Dashboads Management section, choose ‘Index patterns’, and select the one related to Fast2 (`f2_*`).

TIP

You may require to refresh the `f2_*` index, to make sure all the latest properties are fetched from the database.

Click on the "*Scripted field*" tab to create a new one using the data property you just made accessible.

Enter a relevant name (ex/ ContentSize-float), select the new type of this field (`number`, for our example), and write the following script:

```
if (doc.containsKey("punnet.documents.data.ContentSize"))
    return
Float.parseFloat(doc["punnet.documents.data.ContentSize"].value);
else return -1;
```

Save this new field, and create a new visualization displaying a sum of this data per campaign (as this was our initial challenge).

In the sum section, our field is now reachable for any chart requiring numerical data.

The `String` data is now accessible as a numerical chart field !! :partying_face:

And that’s it !

More use cases can be addressed by scripts with higher level of complexity, to create new data, digest or sort data, you name it.

This new dimension of data analysis via Kibana opens up way more possibilities, while increasing the precision of data aggregation to bring the best answers to the project management team.

Troubleshooting

Dashboards do not reach the database

Make sure the database port has been correctly configured in the YAML file of the dashboards. Head to the port section for more details.

Could not ping dashboard on port 1791

Make sure declaring your port in the 2 expected places. Head to the port section for more details.

Connection refused when accessing the dashboards port

In case you cannot reach the dashboards UI remotely, you might want to check several things :

- make sure the dashboards are started, from the Fast2 server (Node process running on the port declared in the `./opensearch-dashboards-X.Y.Z/config/opensearch_dashboards.yml`)
- make sure you have accessing the same port from your remote machine
- check with your IT team to make sure the network securities are allowing outbound rules for this port (as they might have done for the Fast2UI port)
- make sure you updated the dashboards `server.host` configuration (check this section for more details)

FAQ

Run Fast2 without dashboards

It is possible to run Fast2 without the dashboards, as this add-on is just reading data stored in the internal database, to serve them as graphical vizualisation.

By no mean this add-on is **necessary** for the migration.

Access dashboard when not migrating

Visualizations can be reached via 2 different ways, depending on your needs:

- Either from the UI of Fast2,
- Or directly from the browser via the declared port (1791 or other).

The latter option gives access to the tool even when Fast2 is not running.

Since the dashboards fetch data directly from the database, it is not possible to populate the visualizations with migration data if the database is not running as well.

Dashboards do not reach the database

Make sure the database port has been correctly configured in the YAML file of the dashboards. Head to the port section for more details.

Could not ping dashboard on port 1791

Make sure declaring your port in the 2 expected places. Head to the port section for more details.

Catalog

All along this documentation concerning the configuration of Fast2 objects (either tasks or tools used within tasks), consider the default value set to `false` if no default value is mentionned for boolean fields.

Here are the 8 categories we cover:

1. **Sources**

   Scan the source environment to identify documents for migration based on criteria.

2. **Content Sources**

   Extract content, metadata, folder references, and all versions from the source environment.

3. **Credentials**

   Establish connections with remote systems to enable communication with our migration tool.

4. **Conversion Tasks**

   Convert contents, including documents and annotations, as part of the migration process.

5. **Transformation Tasks**

   Focus on metadata-oriented operations, such as transforming, enriching, validating, or changing metadata formats.

6. **Helpers**

   Perform handy tasks for side operations to support and enhance your migration process.

7. **Tools**

   Perform intermediate tasks for basic operations, independent of the source or destination environments.

8. **Injectors**

   Upload or inject migration data and documents into the destination system.

In each category, you'll find detailed information about the specific configuration fields for each workflow task, providing comprehensive guidance for your migration needs.

Catalog / Sources

AWSSource - Complete extractor module from AWS S3

This AWS extractor performs from a list of sources the extraction of your document content. Many options (suffix, prefix...) exist to optimally specify the documents you want to take into account

Mandatory settings

Key	Type	Description
AWS connection provider	AWSConnectionProvider	Must have AmazonS3FullAccess permission
Source buckets	`String list`	Buckets where folders are stored

Optional settings

Key	Type	Description	Default value
Accept quotes in values	`Boolean`	If enabled, this option will accept quotes in values	
AWS start-after key	`String`	Absolute path of S3 object to start after	

Key	Type	Description	Default value
ARN key for KMS encryption	`String`		
New column names to set	`String list`	If empty, populated from first line	
Replace empty titles	`Boolean`	If enabled, any empty title in the CSV file will be replaced by the default value. If several titles miss, the default title will be suffixed with an incremental index.	
AWS suffix	`String`	S3-object will be extracted if its key has such suffix	
Number of lines to skip	`Integer`	This option helps to skip lines, meaning their data will not be processed. By default, only the 1st line is skipped considering it surely consists in the headers row Ex/ In a file of 10 lines, putting '3' in the input field will skip the 1st, 2nd and 3rd lines	`1`

Key	Type	Description	Default value
Default column title	`String`	Default value used for untitled columns. Will be incremented with a number if many. Will only be used if the replace empty titles option is enabled.	`Untitled`
Continue processing CSV on fail	`Boolean`	If enabled, the following errors will not trigger an exception: - CSV file does not exist - CSV file is empty (no line) - CSV file has only headers and no line for documents. Note that if you give 5 CSV paths and the number 3rd is in error, only the Fast2 logs will provide information regarding the failing CSV file.	
Source folders	`String list`	Folders in the S3 bucket(s) containing the files to migrate	
AWS prefix	`String`	S3-object will be extracted if its key has such prefix	
Stop at first error in CSV	`Boolean`	Fast2 will automatically be stopped at the first error encountered in the CSV	`false`

Key	Type	Description	Default value
Column headers in first CSV file only	`Boolean`	Only read column definitions from the first parsed CSV file	`false`
Documents per punnet from CSV	`Integer`	Number of documents each punnet will carry when processing a CSV file Ex/ By setting this value to 2, each punnet created will contain 2 documents	`1`
CSV separator	`String`	Separator between each value. This option will be ignored if 'Process files as list of punnets' is disabled.	`,`

Key	Type	Description	Default value
Process files as list of punnets	`Boolean`	The expected format is a CSV file (1 row for headers, next rows for 1 punnet each), but the `.csv` extension is not mandatory. Only single-documents punnets will be created (ex/ not working for multiversions documents). Multivalue data will be concatenated to one whole String value. The first line of the file will be considered as CSV header line.	
extraColumns	`String list`		

AlfrescoRestSource - Alfresco extractor using Alfresco REST protocol

This task relies on the Alfresco public REST API (with v1.0.4 of the Alfresco REST client) to retrieve documents and metadata into a given Alfresco instance

Mandatory settings

Key	Type	Description
CMIS query or AFTS query	`String`	Query used to retrieve the objects from Alfresco Ex/ SELECT * FROM cmis:document WHERE cmis:name LIKE 'test%' or cm:title:'test%'
Alfresco connection provider	AlfrescoRESTConnectionProvider	

Optional settings

Key	Type	Description
Max item to return per call	`Integer`	Set the paging max items threshold to specify the numbe of Alfresco objects to retrieve per call.

Key	Type	Description
Fields to extract	`String`	The less the better ! Only the 'id' is necessary to start the migration workflow. Separate the different values with a comma, no space. Use properties from com.alfresco.client.api.common.constant.PublicAPIConsta library. Ex/ id,name

AlfrescoSource - Alfresco extractor using CMIS technology

Through an SQL query, this alfresco extractor will use the CMIS technology to fetch the content, the metadata and the annotations of your documents from a given Alfresco repository

Mandatory settings

Key	Type	Description
SQL query to extract documents	`String`	Fast2 will retrieve all documents, folder, references, items and metadata matching this query. If the query is exhaustively specifying data to extract, uncheck the 'Extract document

Key	Type	Description
		properties'. The data `cmis:objectId` will be mandatory. Ex/ SELECT * FROM cmis:document
Alfresco connection provider	AlfrescoCMISConnectionProvider	CMIS version must be 1.1

Optional settings

Key	Type	Description	Default value
Property Helper	PropertyHelper		
Number of items per result page	`Integer`	Maximum number of results provided	`1`
Number of documents per punnet	`Integer`		`1`
Extract document properties	`Boolean`		`true`

Key	Type	Description	Default value
Keep folder structure within document	`Boolean`	requires extractProperties to be true	`true`
Extract document content	`Boolean`	Does not work asynchronously	`false`

BlankSource - Empty punnet generator

This source builds a punnet list containing one or more empty documents. Each document will only contain its identifier : documentId. This punnet can then be enriched by other steps in the processing chain.

Mandatory settings

Key	Type	Description
Document IDs	DocumentIdList	Source list of documents to extract from their IDs

Optional settings

Key	Type	Description	Default value
Document per punnet	`Integer`	Number of documents each punnet punnet must carry on Ex/ The input file includes 10 lines meaning 10 document identifiers to extract. By setting this value to 2, Fast2 will create 5 punnets, each containing 2 documents	`1`

CMODSource - Complete extraction module from a CMOD environment

This task is used to extract documents in the Content-Manager On Demand ECM. One CMOD document is equivalent of 1 punnet of 1 document. Indexes, optional content and annotations will also be extracted. A WAL request is made to find the corresponding documentId in ImageServices. The metadata extraction is then carried out. Relative data are stored in each document of the punnet being processed.Note: All Image Services properties are exported systematically. This task is not a real source task. The documents to be extracted are identified by an BlankSource task generating a set of empty Punnets, i.e. containing only documents each bearing a document number (documentId) to extract.This task relies on the 'libCMOD.dll' library. This library must be in a directory of the Windows PATH. In the wrapper.conf or hmi-wrapper.conf file, activate the use of this library: wrapper.java.library.path.<increment> = ../libCMOD/dll32For the moment, only 32-bit libraries are configured

Mandatory settings

Key	Type	Description
CMOD connection provider	CMODConnectionProvider	
Folders to extract	`String list`	List of CMOD folders which will be scanned. Additional level(s) of filter can be used with the SQL query down below.

Optional settings

Key	Type	Description	Default value
SQL query to extract documents	`String`	Enter here the `WHERE` clause used to filter documents. Since this request is made on the indexes of CMOD documents, the property used to filter out the documents need to be indexed in CMOD prior to any extraction. Ex/ WHERE Date = '2012-11-14'	
Extract document annotations	`Boolean`	The document annotation will be extracted during the process	`false`

Key	Type	Description	Default value
Number of documents per punnet	`Integer`		`1`
Extract document content	`Boolean`	The document content will be extracted during the process	`false`
Maximum results count	`Integer`		`2000`

CMSource - Complete extractor from Content Manager solution

Mandatory settings

Key	Type	Description
CM connection provider	CMConnectionProvider	
SQL query	`String`	Select precisely documents you want to extract through a classic SQL query

Optional settings

Key	Type	Description	Default value
Extract standard system properties	`Boolean`		`false`
Extract advanced system properties from DKDDO object	`Boolean`		`false`
Maximum results returned by the query	`Integer`	Set to 0 to disable limiting number of results	`0`
Number of documents per Punnet	`Integer`	Set the number of documents each punnet will hold	`1)`
Extract custom properties	`Boolean`		`false`

Key	Type	Description	Default value
Query type	`Integer`	See com.ibm.mm.beans.CMBBaseConstant for further details. Default value is XPath (7)	`7`

CSVSource - CSV file parser

This task can be used to start a migration from a CSV file. By default, the first line of your file is considered as the column headers. Whether the column values are surrounded with double-quotes (`\_`) or not, the CSVSource task will process either way. If you need to force the document ID for the whole process, use the metadata `documentId`.

Mandatory settings

Key	Type	Description

CSV paths	`String list`	List of paths to CSV files to be parsed. Check out the following examples for allowed formats
		Ex/
		`C:/samples/myDocument.csv`
		`C:\\samples\\myDocument.csv`
		`C:\\\\samples\\\\myDocument.csv`
		`\"C:\\samples\\myDocument.csv\"`
		`C:/samples/${map}.csv`

Optional settings

Key	Type	Description	Default valu
Accept quotes in values	`Boolean`	If enabled, this option will accept quotes in values	
CSV file path metadata	`String`	Punnet property name containing the CSV file path. Set to empty or null to disable	
File name for error CSV file	`String`	This option might be useful when you need to have a specific file name where to register the lines in error of your CSV file. The name can both be linked to some workflow properties surrounded with `${...}` (ex/ campaign, punnetId, etc) or hard-written. Warning: This value can be overwritten by the *Associate CSV-error file with original CSV filename* option	`lines_in_error`
New column names to set	`String list`	If empty, populated from first line	
Replace empty titles	`Boolean`	If enabled, any empty title in the CSV file will be	

Key	Type	Description	Default valu
		replaced by the default value. If several titles miss, the default title will be suffixed with an incremental index.	
Folder path for error CSV file	`String`	The error file will be stored in your system. You can choose where by configuring this very field. Here as well you can set the path either with workflow properties (`${...}`) or hard-write it	`./csv_errors/`
Number of lines to skip	`Integer`	This option helps to skip lines, meaning their data will not be processed. By default, only the 1st line is skipped considering it surely consists in the headers row Ex/ In a file of 10 lines, putting '3' in the input field will skip the 1st, 2nd and 3rd lines	`1`
Default column title	`String`	Default value used for untitled columns. Will be incremented with a number if many. Will only be used if	`Untitled`

Key	Type	Description	Default valu
		the replace empty titles option is enabled.	
Generate hash of CSV content	`Boolean`	The hash of the content will be generated and stored in the punnet among a property named hashData	`false`
Continue processing CSV on fail	`Boolean`	If enabled, the following errors will not trigger an exception: - CSV file does not exist - CSV file is empty (no line) - CSV file has only headers and no line for documents. Note that if you give 5 CSV paths and the number 3rd is in error, only the Fast2 logs will provide information regarding the failing CSV file.	
File encoding	`String`	CSV encoding character set	`UTF-8`

Key	Type	Description	Default valu
Associate CSV-errors file with original CSV filename	`Boolean`	This checkbox allows you to match your error file with your original CSV file, just suffixing the original name with '_KO'. That way, if you use multiple files, all the lines in error will be grouped by file name. Using this option overwrite the *File name for error CSV file*, but still can be used in addition of the *Folder path for error CSV file*	`false`
Stop at first error in CSV	`Boolean`	Fast2 will automatically be stopped at the first error encountered in the CSV	`false`
File scanner (Deprecated)	FileScanner	*THIS OPTIONS IS DEPRECATED*, consider using the 'CSV paths' instead.	
Column of document ID	`String`	Column header of the metadata to set as the document ID	`documentId`
Document property	`String`	Set to empty or null to disable	

Key	Type	Description	Default valu
name containing CSV file path			
Move to path when finished	`String`	Consider using `${variable}` syntax	
Column headers in first CSV file only	`Boolean`	Only read column definitions from the first parsed CSV file	`false`
Documents per punnet from CSV	`Integer`	Number of documents each punnet will carry when processing a CSV file Ex/ By setting this value to 2, each punnet created will contain 2 documents	`1`
CSV separator	`String`	Separator between each value. This option will be ignored if 'Process files as list of punnets' is disabled.	`,`
Extra columns	`String list`	List of the form target=function:arg1:arg2:...	

DctmSource - Complete extractor from Documentum

This connector will extract basic information from the source Documentum repository. Since Documentum architecture involves particular port and access management, a worker should be started on the same server where Documentum is running.

Make sure to check the basic requirements at the setup for Documentum on the official Fast2 documentation.

Mandatory settings

Key	Type	Description
Connection information to Documentum Repository	DctmConnectionProvider	
The DQL Query to run to fetch documents	`String`	The less attributes you fetch, the faster the query will be executed on the Documentum side. Ex/ `SELECT r_object_id FROM dm_document WHERE ...`
Connection information to	DctmSshProvider	

Key	Type	Description
Documentum server machine		

Optional settings

Key	Type	Description	Default value
Batch size	`Integer`	If size is `<1`, the size will be defined from the Documentum server-side.	`50`
SSH client	DctmSshClient	SSH client used to establish the connection with the Documentum server	

EmbeddedDbSourceRest - Perform requests on Fast2 database without any size restriction

This task is used to retrieve punnets from a previously executed campaign.

Mandatory settings

Key	Type	Description	Default value
Embedded db port	`Integer`		`1790`
Embedded db hostname	`String`		`localhost`
Embedded db scheme	`String`		`http`
Campaign name	`String`	The campaign name that you would like to have the data. Ex/ `myMap_Run1`	
Step Id	`String`	Will return the punnets of this task (UUID of the step)	

FileNet35Source - Complete extractor from FileNet 3.5

The FileNet35Source retrieves existing documents from the FileNet P8 3.5 ECM through a query. This punnet will contain the metadata of the recovered document, its content and annotations

Mandatory settings

Key	Type	Description
FileNet 3.5 connection provider	FileNet35ConnectionProvider	Connection parameters to the FileNet instance
SQL query	`String`	SQL query corresponding to the list of documents to extract

Optional settings

Key	Type	Description	Default value
Attribute used for Document IDs	`String`	Name of the FileNet P8 3.5 attribute corresponding to the values retrieved in the Document IDs list	`Id`
Empty punnet when no result	`Boolean`	An empty punnet will be created even if the result of the query is null	`false`
Documents per punnet	`Integer`	Number of documents each punnet punnet must carry on Ex/ By setting this value to 2, each punnet created will contained 2 documents	`1`

Key	Type	Description	Default value
Document IDs	DocumentIdList	Source list of documents to extract from their IDs	

FileNetSource - Complete extractor from FileNet P8

The FileNetSource source retrieves existing documents from the FileNet P8 5.x ECM through an SQL query. This punnet will contain the metadata of the recovered document, security information and parent folders.

Mandatory settings

Key	Type	Description
Object store name	`String list`	Name of the repository to extract from
SQL query	`String`	SQL query corresponding to the list of documents to extract
FileNet connection provider	FileNetConnectionProvider	Connection parameters to the FileNet instance

Optional settings

Key	Type	Description	Default value
Number of entries per result page	`Integer`	Number of results returned per page by the FileNet P8 query	`1000`
Documents per punnet	`Integer`	Number of documents each punnet punnet must carry on Ex/ By setting this value to 2, each punnet created will contained 2 documents	`1`
Extract object type properties	`Boolean`	The FileNet P8 metadata of the document which are Object type will be saved at the punnet level	`false`
Extract FileNet system properties	`Boolean`	System metadata during extraction is saved at the punnet level	`false`
Properties to extract	`String list`	Exhaustive list of FileNet metadata to extract. If empty, all properties will be extracted.	
Extract FileNet security	`Boolean`	The security of the document will be saved at the punnet level	`false`

Key	Type	Description	Default value
Extract documents instance informations	`Boolean`	The fetchInstance method makes a round trip to the server to retrieve the property values of the ObjectStore object	`false`
Extract folders absolute path	`Boolean`	The absolute path of the folder inside the FileNet instance will be extracted during the process	`false`
Throw error if no result	`Boolean`	Throw exception when SQL Query finds no result.	

FlowerSource - Flower extractor

Allows components extraction from Flower using JSON formatted Flower request. Components can be documents, folders, virtual folders or tasks.

Mandatory settings

Key	Type	Description
FlowerDocs connection provider	FlowerDocsConnectionProvider	

Key	Type	Description
Flower component category	`String`	Choose among DOCUMENT, TASK, FOLDER or VIRTUAL_FOLDER
JSON Flower Search Request	`String`	Patterns can be used too

LocalSource - A generic broker for wildcarded punnet lists

This class will search for local files to analyze them from a defined path

Mandatory settings

Key	Type	Description

Files paths	`String list`	List of paths to files to be parsed. Patterns `${...}` are not supported. The threshold can be maxed-out, exclusions are not supported.
		Ex/
		`C:/samples/myDocument.txt` -> retrieve only one document
		`C:\\samples\\myDocument.txt`
		`C:\\\\samples\\\\myDocument.txt`
		`\"C:\\samples\\myDocument.txt\"`
		`C:/samples/*.*` -> retrieve all files directly at the root

Key	Type	Description
		of the `samples/` folder, no matter their extension `C:/samples/**` -> retrieve all files directly at the root of the `samples/` folder, as well as file inside subfolders `C:/samples/**/*.yes` -> retrieve all files directly at the root of the `samples/` folder, as well as file inside subfolders, whose extension is `.yes`.

Optional settings

Key	Type	Description	Default value
File scanner (Deprecated)	FileScanner	*THIS OPTIONS IS DEPRECATED*, consider using the 'Files paths' instead.	
Fallback XML/Json parsing	`Boolean`	If true, the file will be added as document content in the punnet when XML parsing fails. Consider adding this file as a regular file (not an XML)	`false`
Skip parse exceptions	`Boolean`	The task does not throw an error when XML parsing fails. Do not stop parsing and resume to next candidate	`false`
XSL Stylesheet path	`String`	The XSL stylesheet file to use when parsing XML files	

Key	Type	Description	Default value
Number of files per punnet	`Integer`	If the files are not in XML format, the punnet will contain as many documents as defined in this option	`1`
Allow any kind of file	`Boolean`	All types of files can be added. Otherwise, only XML-based Punnet descriptions are allowed	`true`
Skip XML parsing	`Boolean`	The XML file will not be parsed before being added to the punnet. Not recommended in most cases	`false`
Maximum number of files scanned	`Integer`	If this field is completed, the number of files scanned will not exceed the value filled in. Leave empty to retrieve all files matching input pattern filter	

MailSource - Complete extractor from mail box

The MailSource task extracts messages from an e-mail box. Each extracted message will correspond to a punnet, one document per punnet

Mandatory settings

Key	Type	Description
MailBox connection provider	MailBoxProvider	

Optional settings

Key	Type	Description	Default value
Search in Headers	`String`	Enter a pair of header and pattern to search separated by a colon `:`. Ex/ cc:copy	
Header names	`String list`	List of header names (case-sensitive) to retrieve from the mail. Message-Id, Subject, From, To, Cc and Date are added by default	
Start Id	`Integer`	Index from which the first message should be extracted	`1`
Update document with mail root folder name	`String`	Name of the metadata to add to the document. If filled, the full name of the source folder is indexed in this metadata. Set to null or empty to disable updating	
Folders to scan	`String list`	List of files to scan in the mailbox. If filled, override root folder name	

Key	Type	Description	Default value
		from MailBox connection provider configuration	
AND condition for search	`Boolean`	Checking this options will only retrieve messages matching all search conditions possible (unread messages, text in header, body or subject). If unchecked, the 'OR' operand will be applied.	
Forbidden characters	`String`	List of characters to remove from Message-Id when building the DocumentId	`<>:"/\
Search in Subject	`String`		
Search in Body	`String`		
Only unread messages	`Boolean`		

OpenTextSource - OpenText extractor using OpenText REST protocol

Mandatory settings

Key	Type	Description
OpenText credentials	OpenTextCredentials	
OpenText client	OpenTextRestClient	
Node Id	`Integer`	

Optional settings

Key	Type	Description	Default value
Order by named column	`String`	Format can be 'name' or 'asc_name' or 'desc_name'. If the prefix of asc or desc is not used then asc will be assumed Ex/ asc_name	
Ticket period	`Integer`	Time in seconds between two ticket creation	`60`

RandomSource - Random punnet generator

Randomly produces punnets containing documents, metadata, content...

Mandatory settings

Key	Type	Description	Default value
Number of punnet to generate	`Integer`	If 'minimum punnet number' is set, this value here will be considered as the higher threshold	`1000`

Optional settings

Key	Type	Description	Default value
Maximum document number	`Integer`	Excluded	`1`
Minimum metadata number	`Integer`	Included	`1`
Minimum punnet number	`Integer`	If not set, the number of generated punnets will be exactly the number set at 'Number of punnets to generate'	
Maximum number of metadata values	`Integer`	Included	`6000`

Key	Type	Description	Default value
Minimum number of metadata values	`Integer`	Included	`0`
Maximum metadata number	`Integer`	Excluded	`10`
Minimum document number	`Integer`	Included	`1`

SQLSource - Complete extractor from SQL database

Extract and map to punnet or document layout specified properties

Mandatory settings

Key	Type	Description
SQL connection provider	SQLQueryGenericCaller	

Key	Type	Description
SQL query	`String`	Select precisely documents you want to extract through a classic SQL query

Optional settings

Key	Type	Description	Default value
Property name to group by document	`String`	Column used to group lines by document. If used set an 'ORDER BY' in your sql query	
SQL mapping for punnet	`String/String map`	Mapping of SQL properties to punnet metadata. Use 'punnetId' for Punnet Id	
Allow duplicates data	`Boolean`		
Property name to group by punnet	`String`	Column used to group lines by punnet. If used set an 'ORDER BY' in your sql query	
SQL mapping for document	`String/String map`	Mapping of SQL properties to document metadata. Use 'documentId' for Document Id, otherwise	

Key	Type	Description	Default value
		the first column will be used as documentId	
Push remaining, non-mapped columns as document properties	`Boolean`		`true`

ZipSource -

Catalog / Content sources

AWSContentSource - Extract content from AWS S3 bucket

Mandatory settings

Key	Type	Description
AWS access credentials	AWSConnectionProvider	Credentials of the user (must have been granted AmazonS3FullAccess permission).

Optional settings

Key	Type	Description	Default value
ARN key for getAwsPrefixKMS encryption	`String`		
Bucket name	`String`	Name of the S3 bucket where the content is stored.	`${bucket}`

Key	Type	Description	Default value
Content path (S3 object key)	`String`	Path leading to S3 object corresponding to the content you intend to extract from the bucket. To use this options, you must enable the content extraction option. Ex/ `${contentPath}`	
Extract contents	`Boolean`	All existing contents of documents will be replaced by the newly found contents, retrieved from the S3 bucket. If the S3 objects are parsed as punnets, then the contents will be attached based on the 'Content path' input field.	
Process s3 objects as punnets	`Boolean`		

AlfrescoContentExtractor - Alfresco content extractor using CMIS technology

This alfresco extractor will use the CMIS technology to fetch your document content from a given Alfresco repository

Mandatory settings

Key	Type	Description
Alfresco connection provider	AlfrescoCMISConnectionProvider	CMIS version must be 1.1

Optional settings

Key	Type	Description	Default value
Property Helper	PropertyHelper		
Extract document content	`Boolean`		`true`

AlfrescoRestContentExtractor - Alfresco content extractor using Alfresco REST protocol

This task relies on the Alfresco public REST API (with v1.0.4 of the Alfresco REST client) to retrieve documents and metadata into a given Alfresco instance

Mandatory settings

Key	Type	Description
Alfresco connection provider	AlfrescoRESTConnectionProvider	

Optional settings

Key	Type	Description	Default value
Date format	`String`		`E MMM dd HH:mm:ss Z YYYY`
CMIS query	`String`	CMIS SQL query, pattern resolvable, to fetch document based on alternative data. Using this feature will create new docs in the punnet with corresponding ID of documents. Consider following this task with a secondary AlfrescoRestContentExtractor task to extract data and contents.	
Extract content	`Boolean`		
Extract all versions	`Boolean`	Extract the superseded versions of the documents matching the query	

Key	Type	Description	Default value
Extract parent site	`Boolean`	If the document is not stored in an Alfresco site, nothing will happen. Otherwise, the site details will be attached to the punnet dataset.	
Map permissions	`Boolean`	Map permissions to either the document, folder or site.	
Map parent folder	`Boolean`	Map direct parent folder info onto the related document.	
Extract folders as tree	`Boolean`	Extract folders as tree, with all parent folders. This option must be selected if you wish to map permissions of parent folders.	
Extract users as email addresses	`Boolean`		

AlfrescoRestSiteExtractor - Alfresco Site extractor using Alfresco REST protocol

This task relies on the Alfresco public REST API (with v1.0.4 of the Alfresco REST client) to retrieve sites into a given Alfresco instance.

Mandatory settings

Key	Type	Description	Default value
Alfresco connection provider	AlfrescoRESTConnectionProvider		
AFTS query	`String`	Query used to retrieve all sites from Alfresco	TYPE:"st:site'

CMContentExtractor - Basic content extractor from Content Manager

This class is dedicated to the extraction of content for the Content Manager solution. You'll have the possiblity to extract annotations, custom properties or even logs.

Mandatory settings

Key	Type	Description
CM connection provider	CMConnectionProvider	

Optional settings

Key	Type	Description	Default value
Extact history logs	`Boolean`		`true`
Extract standard system properties	`Boolean`		`true`
Extract advanced system properties from DKDDO object	`Boolean`		`true`
Extract document annotation	`Boolean`		`false`
Extract note logs	`Boolean`		`false`
Extract custom properties	`Boolean`		`true`
Extract note logs as annotations	`Boolean`		`false`
Extract document content	`Boolean`		`true`

CMODContentExtractor - Basic CMOD content extractor

Mandatory settings

Key	Type	Description
CMOD Connection Settings	CMODConnectionProvider	

Optional settings

Key	Type	Description	Default value
Pattern to store resource files	`String`		`${resourceId}`
Export attached CMOD resources	`Boolean`		`true`

DctmContentExtractor - Extract document-related details from Documentum

This Documentum connector is designed for extraction of document versions, metadata, folders and content (only the 1st content of a document) from a Documentum repository. Multiversion documents will be retrieved from the shared 'i_chronicle_id'. Since Documentum architecture involves particular port and access management, a worker should be started on the same server where Documentum is running;

Make sure to check the basic requirements at the setup for Documentum on the official Fast2 documentation.

Optional settings

Key	Type	Description	Default value
Connexion information to	DctmConnectionProvider		

Key	Type	Description	Default value
Documentum Repository			
Extract folders	`Boolean`		`true`
Map empty or unset properties	`Boolean`	Attach Documentum metadata onto document dataset even if the value is missing or unset.	
Extract renditions	`Boolean`	Check this option to extract renditions of each document. They will be attached as side-contents in the document, with properties populated from original renditions properties.	
Whitelist for metadata to extract	`String`	All values need to be separated by comma `,`.	

Key	Type	Description	Default value
Extract metadata	`Boolean`		`true`
Continue on fail	`Boolean`	If `true`, any error which occurs during extraction of either metadata, content or folders will trigger an exception. Otherwise, the error will be found in the logs.	
Extract content	`Boolean`		`true`
Extract all versions	`Boolean`		

FileNet35ContentSource - Extract content from FileNet 3.5

Use this task to retrieve content of documents to extract from a given FileNet instance. This task needs to be preceeded by a FileNet35Source task.

Mandatory settings

Key	Type	Description
FileNet 3.5 connection provider	FileNet35ConnectionProvider	Connection parameters to the FileNet instance

Optional settings

Key	Type	Description	Default value
Ignore documents with zero-sized content	`Boolean`	Document without any content will not be processed	`false`

FileNetContentExtractor - Extract document content from FileNet P8

This task is not a real source task. The documents to be extracted are identified by an BlankSource task generating a set of 'empty' Punnets, i.e. containing only documents each bearing a document number (documentId) to extract.

Mandatory settings

Key	Type	Description
FileNet connection provider	FileNetConnectionProvider	Connection parameters to the FileNet instance

Optional settings

Key	Type	Description	Default value
Property Helper to use	PropertyHelper		
Extract object type properties	`Boolean`	The FileNet P8 metadata of the document which are Object type will be saved at the punnet level	`false`
Compound parent data for children references	`String`	Name of the parent document property under which the children properties will be stored.	
Object store name	`String`	Name of the repository to extract from	
Compound children data to record	`String`	Name of the child property to store in the parent. Consider setting parent data name as well.	
Extract FileNet system properties	`Boolean`	Save the FileNet system properties as document metadata	`false`

Key	Type	Description	Default value
Default mimetype	`String`	Default mimetype to set if the one from FileNet is empty	
Skip annotation exceptions	`Boolean`	Extract documents even if related annotations are in exception like null content	`false`
Extract FileNet security	`Boolean`	The security of the document will be saved at the punnet level	`false`
SQL fetch query	`String`	Use this SQL to fetch documents based on your criteria. Ex/ SELECT [Id], [DocumentTitle] FROM Document WHERE [Property] = `'${myCriterion}'`	
Extract folders absolute path	`Boolean`	The absolute path of the folder inside the FileNet instance will be extracted during the process	`false`
Extract content	`Boolean`	The document content will be extracted during the	`true`

Key	Type	Description	Default value
		process	
Extract all versions	`Boolean`	Extract the superseded versions of the documents matching the query	
Extract annotations	`Boolean`	All annotations owned by the document will be extracted	`true`

FlowerContentExtractor -

Mandatory settings

Key	Type	Description
Flower component category (DOCUMENT, TASK, FOLDER or VIRTUAL_FOLDER)	`String`	

Optional settings

Key	Type	Description	Default value
Extract document annotations	`Boolean`		`false`

Key	Type	Description	Default value
Extract component facts	`Boolean`		`false`
	FlowerDocsConnectionProvider		
Extract document file content	`Boolean`		`false`

IDMISContentExtractor - ImageServices WAL JNI-bridged Extractor

This task extracts documents from the Panagon Image Services ECM (indexes, optional content and annotations). One punnet of one document for each ECM document. However, it's not a real source task. The documents to be extracted are identified by a BlankSource task generating a set of empty Punnets, i.e. containing only documents each bearing a document number (documentId) to extract.

Mandatory settings

Key	Type	Description
Password	`String`	Password of the aforementioned username

Key	Type	Description
Connection organization	`String`	Organization name for the connection
Connection domain	`String`	Domain name of the connection
Username	`String`	Login with scope to access the docbase with proper rights

Optional settings

Key	Type	Description	D
Annotations in ARender format	`Boolean`	Convert annotations to ARender format	`fa`
Annotation converter	ParseISAnnotation	Specific converter from IS format. Allow to resize the extracted annotations	
Annotations in raw format	`Boolean`	Save annotation contents in raw format inside the punnet	`fa`
Version of libIDMIS	`String`	This task is based on the WAL library and on the specific Fast2 library 'libIDMIS.dll'. This library must be in a directory of the Windows PATH. In the wrapper.conf or hmi-wrapper.conf file, activate	`li` `1.`

Key	Type	Description	D
		the use of this library: wrapper.java.library.path.increment = ../libIDMIS/w32For the moment, only 32-bit libraries are configured	
Test scenarios	`Boolean`	Empty testing stub instead of libIDMIS	`fa`
Connection terminal	`String`	Terminal name for the connection	
Use opacity for annotations	`Boolean`		`fa`
Unrecognized annotation file path	`String`	Path of the alternative annotation xml file for unrecognized annotation. If not specified the punnet will go in exception	
Extract document content	`Boolean`	The document will be extracted with its content	`tr`
Extract document annotation	`Boolean`	The associated annotations will be extracted	`tr`

MDOParserExternalContent - Parse FWTF (Fixed Width Text File) with external content to a punnet description

An MDO file is a flat file defined such as: each line corresponds to a document and each line contains information about the document The extraction of information from each line is based on a CSV configuration file, which provides the name of the metadata to be inserted into the punnet document, as well as its characteristics.

It consists of the following columns, separated by a comma:

- Field: name of the metadata to add \n
- Length: length of the metadata. If the value is greater than this length, then it will be truncated. If the value is lower, it will be completed by spaces on the right \n
- Offset: position in MDO file \n
- Mandatory: Y / N \n
- Occurs: number of occurrences allowed for the field. The successive values of the field will then be added to the values of the metadata (respecting the Length parameter for each one) \n
- Type: Type of metadata to add to the punnet document \n

The MDOParserExternalContent task is used to retrieve external content for each document. To do this, the name of the column defining the content path is specified in the task settings.

Mandatory settings

Key	Type	Description
MDO format specification file path	`String`	CSV configuration absolute file path containing MDO format specification

Optional settings

Key	Type	Description	Default value
File scanner	FileScanner	Recovers your files	
Date format	`String`	Date format used in MDO file. Must be the same for each line of the document	`yyyy-MM-dd`
Property name containing path content	`String`	Name of the field in the configuration file that contains the path to the content. If not filled, the content will not be saved in the punnet	
Create one punnet for each document of FWTF	`Boolean`	If true then a punnet with one document will be created for each entry in the MDO file. Otherwise, one punnet will be created containing as	`false`

Key	Type	Description	Default value
		many documents as there are entries in the MDO file	
Dataline property name	`String`	Name of the metadata that will contain the MDO line read. If not specified, the line read will not be saved in the punnet	
contentLocationAbsolute	`Boolean`		
Last punnet property name	`String`	Data name indicating which punnet is the last of document in punnet. If null, data isn't added in punnet. For multipunnet case only	

MDOParserInternalContent - FWTF (Fixed Width Text File) parser with internal content

Like the MDOParserExternalContent task, the MDOParserExternalContent source allows you to parse each line of the MDO file in Punnet. The difference between these two tasks is that the content is stored inside the MDO itself. The start and end of the content is defined by a tag specified in the task settings

Mandatory settings

Key	Type	Description
MDO format specification file path	`String`	CSV configuration absolute file path containing MDO format specification

Optional settings

Key	Type	Description	Default value
File scanner	FileScanner	Recovers your files	
Date format	`String`	Date format used in MDO file. Must be the same for each line of the document	`yyyy-MM-dd`
End tag	`String`	End tag property name signifying the end of the content	
Create one punnet for each document of FWTF	`Boolean`	If true then a punnet with one document will be created for each entry in the MDO file. Otherwise, one punnet will be created containing as many documents as there are entries in the MDO file	`false`

Key	Type	Description	Default value
Dataline property name	`String`	Name of the metadata that will contain the MDO line read. If not specified, the line read will not be saved in the punnet	
Last punnet property name	`String`	Data name indicating which punnet is the last of document in punnet. If null, data isn't added in punnet. For multipunnet case only	
Original text content property name	`String`	Data name containing original text content. If null, data isn't added in the punnet	

OpenTextContentSource - OpenText content extractor using OpenText REST protocol

Mandatory settings

Key	Type	Description
OpenText credentials	OpenTextCredentials	

Key	Type	Description
OpenText client	OpenTextRestClient	

Optional settings

Key	Type	Description	Default value
Extract all versions	`Boolean`	Extract the superseded versions of the documents matching the query	
Extract document metadata	`Boolean`	Save metadata as document metadata	`false`
Extract document categories	`Boolean`	Save categories as document metadata	`false`
Extract content	`Boolean`	The document content will be extracted during the process	`true`
Ticket period	`Integer`	Time in seconds between two ticket creation	`60`

SQLContentExtractor - Extract document content from SQL

Extract clob and blob object-types. Classic types like varchar are extraced as well

Mandatory settings

Key	Type	Description
SQL connection provider	SQLQueryGenericCaller	
SQL query	Pattern	Select precisely documents you want to extract through a classic SQL query

Optional settings

Key	Type	Description
SQL mapping for content	`String/String map`	Mapping of SQL properties to document content.

Catalog / Credentials

AWSConnectionProvider - AWS S3 user credentials

With an access key id and the secret access key, you have the option to connect to an AWS S3 instance by specifying the region concerned. However, to perform this king of connection, Fast2 required the permission : AmazonS3FullAccess

Mandatory settings

Key	Type	Description
Access key Id	`String`	This field is mandatory unless 'Use Instance Profile' is set
Secret access key	`String`	This field is mandatory unless 'Use Instance Profile' is set
AWS Region	`String`	

Optional settings

Key	Type	Description	Default value
AWS URL endpoint	`String`	Service endpoint & signing region	

Key	Type	Description	Default value
Use settings for Snowball	`Boolean`	Snowball S3 endpoint requires specific S3Client settings	`false`
Use Instance Profile instead of Access Key & Secret	`Boolean`	From your local variable, Fast2 will retrieve your connection information	`false`
Role ARN name	`String`		
sessionName	`String`		
AWS extra Client Configuration	ClientConfiguration	Use this AWS class to fine-tune connection details to S3, such as timeouts, connection pool size, ...	

AlfrescoCMISConnectionProvider - CMIS connection provider

From a URI and giving a username with password, this class allow you to access any Alfresco instance

Mandatory settings

Key	Type	Description
Password	`String`	
Username	`String`	
URI for connection settings	`String`	

AlfrescoRESTConnectionProvider -

Mandatory settings

Key	Type	Description
Password	`String`	
URL to connect to Alfresco	`String`	
Username	`String`	

CMConnectionProvider - Connection provider for Content Manager solution

The CM connection provider will help you to manage a pool of connections. For performance reasons, it is sometimes desirable to limit the number of connections created by the pool.The connection pool will allow you to specify the maximum number of connections that should exist at one time, whether in use or in the pool.Once this maximum value is reached, an error may be thrown or you may optionally wait for an existing connection to be freed

Mandatory settings

Key	Type	Description	Default value
Password	`String`	Password of the aforementioned username	
Username	`String`	Login with scope to access the docbase with proper rights	`icmadmin`

Optional settings

Key	Type	Description	Default value
Data source type	`String`		`ICM`
Connection pool size	`Integer`	Maximum number of connections to be created	`64`
Server name	`String`	Name of the server involved in the migration	`ICMNLSDB`
Connection free pool size	`Integer`	Maximum number of connections that may be held in the free pool	`5`

Key	Type	Description	Default value
Internal connection	`Integer`	Maximum number of connections for internal side	`64`
Connection duration	`Long`	Length of time to kill a free connection in milliseconds	`100000`

CMODConnectionProvider - CMOD connection provider

With a username / password and an IP address, this class allow you to connect at your CMOD instance.To optimize connections between Fast2 and CMOD you can use a single connection

Mandatory settings

Key	Type	Description	Default value
Server IP address	`String`		`192.168.0.189`
Password	`String`		
Username	`String`		`admin`

Optional settings

Key	Type	Description	Default value
Port number	`Integer`		`1445`
Singleton connection	`Boolean`	Optimization of the connection in case of regular calls	`false`

DctmConnectionProvider - Documentum connection provider

Module used by Fast2 to establish to communication with the destination Documentum instance.

Mandatory settings

Key	Type	Description
Docbase name	`String`	Name of the docbase involved in the migration.
Password	`String`	Password of the aforementioned username.
Username	`String`	Login with scope to access the docbase with proper rights.

DctmSshClient - Documentum SSH client"

Module used by Fast2 to establish the connection with the destination Documentum server.

WARNING
Using this feature may significantly degrade performance. To use this option, the Client must be set on all Documentum tasks.

Mandatory settings

Key	Type	Description
Username	`String`	Documentum machine username
Password	`String`	Documentum machine password
IP	`String`	Documentum machine IP

EmbeddedDbConnectionProvider - OpenSearch connection provider

Module used by Fast2 to connect to its own database.

Mandatory settings

Key	Type	Description	Default value
Database endPoint	`String`	The endpoint of the Fast2 database	`"http://localhost:1790`

FileNet35ConnectionProvider - Connection provider for FileNet 3.5 solution

This task is used to provide connection information to connect specifically to the FileNet P8 3.5 ECM.

Mandatory settings

Key	Type	Description
Password	`String`	Password of the aforementioned username
Username	`String`	Login with scope to access the docbase with proper rights

Optional settings

Key	Type	Description
WCM Config resource	`String`	
Object store name	`String`	Name of the docbase involved in the migration
URL settings	WcmApiConfigSettings	Class used for setting multiple URLs (download, upload...)

FileNetConnectionProvider - Connection provider for FileNet P8 solution

Using this class allows you to provide connection information to specifically connect to your FileNet P8 5.x ECM

Mandatory settings

Key	Type	Description
Password	`String`	Password of the aforementioned username
URI address	`String`	URI to determine which FileNet instance to connect to Ex/ `http://<ip>:<port>/wsi/FNCEWS40MTOM/`
Username	`String`	Login with scope to access the docbase with proper rights

Optional settings

Key	Type	Description	Default value
JAAS coonfiguration stanza property name	`String`	Property containing JAAS coonfiguration. If null, the default JAAS stanza name is set to FileNetP8	

Key	Type	Description	Default value
Singleton connection	`Boolean`	Reuse the same connection to optimize calls	`false`
Initial naming factory	`String`		

FlowerDocsConnectionProvider - Connection module for FlowerDocs ECM

Module responsible for authentication of Fast2 for FlowerDocs

Mandatory settings

Key	Type	Description
URL endpoint	`String`	Web services target URL. For example : https://www.demo.flowerdocs.cloud/flower-docs-ws/services
Password	`String`	Password of the service account used for authentication
Scope	`String`	Scope of the service account used for authentication
Username	`String`	Username of the service account used for authentication

Optional settings

Key	Type	Description	Default value
Overrides authenticated user	`Boolean`	Allows connections to multiple endpoints	`false`

GenericRestClient - Generic REST client connection

Allows Fast2 to connect to any REST API.

Mandatory settings

Key	Type	Description
Endpoint	`Pattern`	Complete URI address of the endpoint
Method	`String`	HTTP method to use Ex/ GET, POST, PUT, DELETE

Optional settings

Key	Type	Description	Example	Defau valu
Header	`String/Pattern map`	Parameters of the header		

Key	Type	Description	Example	Defau valu
Query parameters	`String/Pattern map`			
Body content type in request	`String`		application/json	
Proxy host	`String`			
Proxy port	`Integer`			
Proxy username	`String`			
Proxy password	`String`			
Socket timeout	`Integer`	Timeout to receive data (in ms)		60000
Request timeout	`Integer`	Timeout until a connection with the server is established (in ms)		30000

Key	Type	Description	Example	Defau valu
Total number of connections	`Integer`	Set the maximum number of total open connections	2048	
Max connections per route	`Integer`	Set the maximum number of concurrent connections per route		2048
Form field body	`String/Pattern map`	Field used to send structured data in Key/Value pairs. Required for content types: multipart/form-data, and application/x-www-form-urlencoded. FILE UPLOADS (Multipart) : To include files within a form,		

Key	Type	Description	Example	Defau valu
		enter the absolute or relative file path as the value and add '_file' at the end of the corresponding key.		
Raw body	`Pattern`	Field used when the request body is a single content. Required for: text/plain, application/xml, and all single binary file uploads (e.g., image/*, application/pdf). FILE UPLOADS : Enter the file path and check the file path option. Otherwise, enter the literal		

Key	Type	Description	Example	Defau valu
		content (text or XML string).		
File Path	`Boolean`	Indicates if the raw body contains a file path		`false`

MailBoxProvider - Mail box connection

This class is used to access any mailbox from some connection information.

Mandatory settings

Key	Type	Description
Protocol	`String`	Protocol used to establish the connection Ex/ imap
Password	`String`	Password of the aforementioned username
Host address	`String`	Exact address of the mail server where to retrieve the mails Ex/ imap.gmail.com

Key	Type	Description
Root folder name	`String`	Name of the root folder to extract mails
Username	`String`	Login with scope to access the docbase with proper rights

Optional settings

Key	Type	Description	Default value
Session debugging	`Boolean`	Keep logs written for the connection into stdout	`false`
Number of retries	`Integer`	Maximum number of times to retry the connection in case of failure	`0`
Read and write permissions	`Boolean`	Open mail session with read and write permissions. If false, the session is only readable	`false`
Extended properties map	`String/String map`	List of additional properties to apply Ex/ myValue.toAdd = true	
Time between two	`Integer`	Time in milliseconds between each connection	`1000`

Key	Type	Description	Default value
connections		attempt	

MailSenderProvider - Email connection provider

This module will grant Fast2 access to send emails on behalf of a given user

Mandatory settings

Key	Type	Description
Password	`String`	The password of the aforementioned user
Properties	`String/String map`	It is expected that the client supplies values for the properties listed in Appendix A of the JavaMail spec. Please provide -mail.store.protocol, -mail.transport.protocol, -mail.host, -mail.user, and -mail.from as the defaults are unlikely to work in all cases
Username	`String`	The username with proper rights to access the email client server

Optional settings

Key	Type	Description
Debug	`Boolean`	Set the debug setting for this Session. Since the debug setting can be turned on only after the Session has been created, to turn on debugging in the Session constructor, set the property `mail.debug` in the Properties object passed in to the constructor to `true`

MFilesConnectionProvider - Connection settings for MFiles

Credentials to connect to M-Files remote system via its REST API.

Mandatory settings

Key	Type	Description	Default value
M-Files REST endpoint	`String`		http://ip.address/REST
Login	`String`		
Password	`String`		
Vault GUID	`String`	ex : `{15c876e7-8462-4a35-83d6-c8c21694eed6}`	

MobiusConnectionProvider - Mobius connection provider

> ⚠️ **Deleted**: The `MobiusConnectionProvider` module is deleted and no longer available in Fast2 from v2025.0.0.

This Mobius connection module is required for Fast2 to successfully establish the connection with your Mobius instance in order to properly migrate metadata and contents.

Mandatory settings

Key	Type	Description
Repository ID	`String`	The Universally unique identifier (UUID) of the destination repository
Mobius Server URL	`String`	

Optional settings

Key	Type	Description
Authentication REST Header	`String`	The value of the 'Authorization' header of the REST request

NuxeoConnectionProvider - Connection settings for Nuxeo

Mandatory settings

Key	Type	Description	Default value	
URL endpoint	`String`	`http://hostname:port/nuxeo`		`http://lo`
Password	`String`			
UserName	`String`			

Optional settings

Key	Type	Description	Default value	Example
Accessible schemas	`String`	List of document schemas accessible with this connexion	`*`	

OpenTextCredentials - OpenText user credentials

Mandatory settings

Key	Type	Description
Password	`String`	
Username	`String`	

SQLQueryGenericCaller - Generic query caller

This modules is responsible of establishing the connection between Fast2 and the designated database

Mandatory settings

Key	Type	Description

Connection definition	`String`	Use a standard jdbc:// syntax. If a driver is needed, the JAR file has to be added to the worker-libs/ folder. Make sure to pick up a version compatible with the JDK used by Fast2. If clear credentials is a problem, please use below fields username and password
		Ex/
		`jdbc:sqlite:C:/sqlite/mydatabase.db;`
		`jdbc:sqlserver://localhost:<port>;user=...;password=...;`
		`jdbc:mysql://<ip-address>:3306/<db-name>?user=...&password=...`

Optional settings

Key	Type	Description
Password	`String`	Password used by connectionString and fully encrypted for security reasons

Key	Type	Description
Driver class	`String`	Optional driver class to load before connection. Leave empty to load none
Throw error if no result	`Boolean`	Throw exception when SQLQueryColumnCaller finds no result.
Skip exceptions	`Boolean`	Fast2 will either throw an error if the statement has not properly been executed, or fail silently
User	`String`	Username used by connectionString

Catalog / Convertion tasks

ArchiveBuilder - Punnet zipper

Zip punnets into a zip file. A zip cannot contains a few punnets but documents carried by the punnet can be splitted into multiple zip files.

Optional settings

Key	Type	Description	Default value
Zip all documents in a single zip file	`Boolean`		`false`
Zip size limit	`Integer`	In megabytes	
Maximum number of documents per zip	`Integer`		
The compression level between 0 and 9 (included)	`Integer`	Use large number for stronger compression	
Also include annotation contents	`Boolean`		`false`

ConvertCMToP8 - Convert CM annotations to FileNet P8 annotations. Input annotation format is INI from a JSON file. Only supports one document per punnet. Supported contents are PDF, TIFF and PNG files

Supported types :

- CMBAnnotationConstants.ANN_TEXT
- CMBAnnotationConstants.ANN_ARROW
- CMBAnnotationConstants.ANN_HIGHLIGHT
- CMBAnnotationConstants.ANN_NOTE
- CMBAnnotationConstants.ANN_RECT
- CMBAnnotationConstants.ANN_LINE
- CMBAnnotationConstants.ANN_STAMP
- CMBAnnotationConstants.ANN_NOTE
- CMBAnnotationConstants.ANN_MASK
- Polygon
- Freehand

Optional settings

Key	Type	Description	Default value
ViewOne annotation	ViewOneAnnotationConverter		

Key	Type	Description	Default value
parser			
Force rectangle transparency	`Boolean`	Force rectangles to be transparent even if dedicated property equals 0	
Apply ratio	`Boolean`	Transform annotation positions using document DPI	`true`
Transform rectangle into highlight	`Boolean`	Rectangles does not support transparency. Turn them into highlight to keep tranparency	`false`

Key	Type	Description	Default value
Watermarks on all pages	`Boolean`	Apply watermark annotations on each pages of the document	`false`

ConvertDoc - Document conversion

Convert your documents using the ConvertDoc library.

Optional settings

Key	Type	Description	Default value
Maximum number of document per call	`Integer`		`200`
Temp folder to generate PDF to	`String`		

Key	Type	Description	Default value
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Excel hack	`Boolean`	Use Microsoft Excel hack	`true`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Conversion exception is fatal	`Boolean`	Conversion exception triggers an exception. Othersie, it's a silent fail	`true`
	`Boolean`		
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check	`Boolean`	You can assume the file extension is accurate, or	`false`

Key	Type	Description	Default value
document before content		ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	
ConvertDoc path	`String`	Path toward ConvertDoc executable	`ConvertDoc.exe`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Temp folder to copy source files to	`String`		
Convert to PDFA	`Boolean`	Convert your document to PDFA format	`false`

ConvertINIToXFDF - Annotation converter from INI to XFDF

Convert ViewOne annotations to XML Form Data Format (XFDF). These operations are supported only for PDF and Tiff files. Content dimensions will be fetched to convert these annotations To improve the performances add these 3

parameters to the startup-worker.bat script before the -jar :
'-Dorg.ini4j.spi.IniBuilder=org.ini4j.spi.IniBuilder'
'-
Djavax.xml.transform.TransformerFactory=com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl'
'-Dorg.ini4j.spi.IniParser=org.ini4j.spi.IniParser'

Supported types :
- [TEXT]
- [RECTANGLE]
- [FREEHAND]

Optional settings

Key	Type	Description	Default value
Postit annotation position	PageRelativePosition	Default position when an exception is thrown	
Overload text and postit page to the first one if related exception is thrown	`Boolean`		
Skip conversion error	`Boolean`	Do not throw an exception for conversion errors	`false`

Key	Type	Description	Default value
All note font are black	`Boolean`		
Keep converted annotations	`Boolean`	Do not interrupt punnets if at least one annotation has been converted. Punnets carrying annotations that haven't been converted will be flagged with a data named 'ToReplay'	`false`
Text annotation position	PageRelativePosition	Default position when an exception is thrown	

ConvertISToFDF - Annotation converter from IS to FDF

Convert your old IS annoations to Form Data Format (FDF). FDF file are used to represents form data and annotations in a PDF file with key/value format. They usually contain more information than XFDFs

Optional settings

Key	Type	Description	Default value
Skip conversion error	`Boolean`	Do not throw an exception when a conversion is triggered	`true`
FileNet XFDF annotation converter	FileNetXFDFAnnotationConverter		
IsAnnotation parser	ParseISAnnotation		
FDF annotation adder	FdfAnnotationAdder		

ConvertISToXFDF - Annotation converter from IS to XFDF

Convert your old IS annoations to XML Form Data Format (XFDF). XFDF file are used to represents form data and annotations in a PDF file through an XML. They usually contain more information than XFDFs

Optional settings

Key	Type	Description	Default value
Popup width	`Float`	Sticky note default popup width	`120`
Skip conversion error	`Boolean`	Do not throw an exception when a conversion is triggered	`true`
FileNet XFDF annotation converter	FileNetXFDFAnnotationConverter		
IsAnnotation parser	ParseISAnnotation		
Popup height	`Float`	Sticky note default popup height	`30`

ConvertLighterPdf - PDF converter

This class allows you to compress your PDF files in order to be lighter than original ones.

Optional settings

Key	Type	Description	Default value
	`Boolean`		
Standard DPI	`Integer`	DPI of a standard PDF document	`72`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Target image format	`String`		`jpg`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Skip invalid image conversion	`Boolean`	Skip image conversion if image is invalid / not supported. Keep the original file	`true`
Maximum converted image ratio	`Float`	After conversion, compute the size ratio before and after. Do not replace source image when converted image size is larger to this ratio	`1.0`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

Key	Type	Description	Default value
Larger threshold to convert	`Integer`	Only convert images which are larger than this minimum byte size	`2000`
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
	Map		
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Maximum ratio to apply	`Float`	Images will not beconverted if the ratio computation is larger to this value	`1.0`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`

Key	Type	Description	Default value
Target compression quality	`Float`		`0.7`
Target DPI	`Integer`	Required target Image dpi	`300`

ConvertP8ToXFDF - Annotation converter from IS to XFDF

Convert your old IS annotations to XML Form Data Format (XFDF). XFDF file are used to represents form data and annotations in a PDF file through an XML. They usually contain more information than XFDFs

Optional settings

Key	Type	Description
Skip conversion error	`Boolean`	Do not throw an exception for conversion errors
Default page height	`Float`	Default page height used for text and mail document
Remove annotations border	`Boolean`	Most of the time, annotations have by default

Key	Type	Description
		a border set to 1. Set to true will remove the border
Keep background transparency	`Boolean`	Make background transparent when it is white
Default page width	`Float`	Default page width used for text and mail document
Font size ratio	`Integer`	Multiply the font size value by this ratio
FileNet XFDF annotation converter	FileNetAnnotationConverter	
Default post it location	PageRelativePosition	Overwrite coordinates for each post it annotation
Create one annotation container per annotation	`Boolean`	
Overwrite border width	`Integer`	Value of border width for proprietary and arrow annotations

Key	Type	Description
Date format	`String`	Date format of your properties stored by annotations Ex/ 2021-10-01T12:04:56.0000765+0200
Highlight opacity	`Integer`	Overwrite the opacity in percent of highlights annotations Ex/ 30

ConvertRipole - Convert ripole files

> ⚠️ **Deleted**: The `ConvertRipole` task is deleted and no longer available in Fast2 from v2025.0.0.

Convert ripoles files to PDF format.

Optional settings

Key	Type	Description	Default value
Temp executable path	`String`		`/tmp/temp_`
Ripole executable path	`String`		

ConvertTalk - Talk converter

Convert your talk files to PDF using iText Library.

Optional settings

Key	Type	Description	Default value
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Maximum input text size	`Long`	Limit in bytes	`1024 * 1024`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Skip PDF creation	`Boolean`	Only perform text parsing	`false`
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the	`true`

Key	Type	Description	Default value
		action has not been properly completed	
Auto-detect encoding	`Boolean`	Automatically detect the source text encoding used	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Keep talk header	`Boolean`	Do not parse the talk header and keep the original one	`false`
Source text encoding	`String`	Encoding used for the source files Ex/ CP850	
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Font description	`String`	Format as {font family} {size} {style} where font family is one of {COURIER, HELVETICA, TIMES_ROMAN, SYMBOL, ZAPFDINGBATS} and style is	

Key	Type	Description	Default value
		0:normal 1:bold 2:italic 4:underline 8:strikethru	

ConvertText - Text converter

Optional settings

Key	Type	Description	Default value
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Maximum input text size	`Long`	Limit in bytes	`1024 * 1024`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Skip PDF creation	`Boolean`	Only perform text parsing	`false`

Key	Type	Description	Default value
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Auto-detect encoding	`Boolean`	Automatically detect the source text encoding used	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Source text encoding	`String`	Encoding used for the source files Ex/ CP850	
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Font description	`String`	Format as {font family} {size} {style} where font family is one of {COURIER, HELVETICA, TIMES_ROMAN, SYMBOL, ZAPFDINGBATS} and style is	

Key	Type	Description	Default value
		0:normal 1:bold 2:italic 4:underline 8:strikethru	

ConvertWangToXFDF - Convert Wang annotations to XFDF

Optional settings

Key	Type	Description	Default value
Charset used for text	`String`		`windows-1252`
Tiff embed annotations	`Boolean`		
Page number property name	`String`	Property of the content used to put the annotation on the correct page content. If null, the annotation will be on first page.	
Fail if no annotation was parsed	`Boolean`		
Skip annotation parse	`Boolean`		

Key	Type	Description	Default value
exceptions			
	`Integer`		

ConvertXFDFToP8 - Annotation converter from XFDF to IS

Convert your XFDF annotations P8 format. This is mostly used to rollback from P8 to XFDF during complex migration.

Optional settings

Key	Type	Description	Default value
Skip conversion error	`Boolean`	Do not throw an exception when a conversion is triggered	`true`

ConverterCMToXFDF - Convert CM annotations to XFDF annotations

Supported types :

- CMBAnnotationConstants.ANN_TEXT
- CMBAnnotationConstants.ANN_ARROW

- CMBAnnotationConstants.ANN_HIGHLIGHT
- CMBAnnotationConstants.ANN_STAMP
- CMBAnnotationConstants.ANN_RECT

Optional settings

Key	Type	Description
fontConversionList	`String`	
	PageRelativePosition	
	double	
	`Float`	
	`Integer`	
	PageRelativePosition	
	`String`	
	`String`	
	[I	
	PageRelativePosition	
	`Float`	
	`Float`	

Key	Type	Description
	`String`	
	`Integer`	
	`Boolean`	
	`Float`	
	`String`	
	`Float`	

DispatchingArchive - Unzip files

This class allow you to unzip the content of archive files. Multiple mime types are supported : application/zip, application/x-zip, application/x-zip-compressed, application/x-rar-compressed, application/x-rar and application/java-archive. Tar or gz folders are not supported yet.

Optional settings

Key	Type	Description	Default value
Charset	`String`	Charset used for file names in zip archives	`CP437`
Process annotation	`Boolean`	If annotations are asked to be migrated, you can filter here to	`false`

Key	Type	Description	Default value
contents		process their content(s) or only their metadata	
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Only extract files that do not match this pattern	`String`		
Delete source archive file	`Boolean`		`false`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Recursive	`Boolean`		`false`
Only extract files that match this pattern	`String`		
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the	`true`

Key	Type	Description	Default value
		action has not been properly completed	
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`

Eml2Pdf - Convert email to PDF

This class allow you to convert emails to a PDF format. Formats supported are application/msword, rfc822 and outlook.

Optional settings

Key	Type	Description
rtf tags list	`String list`	For mapimessageparser if provided, can allow to override the

Key	Type	Description
		list of rtf tags to filter out
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones
Custom mail header	`String`	Mail header to prefix the subject of the mail as pdf title in ARender
includeInlineBodyWithHeaders	`Boolean`	
Convert Configuration	Eml2PdfConfiguration	Mail to pdf convert options. Can be null
Mail subject in title	`Boolean`	Display mail subject in title

Key	Type	Description
Attachment in separate folder	`Boolean`	All attachments are included in a separated folder
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert
Separator	`String`	Separator between prefix and mail subject
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed
Attachment in included folder	`Boolean`	Each attachment is included in a separated folder
Mime-type : Check document before content	`Boolean`	You can assume the file extension is

Key	Type	Description
		accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level
Custom body header	`String`	Body header to swap the default -Content- header set to the body
Supported inline body mime-types	`String list`	
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all

ExcelConvert - Convert Excel to PDF

> ⚠️ **Deleted**: The `ExcelConvert` module is deleted and no longer available in Fast2 from v2025.0.0.

This class allow you to convert an Excel file to a PDF format.

Optional settings

Key	Type	Description	Default value
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`

Key	Type	Description	Default value
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

GenericConvertDoc - Convert from URL to PDF from its URL

This task will convert the content of any document into PDF format. All Fast2 needs is you to specify the URL/path of the initial document. The input file will be automatically resolved by the documents carried by the punnet. The ouput file path got his own field that can be used with a patternThe location of options are managed through an int value specifying the index within the cmd line starting to 0.

Mandatory settings

Key	Type	Description
Destination folder	`String`	Target file path for the locally-created files
Converter path	`String`	Set here the path of the converter

Optional settings

Key	Type	Description	Default value
Ingore output file	`Boolean`	Converter does not takes output file instructions	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Ignore command return value	`Boolean`	Do not perform anything after conversion, whatever the command feedback	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Destination folder for original files	`String`	Dump input file in a specific folder path. Set to null to disable dumps	
Options location	`Integer`	Set here the index of options in the cmd line : O means at the beginning.	`1`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

Key	Type	Description	Default value
Protect path with double quotes	`Boolean`	Surround file paths with quotes when building command-line	`false`
Timeout before stopping	`Integer`	Time to wait before killing the process. Value is in seconds	`3600`
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Conversion option	`String`	See application help for more details	
List of error exit codes to skip	`String list`		

Key	Type	Description	Default value
Extension to append	`String`	The extension to add to the original file name before any conversion. This value should not start with a dot (ex : html)	
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`

HtmlCleanup - HTML cleaner

Utility Class to cleanup inconsistent HTML

Optional settings

Key	Type	Description	Default value
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`

Key	Type	Description	Default value
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

IdentifyImageFormat - Detect image format using ImageMagick

Find automatically the image format with ImageMagick in command line.
Several properties can be added and the assiociated command line filled int the property pattern field

Mandatory settings

Key	Type	Description
Property map	`String/Pattern map`	List of the porperties to use during the process Ex/ targetProperty = documentStringGetter

Optional settings

Key	Type	Description
identify.exe path	`String`	ImageMagick identify.exe path to use

JaTiffMerger - Merge tiff files

From a tiff content punnet, merges all its subcontent merge together. All content must be in tiff format. If the subcontents are already lower than 2 images, the merge is cancelled.

Optional settings

Key	Type	Description	Default value
Throw conversion	`Boolean`	If Fast2 performs document conversion, it can either fail	`true`

Key	Type	Description	Default value
exceptions		silently or pop an error when the action has not been properly completed	
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

JaTiffSplitter - Tiff document splitter

Optional settings

Key	Type	Description	Default value
Dump tiff information	`Boolean`		`false`
Minimum pages for dispatching	`Integer`		`1`
Corrupted JPEGs handled by ImageMagick	`Boolean`		`true`
Handle corrupted JPEGs	`Boolean`		`true`
Sanitize source JPEGs	SanitizeJpeg	> ⚠️ **Deleted**: The `SanitizeJpeg` module is deleted and no longer available in Fast2 from v2025.0.0.	
Maximum number of source pages	`Integer`		`Integer.MAX_VALUE`

Key	Type	Description	Default value
Force JPEG Sanitation	`Boolean`		`false`
Dispatch multi-pages tiff to a bunch of single-page tiff	`Boolean`	Only the first one will be processed when source contains multiple multi-page tiffs	`false`
Minimum number of pages before dispatch errors	`Integer`		`1`

JaTiffWang - Extract wang annotations from Tiff document

Creates one annotation file perf tiff page where annotations are found. A property is set to the annotation content to get back the page

Optional settings

Key	Type	Description
Generate wang hexa in logs	`Boolean`	
Fail if no annotation was found	`Boolean`	

Key	Type	Description
Skip exceptions	`Boolean`	

MDOWriter - Write punnet description to a MDO-format

This task serializes document metadata in an MDO file format with a fixed length.

Mandatory settings

Key	Type	Description
Target file path for (local) files, can be pattern based	`String`	

Optional settings

Key	Type	Description	Default value
Throw exception on missing mandatory data	`Boolean`	Throw exceptions if mandatories data are missing. Otherwise, silent fail	`false`
MDO format specification file path	`String`	CSV configuration file path containing MDO format specification	

Key	Type	Description	Default value
End tag of document content	`String`		
Data name to add dataline into document data	`String`	If null data isn't added in document	
Skip from specific index	`Integer`	Skip writing documents from this index	`0`
Fallback on missing data	`Boolean`	When data is missing in the document being written, try to search it in the first document	`true`
MDO-format with internal content	`Boolean`	Generate MDO-format document with internal content	`false`
Data name containing original text content	`String`	If null text content isn't added to MDO file, internal content only	

MergeAllContents - Merge multiple content

Merge all contents of document from the first depth level. Can be used after Eml2Pdf task to merge header and body.

Optional settings

Key	Type	Description	Default value
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`

Key	Type	Description	Default value
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

MergeAllMails - Merge multiple mails

Merge mail header and body after a mail conversion

Optional settings

Key	Type	Description	Default value
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`

Key	Type	Description	Default value
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

MergePdfBox - PDF merger

Merge your tiff files into PDF format using the PDFBox library (v1.2.1).

Optional settings

Key	Type	Description	Default value
Remove document direct content	`Boolean`	Not referenced from folders (deprecated)	`false`
Convert resursive folders	`Boolean`	Folders referencesd by the document are converted recursively (deprecated)	`false`
Ignore conversion exceptions	`Boolean`	Each exception during the conversion becomes a silent fail indexed in logs	`false`
Supported source mime types	`String list`	List of all accepted mime types	

OOConvert - Convert office file to PDF

> ⚠️ **Deleted**: The `OOConvert` module is deleted and no longer available in Fast2 from v2025.0.0.

Complete converter from office file to PDF format using OpenOffice / LibreOffice.

Mandatory settings

Key	Type	Description
Target path of generated file	`String`	Path to use for generated files during the conversion

Optional settings

Key	Type	Description	Defaul
Temp folder path	`String`	Path to the temp folder	
Output mime type	`String`	Mime type to set at the end of the conversion	`applica`
Delay to despair after an Office operation	`Long`	Delay in milliseconds	`15 * 10`

Key	Type	Description	Defaul
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
	DefaultOfficeManagerConfiguration		
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Shutdown Office Manager at finishTask)	`Boolean`	At the end of the task force Office Manager to shutdown	`false`
Supported mime-types	`String list`	Specify the list of all mime-types of documents	

Key	Type	Description	Defaul
		which Fast2 will convert	
Localhost port number	`Integer`		`8100`
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document	`false`

Key	Type	Description	Defaul
		mime-type. By default, Fast2 will check at content level	
Delay to startup Office Manager	`Integer`	Delay in seconds	`0`
Shutdown Office Manager when exiting	`Boolean`	Force Office Manager to shutdown at the end of the workflow	`false`
Office Manager as singleton	`Boolean`	Use static singleton OfficeManager to reuse process between two campaigns	`false`
officeHome	`String`		
Process all contents	`Boolean`	Fast2 will either only focus on the	`true`

Key	Type	Description	Defaul
		first encountered content, or process them all	
Stop cmd for LibreOffice / OpenOffice	`String`	Command to use to force LibreOffice or OpenOffice process to stop	

PdfAConverter - Convert from PDF to PDF/A

A PDF/A is a PDF file with some constraints to ensure its long time conservation. These constraints are described in ISO 19005. This task takes PDF files as input, and generated a PDF/A-`{1A, 1B, 2A, 2B, 3A, 3B}`.

Optional settings

Key	Type	Description	Default value
Expected pdf conformity level	`String`	If empty default level is PDF/A-1B	`1B`

PdfAnnotationRenderer - Renders annotations into a new PDF document

Requires itext-5.5.13, xmlworker-5.5.13 and jsoup-1.12.2 libs

Optional settings

Key	Type	Description
	`Boolean`	
	`Float`	
	`Float`	
	`Float`	
	`String`	
	`Boolean`	
	`Boolean`	
	`Float`	
	`Boolean`	
	`Float`	
	`String`	
extraFont	`String/String map`	

PunnetXSLSerializer - Export punnet metadata using XSL script

Serialize a punnet to any file (CSV, JSON, XML, custom format) using an XSL stylesheet.

Mandatory settings

Key	Type	Description
XSL stylesheet path	`String`	Or you can enter your xsl:stylesheet in the content section
XSL Stylesheet content	`String`	Enter here your xsl:stylesheet content

Optional settings

Key	Type	Description	Default value
New document properties	`String/String map`	Specify here at least the `documentId` data. You can use punnet properties to resolve pattern Ex/ Setting an entry with 'myKey' and 'myValue' will trigger Fast2 to look up for the metadata entitled 'myValue' in the punnet and its document(s). If this data is not	

Key	Type	Description	Default value
		found, the value will be set to 'myValue'.	
Append	`Boolean`	Attach output stream of you XSL script as a new document in the punnet.	`false`
Replace	`Boolean`	Replace punnet documents by the new produced document (requires Append to be set)	`false`
Encoding	`String`	Enter here your script file encoding.	`UTF-8`

SanitizeTiff - Tiff cleaner

Converts your tiff files to jpeg format.

Optional settings

Key	Type	Description	Default value
Out mime type	`String`	The target mime type for your documents	`image/tiff`

Key	Type	Description	Default value
Out extension	`String`	The extensions to set to your documents	`tiff`
Convert path	`String`		`convert -quiet`
Supported source mime types	`String list`		

SplitPdfItext - PDF splitter

Mandatory settings

Key	Type	Description	Default value
Split definition data name	`String`	Name of the document data for split definition	`splitDefinitions`

Optional settings

Key	Type	Description	Default value
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`

Key	Type	Description	Default value
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

TesseractInvoker - Transform your PDFs and images with text to make them searchable using OCR engine

Based on Tesseract solution, this task will parse each page of your images (TIFF, JPG and PNG) or PDF to extract text and create a new searchable PDF CCITT T.6 image compression is supported but not LZW compression\nBe careful if you are doing multi-document or multi-content : if names are identicals it will overwrite contents (ex\ sample.tiff & sample.gif will both creates sample.pdf. The first doc will overwrote the 2nd one

Mandatory settings

Key	Type	Description
Destination folder	`String`	Target file path for the locally-created files
Tesseract path	`String`	Complete path of your local Tesseract instance Ex/ /usr/share/tesseract-ocr/4.00/tessdata

Optional settings

Key	Type	Description	Default value
Throw rotations exceptions	`Boolean`	If Fast2 performs document orientation detection, it can either fail silently or pop an error when the action has not been properly completed. Throw conversion exceptions should be set to true for this option to work	`false`

Key	Type	Description	Default value
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Allow rotation if needed	`Boolean`	If document is converted but not readable, rotation is will correct the page orientation from landscape to portrait if needed	`true`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document	`false`

Key	Type	Description	Default value
before content		mime-type. By default, Fast2 will check at content level	
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Add file sizes	`Boolean`	If true, add input and output file sizes in the data set	
Allow enhanced orientation detection	`Boolean`	Only used if allow rotation is set to true. Enable the use of an enhanced rotation detection, thus giving more accurate results but with the downside of being slower. Activate it for low-quality images for example.	`false`

Text2PDFConverter - Convert Text file to PDF

Convert your text files into PDF format.

Optional settings

Key	Type	Description	Default value
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their	`false`

Key	Type	Description	Default value
		content(s) or only their metadata	
Image path	`String`		
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Formatted page text provider	FormattedPageTextProvider		
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Formatted page text converter	FormattedPageTextConverter		
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

Tiff2PDFIText - Converter from Tiff to PDF

This task uses the IText library to convert content of TIFF documents into PDF format

Optional settings

Key	Type	Description	Default value
Keep image aspect	`Boolean`	Zoom image to fit in an A4 paper, but conserve aspect ratio	`true`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`

Key	Type	Description	Default value
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Adapt PDF size to source image	`Boolean`	Generate a PDF document with a size related to the original image size	`false`
Ignore conversion exceptions	`Boolean`	Fast2 will either throw an error if the image has not properly been converted, or fail silently	`false`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

Key	Type	Description	Default value
Verbose logs	`Boolean`	Check this item to have more logs for fine-tuning stage	`false`
imageSourceHeightProperty	`String`		
Temp file cleaner	TempFileCleaner	Select here the module you want to clean the temporary files	
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`

Key	Type	Description	Default value
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
DPI correction	`Boolean`	Check this to correct the DPI related to the image dimensions	`false`
Rotate landscapes	`Boolean`	If imge aspect is > 1.4, it will be rotated to fit in a A4 format	`false`

Key	Type	Description	Default value
Clean temporary files	`Boolean`	Remove the temporary input files. To use this option, a temp file cleaner must be set	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
imageSourceWidthProperty	`String`		

Tiff2PdfBox - Convert TIFF to PDF

This task converts TIFF images into PDF documents using the Apache PDFBox lib

Optional settings

Key	Type	Description	Default value
Add pdf dimensions to dataSet	`Boolean`	Store dimensions in document dataSet named contentHeight and contentWidth only for the first page	
Ignore conversion exceptions	`Boolean`	Fast2 will either throw an error if the image has not properly been converted, or fail silently	`false`
Force to re-encode tiffs	`Boolean`	Force to decompress and recompress tiffs before encapsulating in PDF, slower but could handle some exceptions	`false`
Unsupported producers	`String list`	List of unsupported tiff software producers	
Default DPI used for PDF transformation	`Integer`		`200`

WkHtmlToPdfConverter - Converter from Html To PDF

This task will be used to convert HTML content into a PDF document. Fast2 embeds the wkhtmltopdf command-line utility in order to carry out this conversion.

Mandatory settings

Key	Type	Description
Destination folder	`String`	Target file path for the locally-created files
Converter path	`String`	Set here the path of the converter

Optional settings

Key	Type	Description	Default value
Ingore output file	`Boolean`	Converter does not takes output file instructions	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Ignore command return value	`Boolean`	Do not perform anything after conversion, whatever the command feedback	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`

Key	Type	Description	Default value
Destination folder for original files	`String`	Dump input file in a specific folder path. Set to null to disable dumps	
Options location	`Integer`	Set here the index of options in the cmd line : O means at the beginning.	`1`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Protect path with double quotes	`Boolean`	Surround file paths with quotes when building command-line	`false`
Timeout before stopping	`Integer`	Time to wait before killing the process. Value is in seconds	`3600`
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`

Key	Type	Description	Default value
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Conversion option	`String`	See application help for more details	
List of error exit codes to skip	`String list`		
Extension to append	`String`	The extension to add to the original file name before any conversion. This value should not start with a dot (ex : html)	
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Cleanup HTML first	`Boolean`	Ask Fast2 to clean the HTML content regarding syntax and encoding	`true`

Catalog / Transformation tasks

AlterDocumentContent - Create, embed, delete or update document content

Use this task to remove existing content of a processed document, add content to this document based on a dynamically resolved path, of even more.

Mandatory settings

Key	Type	Description
Content path	Pattern	The path of the content. Leave this value empty and check 'delete content' to delete the content. This value will be resolved by Fast2 prior to the task execution. Wildcard characters are supported. Ex/ `${absolutePath}/myFile.txt` `**/myFile.txt` `${pattern}/*.txt`

Optional settings

Key	Type	Description
Check if file exists	`Boolean`	Check if the content referenced by the path is existing and accessible. If not, an exception is thrown. If a wildcard is used as content path, the new content path will be skipped if this option is disabled.

Files to exclude	`String list`	List here all patterns for files you wish to exclude. One line per match.
		Ex/
		`**/*.xml`
		`**/folder/to/exclude/`
		`*.json` to ignore all JSON files

Add content as annotation	`Boolean`	Check this option to add the content as annotation for document. If disabled, the content will be added as regular content to the document.
Delete in-place content	`Boolean`	Override existing content with the new one, or remove the content attached to the document

AlterDocumentFolder - Change document folders classification

Optional settings

Key	Type	Description	Default value
Clear existing folders	`Boolean`	Delete all existing folders	`false`
Create folders	`Boolean`	Set the permission to create new folders	`true`
Append as child to existing folder	`Boolean`	Add as a child of an already existing folder the new folder	`false`
Target folder class name	Pattern	The symbolic name of your new folder, i.e. the path	
Folder name to create	Pattern	Leave empty to not create folder	
Property map	`String/Pattern map`	Ex/ targetProperty = `${variableName}`	

AlterDocumentProperties - Alter multiple document properties

Dive into the punnet to go and modify one or more properties carried by the document

Mandatory settings

Key	Type	Description
Property map	`String/Pattern map`	List of the properties to modify. One line per property Ex/ targetProperty = documentStringGetter

AlterPunnetProperty - Create or update a punnet metadata

Dive at the punnet level to add (or udpate) one or more properties carried by the punnet itself

Mandatory settings

Key	Type	Description
Source metadata	Pattern	This field will be resolved by Fast2 prior to the task execution

Optional settings

Key	Type	Description
Target property	`String`	Name of the porperty to change. If not set or empty, Fast2 will skip the document

ApplyDroolsTask - Rules from Excel file

Apply functional and/or technical rules from an Excel file. Mainly used for mapping properties during complex migrations but can be used for simple data transformations.

Mandatory settings

Key	Type	Description
Worksheets name	`String list`	Apply the content of the mentioned sheet. Take the first sheet if not set
Excel file path	`String`	Path to the Excel worksheet Ex/ ../rules/example.xls

Optional settings

Key	Type	Description	Default value
Static cache	`Boolean`	Use a static cache for rules file. If false, the Excel file will be refreshed for each campaign	`true`
worksheet	`String`		
Worksheets character encoding	`String`	Inform Fast2 of character encoding used by the worksheet	`Cp1252`

CSVKeyValueParser - CSV parser

Parse a CSV content and put parsed values as document data

Mandatory settings

Key	Type	Description
Document Id property name	`String`	Property name where the Id is based on

Optional settings

Key	Type	Description	Default value
Regex to parse each line	`String`	The regex used for parsing each document	`^\"?([^\"]*)\"?;\"?([^\"]*)\"?$`

CSVQueryTask - CSV Mapping: fetch data from a CSV file

Fetch data from CSV using a key and and extra data from the CSV columns.

Optional settings

Key	Type	Description	Default value
Data name to report results	`String`	Name of the property where the result status is	`CsvQueryTaskStatus`

Key	Type	Description	Default value
		stored inside the punnet	
Check result unicity	`Boolean`	If it's not, populate first found result	`false`
Definition of the CSV to read	CsvDescriptor		
Skip exception	`Boolean`	Silent fail instead of throwing exceptions	`false`

ContentURLResolver - Build absolute URL for content

Simple Content URL renaming task : if target property (a list of string) is set, try to find the first path in this list which exists, converting this path to a fully-defined path including intermediary paths, corresponding documentId, extension, ... Otherwise, use Content URL included in Punnet as a wildcarded path (e.g. C:/input//.xml), and resolve to a fully-defined path.

Mandatory settings

Key	Type	Description
Source folder	`String`	Path of the parent folder, which will be the common part for all target paths in target list

Optional settings

Key	Type	Description	Default value
New mime type	`String`	New mime type to set at the end of the new content path	
Extension to use	`String`	Wildcards accepted	`*`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Mime type blacklist	`String`	Restrict action on content with this mimetype	
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	

Key	Type	Description	Default value
List of paths to resolve	`String list`	Regex wildcards accepted	
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Exception on multi-page content	`Boolean`	Ask Fast2 to throw a task exception when running into multi-page contents	`false`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
findMimeType	`Boolean`		
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`

ConvertAndSaveIS - Convert FileNet Image Services annotations

This task is used to perform the IS annotation conversion and its save into a target referential

Mandatory settings

Key	Type	Description
Annotation converter	FileNetXFDFAnnotationConverter	The FileNet XFDF module used to convert the annotations
IS annotation parser	ParseISAnnotation	Specific module to parse the Image Services annotations

Optional settings

Key	Type	Description	Default value
Status for all annotation converted	`String`	Value when all annotation have been converted	`2`
Status for some annotations	`String`	Value when some annotation	`8`

Key	Type	Description	Default value
in exception		are in exception	
Property name result	`String`	Keep the conversion result	`AnnotationConversionSt`
Skip conversion error	`Boolean`	Either Fast2 will throw an exception if an error occured during the conversion, or it will skip to the next annotation	`false`
Convert annotation to XFDF	`Boolean`	Either Fast2 converts during annotation process, or leave the annotation to its original format	`true`

Key	Type	Description	Default value
Annotation accessor	AnnotationAccessor	Choose annotation accessor to save annotations	
Document layout DPI	`Float`	Set here the layout DPI for the document	`72.0`
Status for no-conversion annotation	`String`	Value when no annotation has been converted	`9`

ConvertDateProperties - Convert multiple document/folder date properties

Easily convert date properties from one format to another.

Mandatory settings

Key	Type	Description
Properties names	`List<String>`	Names of the properties to convert. One line per property
Input date format	`String`	Original date format of the properties to convert (Ex: yyyy-MM-dd)
Output date format	`String`	Desired date format of the properties after conversion (Ex: dd/MM/yyyy)
Input locale	`String`	Locale of the input date format (Ex: en-US, fr-FR, de-DE)
Output locale	`String`	Locale of the output date format (Ex: en-GB, es-ES, pt-BR)

DeleteContent - Delete local content

Delete the content of your document within your file system. It will retrieve the files targeted by the URL of all the document contents in your punnet

Optional settings

Key	Type	Description	Default value
Delay between two deletion attempt	`Integer`	Delay in milliseconds	`1000`
File path to delete	`String`	Used as prefix to select files to delete. This field can be see as a whitelist of the contents to delete. If empty, all accessible contents will be deleted from the machine.	
Exception when deletion failed	`Boolean`	Thrown an exception if the file has been found but the delete operation failed.	`true`
Maximum number of tries for deletion	`Integer`	Thrown an exception if the file has not been deleted after this number of tries	`10`
Delete content entry	`Boolean`	Erase the URL entry from the document, in the punnet.	`true`
Exception when file does not exist	`Boolean`	Throw an exception if file does not exist. Otherwise, silent fail	`true`

EmbeddedDbQuery - Query the embedded OpenSearch database

Provide a campaign name, task name, and unique ID field to find a single punnet in the embedded OpenSearch database.

Mandatory settings

Key	Type	Description
Embedded Db connection provider	EmbeddedDbConnectionProvider	
Campaign prefix	`String`	Generally Map name, but ca changed, so the part that co _Run#
Fields for OpenSearch query	`String/Pattern map`	Fields used to construct Ope query. Recommended to inc the following 3: campaign.k stepName.keyword, punnet.documents.docume (or similar unique ID). The c only return one punnet.
Data to enrich current punnet	`String/Pattern map`	List of data coming from the OpenSearch fetched punne the current punnet. Give a l such as edb-hash and an ex fill the value like `${hash}` o

Key	Type	Description
		`${property('file:content` for properties with colons ir

FileNet35ExtraSearchTask - File Net search

Find your documents inside your File Net 3.5 instance.

Optional settings

Key	Type	Description
	FileNet35ExtraSearchTaskSettings	

JSTransform - JavaScript evaluation task

This task serves the purpose of modifying a punnet based on instructions embedded into a JavaScript snippet. The latter can be either directly pasted as task parameter, or read from a given path. If both these ways are used, the task will only consider the script pasted in.

INFO

The core engine responsible for executing JavaScript transformations has been upgraded from Fast2 2025.4.1, replacing the legacy **Nashorn** engine with the modern and high-performance **GraalVM JS**. The new engine is configured to be **fully backward-compatible**. All existing JavaScript transformation scripts written for Nashorn will continue to function seamlessly without requiring any modifications.

Mandatory settings

Key	Type	Description
Script or script file path	`String`	Javascript source or path to JavaScript file with preloaded script. Ex: C:/fast2/script.js OR punnet.getDataSet().addData("data","String", "value");

Optional settings

Key	Type	Description	Default value
(Deprecated) Script file path	`String`	Path to JavaScript file with preloaded script. Deprecated: use script property instead.	
Script engine	`String`	Script engine to use for the JS execution. Use 'nashorn' for JDK8. Other options are available, such as js, javascript, ecmascript.	`nashorn`

MailDeleter - Remove mails

This class allow you to connect to your mail box and select mails to delete. You can search among your mails by term or properties

Optional settings

Key	Type	Description	Default value
Mail connection provider	MailBoxProvider		
Maximum connection ttl	`Long`	Time in milliseconds	`60`
Search term type	`String`		`Message-Id`
Pattern to evaluate property	`String`		`${Message-Id}`
Exception when mail not found	`Boolean`	Throw an exception when the mail is not found. Otherwise, silent fail	`true`
Save message changes	`Boolean`		`true`

RenameDocumentProperties - Rename multiple document properties

Dive into the punnet to rename one or more properties carried by the document.

Mandatory settings

Key	Type	Description
Property map	`String/String map`	List of the properties to rename. One line per property Ex/ targetProperty = documentStringGetter

Optional settings

Key	Type	Description	Default value
Exception when property rename conflict	`Boolean`	Throws an exception if the target property exists, otherwise fails silently	`true`

UpdateSharedObject - Update a shared object value from its name

Use this task to change a system-wide configuration setting at runtime. Mainly used for dynamic campaign variables built as a shared object in Fast2. Be careful: only works with shared object set with Global scope

Mandatory settings

Key	Type	Description
Shared object new value	`String`	Can be almost any type of variable (String, int...) except object and tasks of the Fast2 catalog. You can also access to any custom document data value from the pattern using `${variableName}`. Already known variables : base, campaign, taskFlowMap, step, punnetTraceId, punnetId and documentId.
Shared object name to update	`String`	Name of the object to update. The shared object will be automatically created if does not already exists.

Optional settings

Key	Type	Description	Default value
Number of executions	`Integer`	Means the number of executions per campaign	`1`

XSLTransform - Apply a XSL transformation on XML Punnets

With pretty much straight-forward task you can fine-tune any punnet or document metadata, or even the content targetted by the migration. Build your custom XSL file, and ask Fast2 to apply the changes onto the migration-related data

Mandatory settings

Key	Type	Description
XSL file path	`String`	Specify here the absolute path of the XSL file, as well as the name and the extension of the file. This file can be located on a separate machine

Optional settings

Key	Type	Description
XSL Stylesheet content	`String`	Enter here your xsl:stylesheet content

Catalog / Helper

DctmConfiguration - Module for customized Documentum configuration

Optional settings

Key	Type	Description	Default value
Document ACL	`String`	Default ACL to set for the document. Can be overrided by adding a value 'acl_name' as document metadata.	
Annotations user	`String`	Default user in charge of retrieving the annotations. If not set, an error will be thrown.	`dmadmin`
Retrieve mime-type	`Boolean`	Ask Fast2 to get the	`true`

Key	Type	Description	Default value
from content		document mime-type from the content encoding, instead of using 'DfClientX' provided by the Documentum client.	
Annotation ACL	`String`	This ACL should have write access.	`ar_company_wide`
Date format	`String`	Date format which the documents will have to match in order to properly be loaded into Documentum.	`dd/MM/yyyy`
Annotation path	`String`	Path to the folder where the	`/System/Applications/ARender/A`

Key	Type	Description	Default value
		annotations will be retrieved by Fast2. If not set, an error will be thrown.	

WcmApiConfigSettings - URL configuration

This class allows to configure several elements associated with a URL.

Optional settings

Key	Type	Description
Download URL	`String`	
Upload URL	`String`	
Credentials portection for user token	`String`	
Remote URL	`String`	
Credentials protection	`String`	

Catalog / Tool

AlfrescoRestDeleteNode - Alfresco delete nodes using Alfresco REST protocol

This task relies on the Alfresco public REST API (with v1.0.4 of the Alfresco REST client) to delete nodes.

Mandatory settings

Key	Type	Description
Alfresco connection provider	AlfrescoRESTConnectionProvider	

Optional settings

Key	Type	Description	Default value
Permanent deletion of the node	`Boolean`	If true then the node is deleted permanently, without moving to the trashcan. Only the owner of the node or an admin can permanently delete the node.	`true`
The pattern to get the node Id	Pattern		

Key	Type	Description	Default value
from the document			

AwsMove - AWS S3 file mover

Reorganize your files inside your AWS S3 environment.

Mandatory settings

Key	Type	Description
AWS access credentials	AWSConnectionProvider	Must have granted AmazonS3FullAccess permission
Target key	`String`	The destination path inside your bucket where the document must be placed. Use as standard Pattern (includes S3 Folders)

Optional settings

Key	Type	Description
Target bucket	`String`	The target bucket where you want to move your S3 files. If empty use the same as origin

CheckCompoundDocumentSettings - Check if an Office document contains embedded files

Only Office documents are supported (docx, xlsx, pptx). All other contents will be skipped.

Optional settings

Key	Type	Description	Default value
Data name for found items	`String`	The name of the new data under which the list of found items will be stored	`EMBEDDED_FILES`
Extract as side content	`Boolean`		

CountPdfPages - Count the number of pages in PDF file

This task will add the number of pages as a metadata to the document.

Optional settings

Key	Type	Description	Default value
Throw conversion	`Boolean`	If Fast2 performs document conversion, it can either fail	`true`

Key	Type	Description	Default value
exceptions		silently or pop an error when the action has not been properly completed	
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Property name	`String`	Name of the property to which the number of pages will be	`F_PAGES`

Key	Type	Description	Default value
		linked	

DeduplicatePunnets - De-duplicate tasks based on some pattern

This task is used to get rid of duplicate punnets

Mandatory settings

Key	Type	Description
Save values to this file	Pattern	
The pattern to use to get the unique value	Pattern	

Optional settings

Key	Type	Description	Default value
Iterate over punnets (default: false) or over documents (true)	`Boolean`		`false`

Key	Type	Description	Default value
The pattern to use to explain which element this is duplicate with	Pattern		
Size of per-file hash table	`Integer`	Each file has a table attached to access the existing elements faster (the larger the table, the faster the search). Each individual file will store n elements (size of file / size of each elements), the recommended value for hash table should not be less than 10% of the number elements per file, which means each hash table entry references ~10 elemens. Example : storing 50 bytes of identifications (from the Identification pattern), each element will be ~100 bytes (including key, ..), so a 64MBytes file will store ~671088 elements per file. The hash table being 64k, each hash table entry references ~10 elements	`65536`
Size of each individual file,	`Integer`	Multiple files will be created on-demand to store all elements as	`64`

Key	Type	Description	Default value
in MBytes		required.	
In which property we put the identification of the first element we are duplicate with	`String`		

EndTaskWriter - Create file with custom content when map ends

A task to write a file when all punnets of task are finished.

Mandatory settings

Key	Type	Description
Output file path	`String`	Absolute path to file. This path must include file name and extension. This field will be resolved by Fast2 before the task is run

Optional settings

Key	Type	Description	Default value
File encoding	`String`	Encoding of the file where the content will be written	`UTF-8`
File content	`String`	Text to write in the output file	

ExceptionGenerator - Regularly generate exceptions

This task will generates different exception types : either TaskException or RuntimeException. It will be usefull for your when dealing with exception routing.By default, Fast2 will produce 3 task exceptions, then 4 runtime exceptions, and finally 4 no-exception punnets. To force exceptions, set the no-exceptions ratio to zero.

Optional settings

Key	Type	Description	Default value
Runtime exceptions ratio	`Integer`	The number of runtime exception which will be thrown by Fast2	`3`
No-exception ratio	`Integer`	The number of no-exception which will be thrown by Fast2	`4`

Key	Type	Description	Default value
Task exceptions ratio	`Integer`	The number of task exception which will be thrown by Fast2	`3`

FlowerDocsQuerier - FlowerDocs querier

Allows components extraction from FlowerDocs using JSON formatted FlowerDocs request. Components can be documents, folders, virtual folders or tasks.

Mandatory settings

Key	Type	Description	Default value
FlowerDocs connection provider	FlowerDocsConnectionProvider		
JSON Flower Search Request	`String`	This field supports pattern `${...}` syntax.	

Key	Type	Description	Default value
Flower component category	`String`	Choose among DOCUMENT, TASK, FOLDER or VIRTUAL_FOLDER. This field supports pattern `${...}` syntax.	

GenerateExceptionTask - Throw exception when condition is verified

This task is responsible for exception generation based on a condition which can be dynamically built for each punnet.

Optional settings

Key	Type	Description	Default value
Condition	`String`	Set here the condition to trigger exception. This field will be resolved by Fast2 before the task is executed Ex/ mimeType == application/pdf	`true`

GenericRestApiTask - Consume a REST API

Consume any generic REST API task and add the response in the punnet. Supported Methods: GET, PUT, POST and DELETE. Configuration: Allows defining the API endpoint URL, headers, and query parameters. Response Handling: The HTTP status code (e.g., 200) and the response body (e.g., JSON) are automatically mapped to punnet metadata fields, with configurable key names. Error Handling: In case of a failed API call, the error code and response body (if available) are still added to the punnet to allow for diagnostics.

Mandatory settings

Key	Type	Description	Default value
Response code metadata key	`Pattern`	Property name of the response code in the dataset	
Body metadata key	`Pattern`	Property name of the body in the dataset	
REST API client	GenericRestClient		
Continue task after document failed	`Boolean`	Continue the task process even if an API call failed with a document	`true`

HashSignTask - Compute content hash

This task computes the hash of a given document content. This new hash can be confronted to an already existing one.

Optional settings

Key	Type	Description	Default value
Suffix of ouptut file	`String`	Suffix of the external file containing the hash value to compare with	
Conten key for hash	`String`	Name of the metadata where the hash value will be stored. This value will be attached to the content and the document dataset itself	`hash`
Algorithm	`String`	The algorithm of hashing which will be used for document content	`SHA-256`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`

Key	Type	Description	Default value
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Block size	`Integer`	In bytes. The default value is 256 * 1024	`262144`
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Existing hash to compare	`String`	Document data name to compare the new hash with. Throws TaskException when different	
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`

Key	Type	Description	Default value
Document key for hash	`String`	Name of the metadata where the hash value will be stored, at document level	

MailMover - Move email conversation into folder

This task will be useful when your needs will be to move a given email conversation into a dedicated folder. Whether this folder exists or not, Fast2 will be able to retrieve or create it.

Mandatory settings

Key	Type	Description
Email provider	MailBoxProvider	The Fast2 module establishing the connection to the email server, from the account of a given user. For more about the configuration of the object, please refer to the appropriate section
Destination folder	`String`	The folder where the email will eventually be moved to. This value will be resolved by Fast2 prior to the task execution

Optional settings

Key	Type	Description	Default value
mailNotFoundException	`Boolean`		
Maximum connection TTL	`Long`	Fill the value in milliseconds	`60`
Data to find	`String`	The data to look for. This value will be resolved by Fast2 prior to the task execution	`${Message-Id}`
Create destination folder	`Boolean`	Ask Fast2 to create the destination folder to move the email into, in case this specific folder does not exist yet	`false`
Search field name	`String`	The name of the field where to find the data. Only 'Subject' and 'Message-Id' are available	`Message-Id`

MimeTypeFinder - Find mime-type of documents

This task is used for automatic detection of mime type for documents

Optional settings

Key	Type	Description	Default value
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`
Stop at first exception	`Boolean`	Stop processing punnets when one could not been properly processed	`false`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Use most specific mime type	`Boolean`	Otherwise use the first mime type found	`false`
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the	`true`

Key	Type	Description	Default value
		action has not been properly completed	
Default mime type	`String`	The default mime-type to set if none has been found. This value must be set, or it will throw a RuntimeException	`application/octet-stream`
Mime-type : Check document before content	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document mime-type. By default, Fast2 will check at content level	`false`
Update document mime type	`Boolean`	Otherwise update only page or content mimetype	`true`

Key	Type	Description	Default value
Force to identify mime type	`Boolean`	If the mime type could not be found by looking at the metadata, either Fast2 skips the document or digs deeper into the content to retrieve the mime type	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`

MimetypeToExtension - Append extension to name

Based on the mime-type of the content, this task will resolve the correct extension to append to the name of the document. Only supported for one content per document

Optional settings

Key	Type	Description	Default value
Document mime-type	`String`	This value will be resolved by Fast2, `${...}` syntax is supported. Use this option when only the document	

Key	Type	Description	Default value
		mime-type has been provided, without the actual content.	
Key of name property	`String`	Key of the current name metadata, whose value will be appended by the matching extension.	`name`

MoveAnnotationContent - Move the content of any annotation

This task is responsible for moving content of annotations from a given folder into a new one.

Mandatory settings

Key	Type	Description
Source folder	`String`	Absolute path of the folder where to find the annotations to move
Destination folder	`String`	Absolute path of the folder where to move the annotations

MoveContent - Move or copy the content of a document

This task is responsible for moving content of documents from a given folder into a new one.

Mandatory settings

Key	Type	Description
Destination folder	`String`	The path of the folder where the contents will be moved into

Optional settings

Key	Type	Description	Default value
Copy files	`Boolean`	Copy the file to the destination folder instead of moving it	`false`
Delete original copy	`Boolean`	Delete the file in the source folder when it has been migrated	`false`
Process annotation contents	`Boolean`	If annotations are asked to be migrated, you can filter here to process their content(s) or only their metadata	`false`
Checking interval	`Integer`	Time to wait between two checks if target file exists	`1000`
Scan recursive content	`Boolean`	Only convert terminal contents and not container ones	`false`

Key	Type	Description	Default value
Exclude file name for renaming file	`Boolean`	Renamed the file based on the value of the destination folder only	`false`
Wait until target file exists	`Boolean`	Only process next document when the current has successfully been migrated	`false`
Supported mime-types	`String list`	Specify the list of all mime-types of documents which Fast2 will convert	
Allowed retries	`Integer`	Number of checks before despair and exception	`60`
Source folder	`String`	Set here the common prefix of all document contents to move. If null use file folder	
Throw conversion exceptions	`Boolean`	If Fast2 performs document conversion, it can either fail silently or pop an error when the action has not been properly completed	`true`
Mime-type : Check document	`Boolean`	You can assume the file extension is accurate, or ask Fast2 to check the content encoding to identify more precisely the document	`false`

Key	Type	Description	Default value
before content		mime-type. By default, Fast2 will check at content level	
File extension	`String`	The extension to append to the name of the files once they'll be moved	
Keep original filename	`Boolean`	Set the destination file name to the 'title' property defined at the content level. Otherwise, keep the name of the file pointed by the URL	`false`
Process all contents	`Boolean`	Fast2 will either only focus on the first encountered content, or process them all	`true`
Files to exclude	`String list`	The path of the folder to exclude. Its whole content will remain in place. Leave empty to move all folders children Ex/ *.out, folder/**/exclude	

MovePunnet - Move a punnet from folder to folder

This task is responsible for moving a punnet from an embedded path into a new folder.

Mandatory settings

Key	Type	Description
Destination path	`String`	The path where to move the punnets. This field will be resolved by Fast2 prior to the task execution

Optional settings

Key	Type	Description	Default value
Time between two retries	`Integer`	In milliseconds	`1000`
Allowed retries	`Integer`	Maximum number of retries before throwing an exception when renaming failed	`10`
Override existing punnet	`Boolean`	If Fast2 finds an already existing punnet, it will override it with the one being processed	`true`
Look for path at punnet level	`Boolean`	Tells Fast2 to look for the absolute path variable (whose key is 'absolutePath') into the punnet dataset. Otherwise Fast2 will look at the first document dataset level	`true`

Noop - Blank task performing no operation

This task does not perform anything, hence you don't have to configure it. All documents and punnets will go through it without having their state updated.

NuxeoQuery - Query nuxeo from NXQL

This task only works with JDK-11. If any record matches the input query, the UUID of the Nuxeo items will be added to the F2 document as a new dataset.

Mandatory settings

Key	Type	Description
Nuxeo connection details	`NuxeoConnectionProvider`	
Nuxeo query	`String`	NXQL query, with double-quotes around values (Ex: `SELECT * FROM Document WHERE dc:title = \"${nom}\" AND ecm:isTrashed = 0`)

ParseJsonAsProperties - Task to add new data from a JSON file as data

Mandatory settings

Key	Type	Description	Example	Default value
Path of JSON file to parse	Pattern	`${absolutePath}/myFile.json`	Path of JSON file for properties to map. Pattern syntax `${...}` is supported, wilcards are not.	

PropertyHelper - FileNet submodule for properties management

Optional settings

Key	Type	Description	Default value
Blacklist for extraction	`String list`	Default blacklist is: `ActiveMarkings`, `AuditedEvents`, `Annotations`, `ChildDocuments`, `ChildRelationships`, `CmHoldRelationships`, `CmThumbnails`, `Containers`, `CoordinatedTasks`, `CurrentVersion`,	

Key	Type	Description	Default value
		`DependentDocuments`, `DestinationDocuments`, `ExternalReplicaIdentities`, `ParentDocuments`, `ParentRelationships`, `Permissions`, `ReleasedVersion`, `StorageArea`, `StoragePolicy`, `This`, `Versions`, `WorkflowSubscriptions`	
Force user names	`Boolean`	Force assigning users (e.g. Creator, LastModifier) when they don't exist in the destination environment	`true`
Date format	`String`		`MM dd HH:mm:ss z yyyy`
Property name used to explicitly skip Data	`String`		
Do not throw Date parsing exceptions	`Boolean`		

Key	Type	Description	Default value
Store object-store as name	`Boolean`	By default, Fast2 is expecting FileNet UUID as object-store reference on object-typed properties. Enable this option to deal with the object-store name instead of its UUID. This parameter is only use at extraction.	

PunnetSerializer - From-java-to-XML punnet converter

This task is responsible for serializing a punnet to an XML file. That can be interesting to check punnet metadata or freeze a punnet at a certain state.

Optional settings

Key	Type	Description	Default value
Serialize punnets as JSON	`Boolean`	If enabled, punnet will be serialized as Json file. Otherwise, it will be a XML file.	`false`

PunnetWriteId - List all punnet IDs into a file

This task is responsible for writing all punnet IDs into a given text file. Whether the punnet has documents or not, you can keep a trace of all created and encountered punnets.

Mandatory settings

Key	Type	Description
Output file	`String`	The absolute path of the output file where to store all punnet IDs. Specify file name and extension

Optional settings

Key	Type	Description	Default value
	OutputStream		
Append	`Boolean`	Create FileOutputStream with this append option	
Always write punnet ID	`Boolean`	Write punnet Id event when it contains no document	`true`

ReadContent - Resolve mime type from content

This task is responsible to find the mime type of a document accross either its metadata or its content.

Mandatory settings

Key	Type	Description
Mime type retriever	MimeTypeFinder	Module to find content mime type

Optional settings

Key	Type	Description	Default value
Dry run	`Boolean`	Process all punnets without editing their state or metadata	`false`
Maximum number of page read per content	`Integer`	Only for multi-page content	`Integer.MAX_VALUE`
Force to identify mime type	`Boolean`	Ask Fast2 to dig deeper into the content to find a mime type. The metadata will be added to the content	`false`

SQLMultiQueryTask - Perform SQL statements between database tables and documents in Fast2

Perform SQL INSERT or UPDATE statements to documents in database, or SELECT from data existing in the database to attach them onto the document dataset.

Mandatory settings

Key	Type	Description
Query caller	SQLQueryColumnCaller	This modules is responsible of establishing the connection between Fast2 and the designated database
Source attributes	`String/Pattern map`	Key: Desired column for where clause; Value: Source Document data to use for query

Optional settings

Key	Type	Description	Default value
Use PreparedStatements	`Boolean`	Use PreparedStatements instead of plain SQL statements	
Target attributes	`String/Pattern map`	<key, value> set where 'key' refers to the SQL name of the data, and 'value' refers the name of the data in the Fast2 dataset.	

Key	Type	Description	Default value
Reset target data	`Boolean`	Clean content when target already exists	`true`
SQL column types	`String/String map`	<key, value> where 'key' is the SQL data name, and 'value' is its type. Supported types are : String, float, int, Date.	
SQL query	`String`	Select precisely data you want to extract through a classic SQL query. All retieved values will be attached to the document dataset based on the data listed in the 'Target attributes' configuration section.	

SingleCallTask - Call a task only once per campaign

This task will be useful to perform a given subtask only once in a map execution. You choose to call this subtask at the very begining of the campaign, or at its very end.

Mandatory settings

Key	Type	Description
Subtask	Task	The task to call only once in the campaign execution

Optional settings

Key	Type	Description	Default value
Call at end	`Boolean`	Execute the subtask when the first punnet is processed	`false`
Call at begining	`Boolean`	Execute the subtask when the first punnet is processed	`false`

SleepTask - Blocks punnet on thread for a given period of time

Task blocking a thread per punnet to wait some time before processing the punnet, without updating its state or metadata.

> **WARNING**
>
> As the thread is asleep for a defined time, all tasks are slowed down.

Optional settings

Key	Type	Description	Default value
Sleep time	`Integer`	In milliseconds	`500`

Catalog / Injectors

AlfrescoBulkImporter - Perform a bulk import on Alfresco

Load documents and metadata into Alfresco without changing the current tree structure of those same documents. The good performances of such injection are restrained with the complexity of the tree-view setup

Mandatory settings

Key	Type	Description
Source directory	`String`	Path to the folder to migrate into Alfresco
Target path	`String`	Path where the folder will be stored into Alfresco
Alfresco connection provider	AlfrescoConnectionProvider	This modules is responsible of the two-way communication between Fast2 and the designated Alfresco instance.
Target NodeRef	`String`	NodeRef of the parent where the folder will be stored into Alfresco

Optional settings

Key	Type	Description	Default value
Clean destination	`Boolean`	Replace all existing content or metadata present in the destination folder before injection	`false`
Disable rules	`Boolean`	Disable rules for injection to prevent Alfresco to run checks on each document	`false`
Add metadata	`Boolean`	Load document content and its metadata. All metadata might not me compatible with Alfresco standards	`false`
Number of threads	`Integer`	Number of threads to allocate for the bulk import	`1`
Timeout	`Integer`	Time to wait before closing the session. Of not set, the value will be set to 300'000	
Copy files	`Boolean`	Leave all documents in the source folder	`false`
Batch size	`Integer`	Size of the batch to build for upload	`0`

AlfrescoInjector - Injection into Alfresco ECM using CMIS protocol

This task can be used to inject documents into Alfresco, using the CMIS protocol on top of HTTP. We rely on v1.0 of the opencmis module made available by Alfresco.

Mandatory settings

Key	Type	Description
Alfresco connection provider	AlfrescoCMISConnectionProvider	This modules is responsible of the two-way communication between Fast2 and the designated Alfresco instance

Optional settings

Key	Type	Description	Default value
Alfresco ID key	`String`	Document metadata key with the Alfresco ID of the injected document.	`alfDocumentId`
Property Helper	PropertyHelper		
Properties regex	`String`	Regex pattern to filter the properties to inject	`(cmis:.*)`

Key	Type	Description	Default value
		with the document.	
Hash content column name	`String`	Hash content column name to version a document only when the content is different (but same index)	
Destination folder	`String`	Folder where the documents will be loaded into	
SQL update query	`String`	CMIS SQL update query to select the document to update.	
Overwrite 'can create'	`Boolean`	Ask Fast2 to create destination folder(s) if they do not already exist	`true`
Hash index column name	`String`	Hash index column name to version a document only when the content is different (but same index)	
Force update	`Boolean`	Throw an error if the document did not exist prior to the loading call	

Key	Type	Description	Default value
Prevent duplicate	`Boolean`	Fast2 will throw an error if the document has already been injected	

AlfrescoRestInjector - Alfresco injector using Alfresco REST protocol

This task relies on the Alfresco public REST API (with v1.0.4 of the Alfresco REST client) to load documents and metadata into a given Alfresco instance.

To force the type of resource to create in the destination system, use the `nodeType` data into the document dataset. Default value is `cm:content`.

Mandatory settings

Key	Type	Description
Alfresco connection provider	AlfrescoRESTConnectionProvider	

Optional settings

Key	Type	Description	Default value
Root folder name to inject in a	`String`	'-my-', '-shared-', '-root-' are equivalent	`-my-`

Key	Type	Description	Default value
specific repository			
Alfresco destination path	`String`	The path of the folder where the documents will be saved in Alfresco. This field supports patterns (based on punnet, document and campaign metadata). If this path starts with Alfresco nodeRef prefix 'workspace://SpacesStore/', the document will be injected into the corresponding folder. However, such folder needs to be created beforehand.	
Regex pattern filter for document properties	`String`		`(cm:.*)`
Safe update	`Boolean`	If the document does not already exist, the first version will create the document. Later versions will be incremented on top of the existing version based on the data 'cm:versionLabel' property.	

Key	Type	Description	Default value
Auto rename feature	`Boolean`	Triggers the Alfresco auto-rename feature, to prevent Alfresco to throw a 'duplicate document' error.	`false`
Pivot metadata for multiversion	Pattern	If all documents of the punnet have the same value for this metadata, they will be considered as being the different versions of a same document in Alfresco.	
Overwrite documents when they already exist	`Boolean`	Triggers the Alfresco overwrite feature, where the incoming document will replace an existing document having the same key.	`false`
Alfresco ID for update	`String`	Specify here the Alfresco UUID of the document to update. The value will be resolved by Fast2, syntax `${...}` is supported. This value can start with 'workspace://SpacesStore/' Ex/ `${property('alfcmis:nodeRef')}`	

AwsInjector - Injector into AWS S3 buckets

Fast2 proposes this task to load your documents, metadata and more within designated S3 buckets. Both client- and server-side encryption are supported (v1.6 of AWS encyption SDK). This loader relies on v1.11.848 of AWS Java SDK. The uploaded file will be title according to the `name` metadata of the processed document.

Mandatory settings

Key	Type	Description	Default value
Destination bucket	`String`	The name of the bucket where the documents will be migrated to.	`fast2-default`
AWS credentials	AWSConnectionProvider	Must have granted AmazonS3FullAccess permission	

Optional settings

Key	Type	Description	Defa val
Encryption key	`String`	Key used for server-side encryption. Ex/ 01234567-abcd-efgh-ijkl-0123456789ab	
Destination folder	`String`	The parent folder of the documents to inject. This field supports pattern (using punnet, document or campaign metadata).	

Key	Type	Description	Defa val
		Leave empty for storing at the root of the bucket.	
Dry run	`Boolean`	Simulates an injection, performs document integrity controls, but does not load the document into AWS S3	`fals`
Destination file name	`String`	Metadata for the file name once injected into the S3 bucket. Pattern syntax is supported.	`${na`
Encryption context	`String`	Context used for server-side encryption. This context is a JSON map. Ex/ `{\"testKey\":\"testValue\"}`	
ARN key	`String`	Key used for client-side encryption, before loading the document into S3. Ex/ arn:aws:kms::111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab	
Update only	`Boolean`	Only changing metadata, content is left as is	`fals`

CSVWriter - CSV file writer

Use this task to write punnet and document related data into a CSV. You can specify the name of such file as well as the path where your want Fast2 to create and populate it.

Mandatory settings

Key	Type	Description
Path for (local) output CSV files	`String`	Using a pattern for this field will trigger a resolution by Fast2 at runtime

Optional settings

Key	Type	Description	Default value
Close output file at each line	`Boolean`	Close CSV file after each punnet processed. This option can come useful to prevent too many files opened errors when each punnet created a dedicated CSV file.	
New column headers	`String list`	The new column names to use for the document metadata. If no set, the column headers will be populated from the names of the document metadata. By default, the data 'punnetId' and 'documentId' will be appended to the existing data.	

Key	Type	Description	Default value
Upload punnet data	`Boolean`	For each document add the punnet data to wich it was attached to	`true`
CSV separator	`String`	The separator used between columns in the resulting CSV file.	`,`
Write all in the same CSV file	`Boolean`	Merge metadata of all punnets in a single output CSV file. The missing columns headers will be added on the fly, although it is wiser to list them all in the 'New column headers' field. If set to false, any existing CSV file with the same name will be overwritten.	`true`
Protect with double quote	`Boolean`	This option will surround every value with double quotes. Such use will be mostly relevant when dealing with multivalued metadata.	`true`
Add Folder Metadata	`Boolean`	For each document add its folder metadata on the same CSV line. Only the first level of folders will be stored into the CSV file. If a document is attached to more than one folder, its line will be cloned in the CSV file, to display one folder per line.	`false`

NOTE

CSVWriter folders behavior and limitations (v2025.5.0):

- Single level of depth: only the direct parent folders of documents are considered.
- Single combination per CSV line: one row per unique document/folder combination; documents without a folder appear with blank folder columns.

DctmInjector - Injection into Documentum

Use this task to inject into Documentum ECM system. Fast2 embeds v6.7 of Documentum modules to take the most out of of this injection phase.

Mandatory settings

Key	Type	Description
Credentials	DctmConnectionProvider	Connection module establishing the communication with a given Documentum instance.

Optional settings

Key	Type	Description
Documentum configuration	DctmConfiguration	Customize here the Documentum details related to the instance you are planning to inject documents

Key	Type	Description
		into. For more, refer to the appropriate section.
SSH client	DctmSshClient	SSH client used to establish the connection with the Documentum server

FileNet35Injector - Injector for FileNet P8 3.5

Use this task to inject documents and data into a FileNet P8 3.5

Mandatory settings

Key	Type	Description
The FileNet connection provider	FileNet35ConnectionProvider	Module to establish the connection with the destination FileNet infrastructure
ObjectStore name	`String`	Name of the destination object-store where the documents and metadata will be injected

Optional settings

Key	Type	Description	Default va
Try using 'DocumentTitle' property	`Boolean`	When filing a document in a folder, try to use the FileNet DocumentTitle property for the RelationShip name	
Process multi-pages content as multi-content	`Boolean`	Treat the content of multi-pages document as multi-content document in FileNet.	
Dry run	`Boolean`	Needs to set 'UpdatingBatch on documents' to `true`	
WHERE clause for folder case	`String`	where clause to define for case research	
Restrain search results to documents	`Boolean`	Force Document search to limit to the class name provided on the document	
Only process 1st content	`Boolean`	Relevant for multi-content documents	
Restrain search results to folders	`Boolean`	Force folder search to limit to the class name provided on the folder	
Skip content injection	`Boolean`	Skip document content injection, only load the	

Key	Type	Description	Default va
		metadata and/or annotations of the processed document	
Skip document unfiling	`Boolean`	Force to skip Document unfiling to existing folders before linking it to the provided folders ; new linkages will be added to the existing ones	
Clear in-place annotations	`Boolean`	If source document contains annotations, cleanup existing ones in P8	
Restrain search results to case	`Boolean`	Force case search to limit to the class name provided on the case	
Synchronous folder creation	`Boolean`	Enforce synchronous folder creation, to make them more thread-safe	`true`
Metadata carrying document UUID	`String`	Leave empty to disable updating	`fileNetDocum`
Variable name of annotation ID	`String`	Variable name of annotation id used to replace it by generated FileNet annotation id	`${annotation`

Key	Type	Description	Default va
Force deletion	`Boolean`	If no matching document can be found, an error is thrown	
WHERE clause for folder research	`String`	Fast2 will update all folders matching the following WHERE statement	
Associate annotation FileNet ID to its content	`Boolean`	Update annotation content with its generated FileNet id according to annotation id variable	
Delete in-place version	`Boolean`	Delete the last document version after checkin a new one	
Update system properties	`Boolean`	It can only be used for either document creation or update (when a new version is created)	
Default MimeType	`String`	Mime-type to set when none has been found	
Fields to update	`String`	Default query to select fields to update	`*`
Post-commit delta	`Integer`	Add a post-commit time, may be usefull to let FileNet	`0`

Key	Type	Description	Default va
		perform asynchronous handling of document injection	
Limit CE connection life-time	`Long`	Limit CE Session life-time : at end of TTL, the Session will be replaced by a brand new one	`Long.MAX_VAL`
Force to perform update	`Boolean`	In case the document did not exist, an error is thrown	
WHERE clause for update	`String`	Fast2 will update all documents matching the following WHERE statement Ex/ [Id]=`${myFileNetDocumentId}`)	

FileNetInjector - Injector for FileNet

Use this task to inject documents and data into a FileNet. If all documents have the same UUID for the VersionSeries data provided in the task configuration, then the FileNet versionable object will be created based on the 'MajorVersionNumber' and 'MinorVersionNumber' properties. 'Custom objects' and 'Referential Containment Relationship' are supported. If a document is injected with a folder having a FileNet compatible ID, then the folder will be created with this UUID, if not existing already.

Mandatory settings

Key	Type	Description
The FileNet connection provider	FileNetConnectionProvider	The module establishing the connection to the remote FileNet instance. For more configuration about this object, refer to appropriate section.
ObjectStore name	`String`	Name of the destination object-store where the documents and metadata will be injected

Optional settings

Key	Type	Description	Defa
Property Helper	PropertyHelper		
Try using 'DocumentTitle' property	`Boolean`	When filing a document into a folder, try to use the FileNet 'DocumentTitle' for the RelationShip name	
Process multi-pages content as multi-content	`Boolean`	Treat the content of multi-pages document as multi-content document in FileNet	
Keep original VersionSeries	`Boolean`	If true, Fast2 will create the multiversion documents with	

Key	Type	Description	Defa
ID		the VersionSeries ID specified in the 'VersionSeries metadata' configuration field.	
Dry run	`Boolean`	Do not perform anything, just prepare UpdatingBatch and drop when finished. It implies to activate 'Use UpdatingBatch on documents'.	
Restrain search results to documents	`Boolean`	Force Document search to limit to the class name provided on the document	
Metadata carrying parent folder UUID	`String`	Name of the metadata where Fast2 will store the UUID of the parent folder of the document injected into FileNet P8. Leave empty to disable updating	
Only process 1st content	`Boolean`	When a document has multiple contents, its forces to process the first one only. The others are then skipped	
Inject FileNet security	`Boolean`	Document dataset must have data 'security'. Syntax must be an array of concatenated	

Key	Type	Description	Defa
		Strings as so : `gType=<String>/gName=<String>/mask=<Integer>/depth=<Integer>/aType=<String>` where `<Integer>` and `<String>` values are replaced by the corresponding business values.	
Accept unset properties	`Boolean`	Allow registration of blank metadata in FileNet	
Skip content injection	`Boolean`	Skip document content injection, only load the metadata and/or annotations of the processed document	
Name of ID property	`String`	Name of the document property, found in the document dataset, which will be used to force Id at document creation. Leave blank to disable this feature	
Throw exception if document already exists	`Boolean`	An exception is thrown in case an older document has been found. To properly use this options, 'Prevent document	

Key	Type	Description	Defa
		overwriting' requires to be `true`	
Skip document unfiling	`Boolean`	Force to skip Document unfiling to existing folders before linking it to the provided folders. New linkages will be added to the existing ones	
Auto-classiify at checking	`Boolean`	Enable the FileNet Auto-Classify feature when the document is at checking stage	
Clear in-place annotations	`Boolean`	If source document contains annotations, clean up existing ones in P8	
Synchronous folder creation	`Boolean`	Enforce synchronous folder creation, to make them more thread-safe	`true`
Safe update of document	`Boolean`	Try updating a document. If no older version of the document can be found, create it	
Metadata carrying document UUID	`String`	Name of the metadata where Fast2 will store the UUID of the document injected into	`fileNet`

Key	Type	Description	Defa
		FileNet P8. Leave empty to disable updating	
Use UpdatingBatch of folders	`Boolean`	Use FileNet UpdatingBatch also for folders creation, which may not be thread-safe	
Variable name of annotation ID	`String`	Variable name of annotation id used to replace it by generated FileNet annotation id	`${annot`
Force deletion	`Boolean`	Force document delete action. If no matching document can be found, an error is thrown	
Associate annotation FileNet ID to its content	`Boolean`	Update annotation content with its generated FileNet id according to annotation id variable	
Delete in-place version	`Boolean`	Delete the last document version after checkin a new one	
Update system properties	`Boolean`	It can only be used for either document creation or update (when a new version is created)	

Key	Type	Description	Defa
Use UpdatingBatch on documents	`Boolean`	Run the 'UpdatingBatch' feature of FileNet, at each punnet being processed.	
Default MimeType	`String`	The mime-type to set when no MimeType has been provided neither in document nor its content	
Limit CE connection life-time	`Long`	At end of TTL, the connection will be replaced by a brand new one. Default value is Long.MAX_VALUE	`9223372`
Fields to update	`String`	Default query to select fields to update	`*`
VersionSeries metadata	`String`	Name of the VersionSeries property common to all documents in punnet. If all documents have the same value, they will be considered as one same multiversioned document in FileNet.	`Version`
Post-commit delta	`Integer`	Time to wait after a commit instruction, may be useful to let FileNet perform	`0`

Key	Type	Description	Defa
		asynchronous handling of document injection	
Force folder creation	`Boolean`	Overwrite folder canCreate property : create folders when they do not exist	
Prevent document overwriting	`Boolean`	Check if the document already exists before creating it using `WHERE` clause. You can throw an exception in case an older document can be found (see *Throw exception if document already exists*). If false, create all documents without control	
Force to perform update	`Boolean`	Force document Update action. In case the document did not exist, an error is thrown	
WHERE clause for update	`String`	The criteria which the documents to update will have to match Ex/ [Id]=`${myFileNetDocumentId}`	

FlowerDocsIndexPlainText - Push input for plain-text search indexing in FlowerDocs

Mandatory settings

Key	Type	Description
FlowerDocs REST endpoint	`String`	
Username	`String`	
Password	`String`	
Scope	`String`	
Text for indexing	Pattern	

Optional settings

Key	Type	Description
FlowerDocs document ID	Pattern	

FlowerInjector - Fast2 injector module for FlowerDocs

Allows to load a component (document, task, folder or virtual folder) into Flower. Can load facts, document content and annotations

Mandatory settings

Key	Type	Description
FlowerDocs connection provider	FlowerDocsConnectionProvider	
Component category	`String`	FlowerDocs component category can be DOCUMENT, TASK, FOLDER or VIRTUAL_FOLDER

Optional settings

Key	Type	Description	Default value
	Component		
Load document annotations	`Boolean`		`false`
Load document file content	`Boolean`		`false`
Load component facts	`Boolean`		`false`
Mode update	`Boolean`	Does not update content	`false`

MailSender - Email sender task

This task will send custom built emails to specific people or mailing list of your choice.

Mandatory settings

Key	Type	Description
Email provider	MailSenderProvider	The Fast2 module establishing the connection to the email server, from the account of a given user. For more about the configuration of the object, please refer to the appropriate section

Optional settings

Key	Type	Description
Email subject	`String`	The subject/object of the email to send.
Sender address	`String`	The value of this field can be resolved by Fast2 (ie you can use dynamic values)
Recipient address(es)	`String list`	The value of this field can be resolved by Fast2 (ie you can use dynamic values)
Multi-line content	`String list`	The content of the email. It can be composed with different paragraphs, please note Fast2 is not responsible for the text formatting though.

MFilesInjector - Module to inject into M-Files via its public REST API. Java 11 is required for this module.

Module to inject into M-Files via its public REST API. Java 11 is required for this module.

Mandatory settings

Key	Type	Description
M-Files connection provider	`MFilesConnectionProvider`	Credentials to connect to M-Files remote system via its REST API.

MobiusInjector - Inject documents into your ASG Mobius system

> ⚠️ **Deleted**: The `MobiusInjector` task is deleted and no longer available in Fast2 from v2025.0.0.

This task will upload documents and metadata onto Mobius, from version 8 up to version 11. Based on the `className` and `section` properties, specify exactly where you'd like your documents to be stored.

Mandatory settings

Key	Type	Description
Mobius connection provider	MobiusConnectionProvider	The Fast2 module required to establish the communication with the destination Mobius instance

Optional settings

Key	Type	Description
Properties to inject into Mobius	`String list`	List of names of the properties which will be added to the topic of the document. These properties have to be attached to the document

MultiUpdateSQLQueryTask - Update a database with several document data

Mandatory settings

Key	Type	Description
WHERE clause	`String`	All matching rows will be updated. This field will be resolved by Fast2 before the task is executed
Query caller	SQLQueryGenericCaller	This modules is responsible of establishing the connection between Fast2 and the designated database

Key	Type	Description
Table name	`String`	The table of the row to update

Optional settings

Key	Type	Description	Default value
Skip exception	`Boolean`	Fast2 will either throw an error if no update action has been executed, or proceed to next document	`true`
Data to update	`String list`	The list of all key-values pairs to update the given rows with Ex/ targetColumnName/documentData	

NuxeoInjector - Nuxeo injector using Nuxeo REST API

This task load documents and metadata into a given Nuxeo instance using the Nuxeo REST API. If the document does not have any folder property, he will be injected in the workspace root folder. The documents **have to be in the correct version order** before entering the Nuxeo task. For Nuxeo to identify the versions as different, either the 'name' or the 'dc:description' data needs to be different.

Mandatory settings

Key	Type	Description
Nuxeo connection provider	NuxeoConnectionProvider	

Optional settings

Key	Type	Description	Default value
Attach punnet data	`Boolean`	Check this option to map the punnet data to Nuxeo properties	`true`
Blacklist	`String list`	List of metadata (either on punnet or document) not to map to the Nuxeo documents.	
Attach document data	`Boolean`	Check this option to map the document data to Nuxeo properties	`true`
Injection path	`String`	Default path to inject your documents	`/`
Attach folders	`Boolean`	Check this option to upload the documents into a specified folder architecture based on the details embedded in the document	`true`

Key	Type	Description	Default value
Delete annotations when they already exist	`Boolean`		`true`
Replace document if already present	`Boolean`	Check this option to replace all versions of a document in Nuxeo, based on the data `documentId`. This feature acts like a replacement. If the document did not already exist, then it will be created from scratch.	
Attach content	`Boolean`		`true`

OpenTextInjector - OpenText Content Server injector based on custom Rest API

Mandatory settings

Key	Type	Description
Attribute file path	`String`	OpenText category must be associated with their ids within the file. Fast2 will automatically

Key	Type	Description
		translate the data name to the correct id specified by the file Ex/ ../config/attributes.properties
OpenText credentials	OpenTextCredentials	
Expected folder architecture	`String list`	
OpenText client	OpenTextRestClient	

Optional settings

Key	Type	Description	Default value
List of properties not to inject	`String list`	These properties will be excluded	
NodeId of the webReport parameter to use	`String`	Opentext webReports allow users to build search request with specific parameters	

Key	Type	Description	Default value
List of properties to inject	`String list`	If empty the whole dataSet will be injected	
Version document if data exists	`String`	Fast2 will check if the data filled in this field for carrying the version and the 'nodeId' data are available at document level. If so, the document will be injected within OpenText and its version increased by one. A new data 'createdVersion' will be added to the document	
Ticket period	`Integer`	Time in seconds between two ticket creation	`60`

SQLQueryTask - Add data to documents in database

Simple task to query a SQL database and fill each Document data with results

Mandatory settings

Key	Type	Description
Key of target	`String`	The name of the data where the value must be added

Key	Type	Description
data		
Query caller	SQLQueryColumnCaller	This modules is responsible of establishing the connection between Fast2 and the designated database
Key of source data	`String`	The name of the data to update the documents with. If the data is not retrieved from the document, Fast2 will skip this document

Optional settings

Key	Type	Description	Default value
Reset target data	`Boolean`	Clean content when target already exists	`true`

SQLStatementTask - Insert or updated database

With this task, you will be able to perform any SQL instruction (such as insertions and updates) on any given table of the specified database

Mandatory settings

Key	Type	Description
SQL statement	`String`	The statement you want Fast2 to run on the database. The syntax needs to match SQL standards. Use a ? to reference your annotation Ex/ INSERT INTO table_name (doc_id, annotation) VALUES (`'${documentId}'`, ?);
Query caller	SQLQueryGenericCaller	This modules is responsible of establishing the connection between Fast2 and the designated database

Optional settings

Key	Type	Description
Inject annotations	`Boolean`	Fast2 will either throw an error if the statement has not properly been executed, or fail silently
Skip exceptions	`Boolean`	

UpdateSQLQueryTask - Update SQL database

This task will perform update instructions base on document data onto a given SQL database

Mandatory settings

Key	Type	Description
SQL connection provider	SQLQueryGenericCaller	The module establishing the communication between Fast2 and the designated database
Name of the new column	`String`	The name of the column which will be added to the row with the value to update
Table name	`String`	The name of the SQL table on which all update statements will be performed

Optional settings

Key	Type	Description	Default value
WHERE clause	`String`	All matching rows will be updated. This field will be resolved by Fast2. Leave empty to target all rows.	
Ignore when no row updated	`Boolean`	Skip exception when no database row has been updated	`true`

Key	Type	Description	Default value
Value to update	`String`	Name (= key) of the document metadata whose value will be inserted into the row. If none is found in the document, this latter is skipped	

Catalog / Importing or Replacing JAR Files from the Fast2 UI (available from v2025.2.0)

Overview

This feature allows you to import new JAR libraries or replace existing ones directly from the Fast2 user interface, without manual server intervention. The relevant workers are automatically restarted to apply the changes.

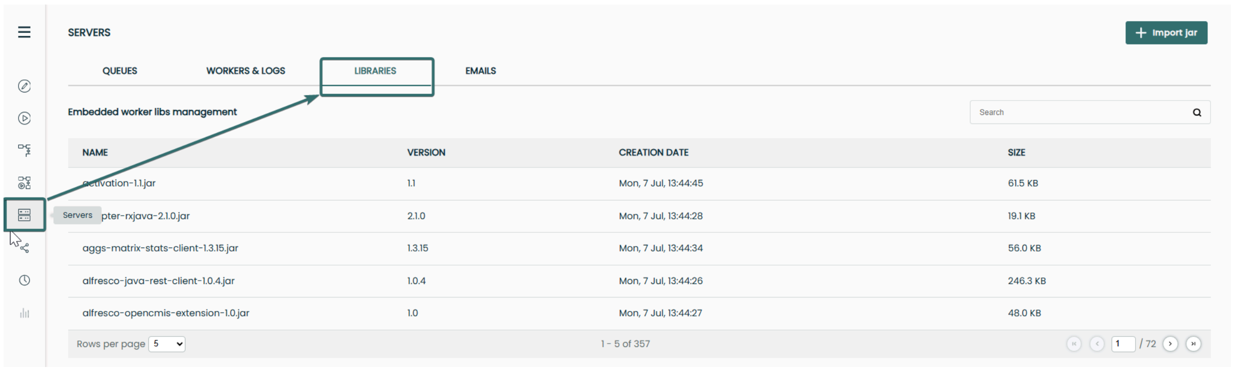
> **Note:** Due to server isolation, only embedded workers can be updated this way.

User Scenarios

1. Uploading a Valid JAR

> You must be authenticated as **ADMIN** or **SUPER ADMIN** to upload a JAR.

1. Go to **Servers Place → LIBRARIES** tab



Embedded worker libs management

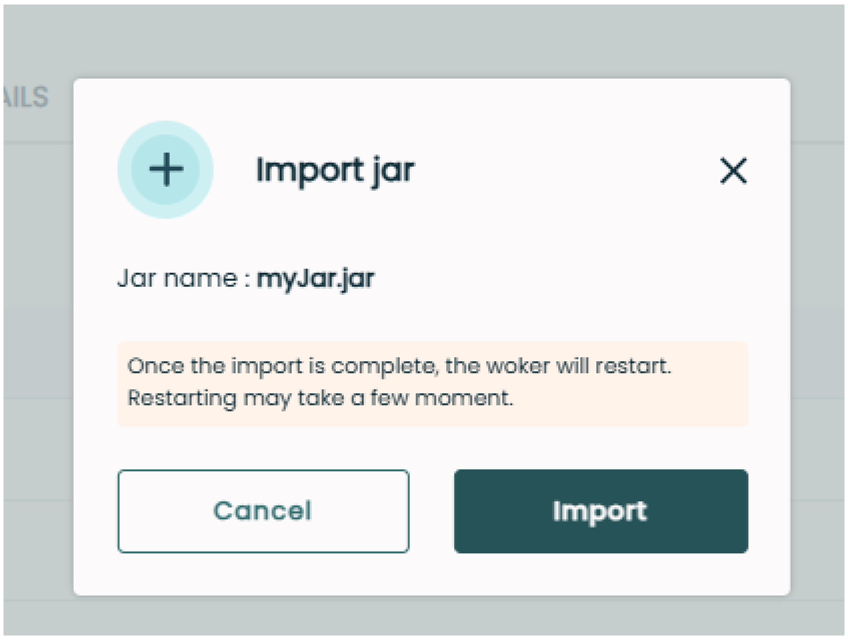
NAME	VERSION	CREATION DATE	SIZE
activation-1.1.jar	1.1	Mon, 7 Jul, 13:44:45	61.5 KB
pter-rxjava-2.1.0.jar	2.1.0	Mon, 7 Jul, 13:44:28	19.1 KB
aggs-matrix-stats-client-1.3.15.jar	1.3.15	Mon, 7 Jul, 13:44:34	56.0 KB
alfresco-java-rest-client-1.0.4.jar	1.0.4	Mon, 7 Jul, 13:44:26	246.3 KB
alfresco-opencmis-extension-1.0.jar	1.0	Mon, 7 Jul, 13:44:27	48.0 KB

Rows per page 5 1 - 5 of 357 1 / 72

2. Click on **Import jar** (top right button)

3. Select a JAR that does not exist in `/worker-libs`

4. Either **Cancel** or **Import** the JAR



Jar name : myJar.jar

Once the import is complete, the woker will restart. Restarting may take a few moment.

5. Workers on the same server as the broker are restarted

6. The JAR is added to `/worker-libs`

7. Worker restart and success messages are displayed

2. Replacing an Existing JAR

> You must be authenticated as **ADMIN** or **SUPER ADMIN** to replace an existing JAR.

1. Go to **Servers Place** as in step 1 above
2. Select a JAR already present (different version)
3. Workers are restarted
4. The old JAR is moved to `/worker-libs/versions` and renamed with `.old`
5. The new JAR is added to `/worker-libs`
6. Worker restart and success messages are displayed

Good Things to Know

- If one or more campaigns are running, they are stopped before upload
- Attempting to upload a non-.jar file will result in an error message

> To perform any operation, ensure you are logged in with the appropriate role (**ADMIN** or **SUPER ADMIN**) and use valid JAR files. Version management and campaign protection ensure Fast2 remains stable and available during library updates.

Advanced section

Learn here advanced handling of Fast2 for optimizing your migration process !

This section related of diverse topics for more specific uses of Fast2, but you will need to have the basic understanding of the overall concepts closely related to the architecture,

"DIVE RIGHT BACK IN"

In case of any confusion, please refer to the appropriate sections of the documentation, such as :

- the broker details,
- how the workers work,
- tool architecture

and any other relevant resources.

Advanced / Drools: the Java rules engine

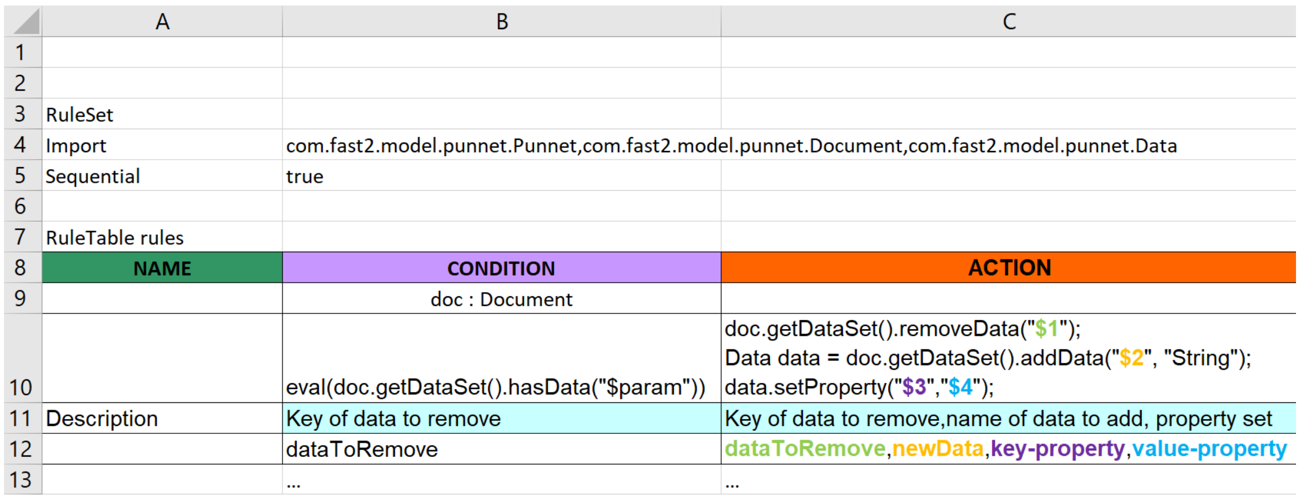
Based on Excel document, “drools” is a rule engine used to execute code scripts, Java code in our context. Users can define business and/or functional rules as data transformations, mapping, etc. One of the key benefits is its adaptation to any structure and any level of complexity as long as your code respects the punnet structure (quick reminder here if need be 😉). It can easily be shared between your team members for complex project to have concerned people seamlessly involved. Another upside: no development skill is required to build your own rules. Fast2 supports such feature with the ApplyDroolsTask.

A sample of Drools spreadsheet can be downloaded to help you getting started.

Download drools template

Spreadsheet structure

The following picture represents a drool sheet as you could find one in an Excel document:



	A	B	C
1			
2			
3	RuleSet		
4	Import	com.fast2.model.punnet.Punnet,com.fast2.model.punnet.Document,com.fast2.model.punnet.Data	
5	Sequential	true	
6			
7	RuleTable rules		
8	NAME	CONDITION	ACTION
9		doc : Document	

10		eval(doc.getDataSet().hasData("$param"))	doc.getDataSet().removeData("$1");
			Data data = doc.getDataSet().addData("$2", "String");
			data.setProperty("$3","$4");

11	Description	Key of data to remove	Key of data to remove,name of data to add, property set
12		dataToRemove	dataToRemove,newData,key-property,value-property
13		...	...

It’s composed with :

- **RuleSet**

  means that the current speadsheet is a decision table

- **Import**

  all java classes required, separated by a comma. These are the same packages that would be imported in a regular Java class in order to have the code running properly.

- **Sequential** (optional)

  specify here the order in which rules should apply

- **RuleTable** rules

  name of the table

- **NAME** column

  represents the name of the differents rules

- **CONDITION** column

  condition to verify to perform an action

- **ACTION** column

  action to perform if all previous conditions have been validated

- Variables used are indicated below the column CONDITION (doc : Document)

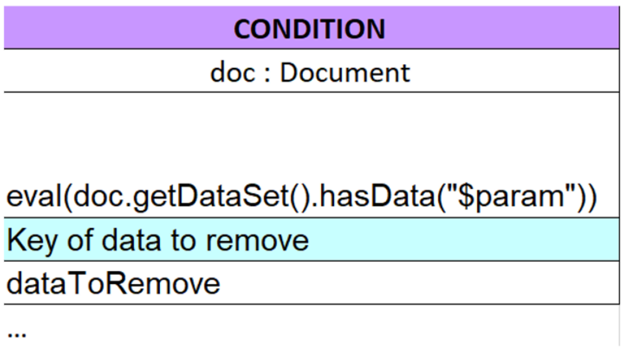
How to read

Quite simple! A rule is a row read from left to right, as regular code.

An empty row is interpreted by the rules engine as the end of the process. Each rule will have to meet particular criteria. There must be at least one condition and one action per rule.

Read a condition

- A rule can have multiple conditions
- All conditions must be validated to apply the action of the same row
- If no value is present in the condition column, the condition is skipped (considered `true`)

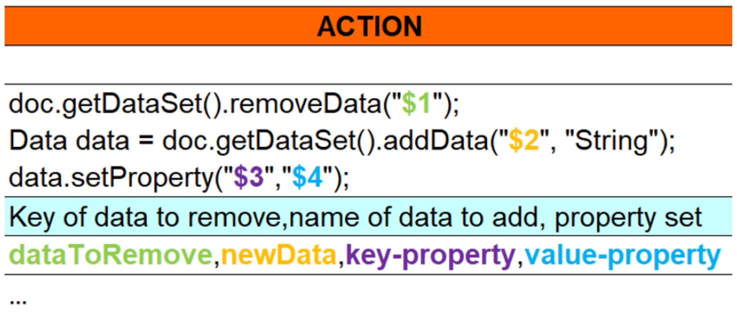


CONDITION
doc : Document
eval(doc.getDataSet().hasData("$param"))
Key of data to remove
dataToRemove
...

A condition cell will only hold one statement. If several conditions have to be met, they will be in the next columns.

Read an action

- A rule can have multiple actions
- Actions are performed from left to right
- Inside a cell, the actions are separated by a semicolon ;



ACTION

doc.getDataSet().removeData("$1");
Data data = doc.getDataSet().addData("$2", "String");
data.setProperty("$3","$4");

Key of data to remove,name of data to add, property set
dataToRemove,newData,key-property,value-property
...

The actions are read just like any code snippet, similarly to a regular script file.

Parameters

There are two different ways to use parameters:

- You only need one parameter for your condition or action : `$param`.
- Otherwise, separate values by comma, and use `$1`, `$2` and so on in you condition/action.

Write a condition

Conditions, just as in a regular coding snippet, must be performed as a boolean. Actions are executed only if condition is *TRUE*. It's highly recommended to use `eval(<condition>)` or `!eval(<condition>)` for conditions.

TIP

Just as you would write any condition in your code,

- Conditions must not end by a semi-colon (`;`)
- Characters allowed : `<`, `>`, `<=`, `>=`, `||`, `&&`, ...

If you want to perform an action no matter what, do `eval($param)` with `$param = true`.

If you need the document to have a specific data before making any action, do:

`doc.getDataSet().hasData($param)` with `$param = yourDataName`.

Action examples

TIP

You can put any Java code to perform an action, as long as you end each instruction by a semi-colon (`;`).

Add new data

To add a new data, if you know both the key/name and the value, use the following code :

```
doc.getDataSet().addData("<key>", "<type>","<value>");
```

In case the value is unknown at the moment or you object is too complex and you might need to add properties to the data object:

```
Data data = doc.getDataSet().addData("<key>", "<type>");

data.setProperty("<key>","<value>");
```

When performing such operation, though, don't forget to add the proper Fast2 package to manipulate Data type.

Add new value to existing data

Add one value:

```
doc.getDataSet().getData("<data-key>").addValue("$param");
```

Add multiple values to the same data:

```
doc.getDataSet().getData("<data-key>").getValues().addAll(<list-of-
values>);
```

Stop a rule

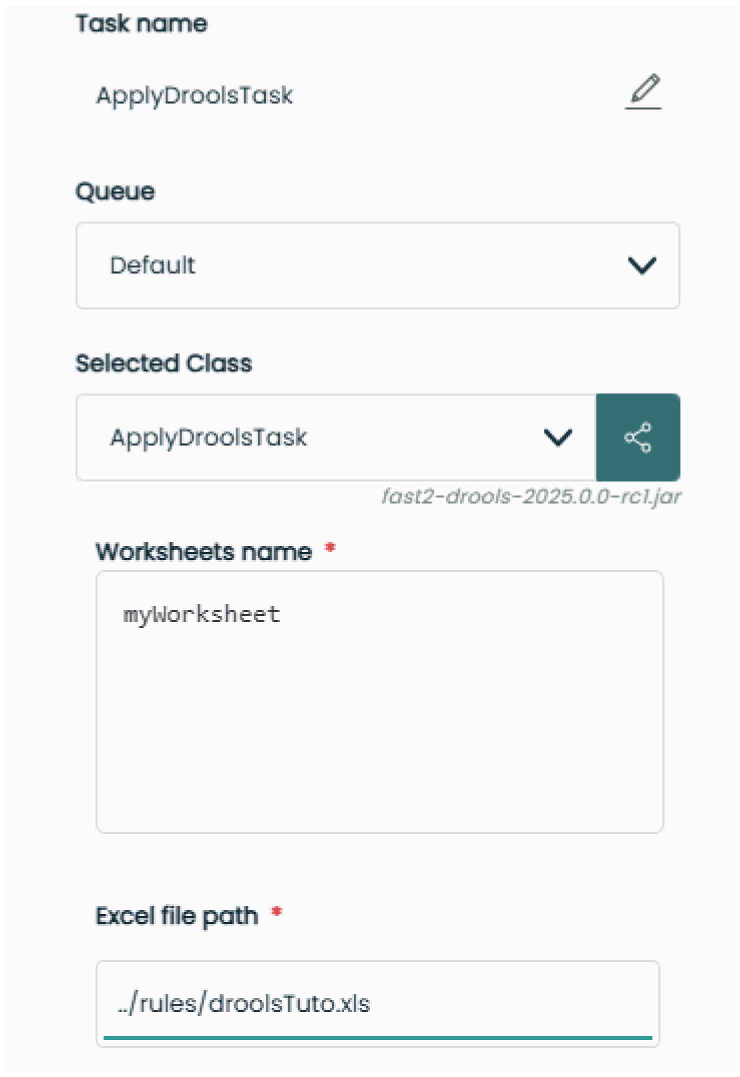
You can stop the rule execution at a specific time when an action has been performed. Use `drools.halt();` in the action section.

The next action(s) will not be performed as the rule execution is stopped (useful in case of error management).

Good practices

We advise you to create a folder at the root of Fast2 and name it Rules. However, Fast2 will be able to fetch your drools files anywhere as long as the specified path is accessible to the Fast2 server.

This path will be fill in the the task ApplyDroolsTask.


Task name

ApplyDroolsTask

Queue

Default

Selected Class

ApplyDroolsTask

fast2-drools-2025.0.0-rc1.jar

Worksheets name *

```
myWorksheet
```

Excel file path *

../rules/droolsTuto.xls

Advanced / Patterns

A pattern is a sequence of instructions, a model, which can be easily recognized by an aware glance. It is strictly under this definition that Fast2 patterns stand.

Our migration tool relies on a specific syntax used to dynamically retrieve information from different data-layer of the whole process, whether document, map execution or else.

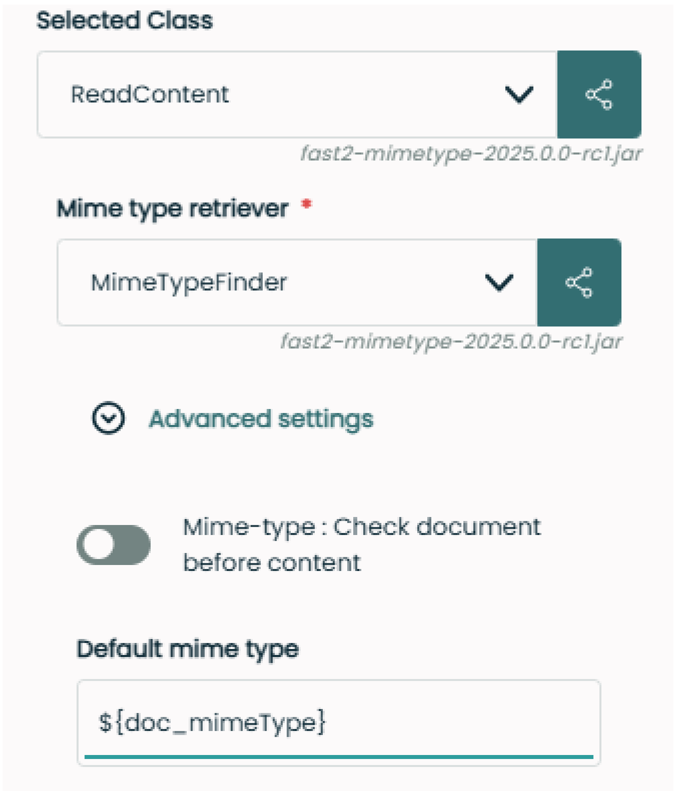
TIP

For a successful pattern recognition, only use the data known by Fast2, such as :

- Data stored in the punnet and/or documents datasets
- Migration specific details, listed down below.

Patterns, what are they anyway ?

In Fast2, several data can be retrieved and accessed dynamically from a dedicated syntax which Fast2 supports for your convenience. This `${...}` syntax can be used in most of the configuration fields of the tasks composing your migration workflow.


Selected Class
ReadContent
fast2-mimetype-2025.0.0-rc1.jar
Mime type retriever *
MimeTypeFinder
fast2-mimetype-2025.0.0-rc1.jar
Advanced settings
Mime-type : Check document before content
Default mime type
${doc_mimeType}

Using such syntax will comes in handy when you will have to rely on a value whose you only know the name. In other words, retrieving a metadata whose key is `doc_mimeType` and value is unique for every document, will just be `${doc_mimeType}`.

No need to list all your possible values, Fast2 will resolve this expression by looking first at the document dataset level, then at the punnet level, an later at map/campaign level. Some applications of the latter could be to store the name of the map, or even accessing map- or global-scoped shared objects for cross-campaign communications.

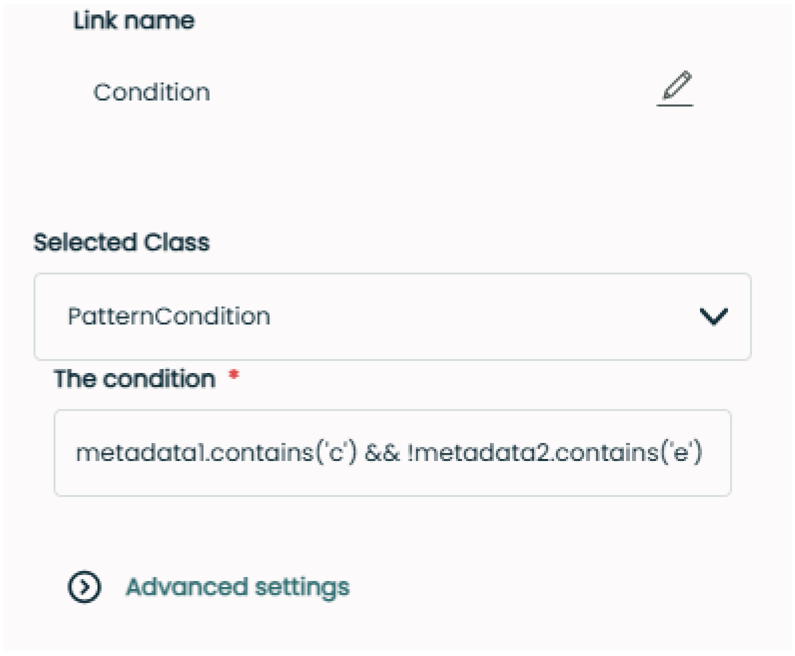
Patterns in links

Although links are designed to offer basic statements for conditional routing, they also digest pattern for higher-complexity conditions. For example, new conditions can be value-dependent: not only you can check whether the document has a given data, but now it is possible to narrow down the eligible documents based on the value itself of this data.

NOTE

Pattern-related syntax `${...}` is not required, fill the field with your expression directly !

Based on SpEL (Spring Expression Language), the syntax of these conditions will sound familiar to anyone who's already coded one day:


Link name
Condition

Selected Class
PatternCondition

The condition *
metadata1.contains('c') && !metadata2.contains('e')

In the same way, you'll now be able to sort documents based on their mime-types, on their structure (does my document has a content ? Is its creation date matching the time range which this campaign is focusing on?).

As mentioned earlier, the list of data which you can evaluate in a condition is the same list as in a task configuration (document properties, punnet properties, map and campaign names).

As long as it returns `true` of `false`...

More complex use-cases can be built out of the given tools, as long as the syntax matches the SpEL expression. Java-based condition are therefore supported, here are some examples to help you getting started :

- `documentId.endsWith("0")`
- `mimeType.startsWith("image/")`
- `documents.size() >= 2 || punnet.getDataSet().hasData("multiversioned")`

And the list goes on, it's your turn now to build the condition meeting your needs!

Patterns subtleties

Properties with colon

As handy as they may sound, patterns do embed specifications due to the particular syntax they are subjected to.

The most common issue is when dealing with colon character `:`, but the approach also addressed data name with space characters. To prevent running into a SpEL syntax error which would wipe its interest out, the syntax has to be slightly expanded. Where before you were accessing the value with `${key}`, you now need to write it as follows:

```
${property('prefix:suffix')}

${property('with space')}
```

You can now safely extract data with namespaces, or any special character which may eventually break the SpEL syntax.

Propose default value

In case the pattern value is not known by Fast2, an empty String is return. However, you might be willing to set a default value to ease the upcoming operations.

To do so, use the Elvis ternary operator along the `property()` function:

```
${property('missingData')?:'defaultValue'}
```

From now on, if the 'missingData' is not fount either at the punnet or document level, the value you earlier planned to retrieve will be replaced with the value set as default, although no additional property is created.

Access data of Fast2 objects

Whether you need subtypes properties for conditional routing or metadata elaboration, Fast2 gives you access to any data stored in the punnet.

However targetting object is not always intuitive, so here are the different keywords required to access the Fast2 objects :

Keyword	Description	Examples
`${CurrentDocument}`	Access the focused document, to call its	`${CurrentDocument.getDataSet().g` `data').getValues().get(0).split(`

Keyword	Description	Examples
	metadata. This can be quite useful when dealing with multi-document punnets.	
`${CurrentContainer}`	Access the focused content, to call its properties. This can be quite useful when dealing with multi-contented documents.	`${CurrentContainer.mimetype}`
`${CurrentAnnotation}`	Access the annotation of the document.	`${CurrentAnnotation.annotationId`

Keyword	Description	Examples
`${punnet}`	Access the punnet as an object. From there, all datasets and subobjects can be accessed. The accessor is generally used for conditions.	`${punnetId.toString().startsWith`
`${documents}`	The list of the documents stored in the punnet.	`${documents.size()}` `${documents.`
`${step}`	The name of the step where the pattern is called.	
`${map}`	The name of the map	`${map}/my_output_file.csv`

Keyword	Description	Examples
	which is run during this campaign. Often used for output directory names	
`${campaign}`	The name of the campaign. Often used for output directory names	`${map}/${campaign}/my_output_fil`

Using Java classes

Pattern can also be used to enrich data, relying on the basic Java classes.

TIP

The required syntax is `T(clazz)`.

For example, adding an UUID created on-the-fly would just required using the following pattern:

```
${T(java.util.UUID).randomUUID().toString()}
```

An other example could be to add today's date as a new data. Such a pattern might go like:

```
${T(java.time.LocalDate).now().toString()}
```

Bringing it further, we might also want the time when the document got through the migration (namely this AlterDocumentProperties task), with

```
${T(java.time.LocalDateTime).now().toString()}
```

Mapping these patterns respectively to a new `today` and `currentTime` data will result in the following punnet :

```
{
        "punnetId": "doc_0_0#1",
        "data": {
        ...
        },
        "documents": [
                {
                        "documentId": "doc_0_0",
                        "data": {
                                ...
                                "currentTime": "2023-06-
14T10:42:15.204958800",
                                "today": "2023-06-14"
                        }
                }
        ]
}
```

Advanced / Schedule your campaigns

Fast2 has a module allowing you to schedule your next runs. This feature is accessible only through the run place. Click on the clock icon at the top banner and you're in.

Jobs purpose

A table of jobs is displayed, each line representing a job. The latter will be used to plan your next runs. However, a few details are needed in order to let the job perform the given action.

A job is composed as follow :

- A **unique name** composed exclusively of alphanumerical characters. Dash and underscore are accepted, others are forbidden. It's imposible to create two jobs with the same name.
- The map to run : you can pick in the dropdown list any map created earlier.
- A campaign : specify the name the dedicated input field. You can either write the name of an already existing campaign or a new one.
- A CRON expression to schedule precisely your runs.
- An action among `Start as new`, `Rerun`, `Stop` or `Resume`. It's precisely the same actions you can do in the run place.
- A maximum number of executions, if you want to limit the number of times the job will run.
- A boolean *activate* to indicate if you're job must be ran or not (useful feature to have a job stand by, already configured for later operations).

The other columns are purely indicatives. You will find the number of executions for each job, the date of the last and when the next run has been planned.

Jobs are automatically saved when all their fields are correctly filled. If the name or the CRON expression have errors, you have to fix these errors to make the job savable.

Jobs creation and deletion

- Button **CREATE**: Located at the end of the first row of the table, this button allows adding the entered values to create a new job. Once clicked, the entered values are used to create a new job, which is then added to the list of already created jobs. Users can then see their new job appear in the list of jobs.
- Button **DELETE**: Found at the end of each row representing an existing job, this button allows users to delete a job from the list. By clicking on this button, users can permanently delete a selected job. Before confirming the deletion, a confirmation is requested to prevent any accidental deletion.

Caution

Despite Fast2 validates the name and the CRON expression, the match between the campaign and the job action is not tested. It is the user's responsibility to have a clear idea about what to do with which campaign.

If the action of a job is to stop a campaign, nothing will happen if the campaign is not running. The behavior is the same as the run place. See our run section as a reminder in case you need it.

Additionally, it is important to note that when executing a job as *START AS NEW*, the next-to-come campaign will be triggered only once the previous campaign ends (status *FINISHED*). This ensures sequential execution and prevents overlapping campaigns, maintaining the integrity of the workflow.

"FOR EXAMPLE"

For example, even if the CRON expression indicates a frequency of 10 seconds with the CRON expression `*/10 * * * * *` between each trigger, the next campaign will only start once the previous one is finished.

:

Cron expression

A CRON is a String composed of 6 or 7 fields separated by spaces. Each field represents a specific section of time. The 1st field will be the second unit, the 2nd one will concern the minutes and so on. All the fields are listed below with the allowed characters.

Field Name	Mandatory	Allowed Values	Allowed Special Characters
Seconds	✅	0-59	, - * /
Minutes	✅	0-59	, - * /
Hours	✅	0-23	, - * /
Day of month	✅	1-31	, - * ? / L W
Month	✅	1-12 or JAN-DEC	, - * /
Day of week	✅	1-7 or SUN-SAT	, - * ? / L ##

Field Name	Mandatory	Allowed Values	Allowed Special Characters
Year	❌	empty, 1970-2099	, - * /

Examples of cron

These examples are basic ones but they highlight the interest of using cron expressions.

- Every 2 minutes

Seconds	Minutes	Hours	Day Of Month	Month	Day Of Week	Year
0	0/2	*	?	*	*	*

- Fire at 10:15am every day

Seconds	Minutes	Hours	Day Of Month	Month	Day Of Week	Year
0	15	10	?	*	*	*

- Fire every Sunday at noon

Seconds	Minutes	Hours	Day Of Month	Month	Day Of Week	Year
0	0	12	?	*	SUN	*

Cron expressions can be really useful but might quite hard to use.

"BUILD IT YOURSELF"

To generate your own cron expressions easily, we suggest you to use some web generator (like freeformatter.com.

Maximum number of executions

The **Max # of executions** input field allows users to specify the maximum number of times a campaign should be executed.

- Blank: If left blank, the campaign will be executed indefinitely.
- Invalid Input: If the user enters a number that is equal to or less than the number of runs already completed, an error message "Must be higher than actual # of executions" will be displayed. The input field will retain its last valid entry.
- Valid Input: A valid input is one where the number entered is greater than the number of executions already performed, or left blank. Upon entering a valid number, the value will be saved. If the "Active" checkbox was disabled and all other fields are valid, it will become enabled.
- Execution Limit Reached: Once the number of runs reaches the value entered in this field, returning to the scheduler will display the message "Number of executions reached."

Advanced / Shared objects

WARNING

This page has been moved to the Knowledge-Base at 🔗Understand the Shared Objects in Fast2

Advanced / JavaScript

Using the JSTransform task can comes in quite handy for any tweaking of metadata, but will also get you covered in case of heavier operations, as long as your JavaScript talents match you problem solving skills ! To iterate through all documents crossing your JS task, here is a short code snippet to help you get started:

```
punnet.getDocuments().forEach(function (doc) {
    // do something
});
```

Any data within a punnet can be accessed, added, edited or remove ! Make sure beforehand to respect the punnet object architecture (quick reminder here if need be 😉).

Let's now go through 3 quick scenarios.

Example #1 Map document properties from JSON

Depending on your use-case, the metadata could have been stored within a JSON file. Parsing such file and build a punnet based on its content is another kind of operation where this Fast2 task comes in handy !

The following script is one way to parse a basic JSON metadata file.

```
// First, get content as String
var content =
punnet.getDocumentList().get(0).getContentSet().getContent().get(0);
var bytes =
manager.getPunnetContentFactory().getContentAsByteArray(content);

var String = Java.type("java.lang.String");
var lines = new String(bytes);

// Then, parse this fragment as JSON
var jsonObject = JSON.parse(lines);

// Clear existing document and create a new one
punnet.getDocumentList().clear();
var doc =
punnet.addDocument(com.fast2.model.punnet.DocumentId.id());

// Fetch all properties from existing fragment and create them as
Document data
for (pty in jsonObject) doc.getDataSet().addData(pty, "String",
jsonObject[pty]);
```

Once this script is executed, you'll end up with a punnet whose first and only document will have its dataset full of metadata matching both keys and values from the JSON file.

Considering the following input embedded in a JSON file:

```
{
    "name": "testName",
    "contentPath": "C:/path/to/sample.pdf",
    "key": "value"
}
```

the ouput punnet would then look like this:

```
{
    "documents": [
        {
            "data": {
                "contentPath": "C:/path/to/sample.pdf",
                "key": "value",
                "name": "testName"
            },
            "documentId": "ffde4769-3acd-4964-ab72-5912f1e65e1e"
        }
    ],
    "punnetId": "punnet.json#1"
}
```

Next step could be to attach the document content to your document, now that you have the `contentPath` data with its value easily resolved by the AlterDocumentContent task.

Example #2 Delete content based on property

Let us now supposed we want to filter out document contents based on a given property. For convenience, the reference value is stored at the punnet level, under the property `punnetKeyA`.

The filter criterion is the following: if the content has the value of its data `contentKeyA` matching the punnet value, the content is left in place. Otherwise, the content is deleted.

Before the punnet enters the JSTransform task, its structure looks like this:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<ns:punnet xmlns:ns="http://www.arondor.com/xml/document"
punnetId="doc_0_0#1">
        <ns:documentset>
                <ns:document documentId="doc_0_0">
                        <ns:contentset>
                                <ns:content>
                                        <ns:property
name="contentKeyA" value="valueA" />

<ns:url>path/of/first/content</ns:url>
                                </ns:content>
                                <ns:content>

<ns:url>path/of/second/content</ns:url>
                                </ns:content>
                        </ns:contentset>
                        <ns:dataset />
                        <ns:folderset />
                        <ns:annotationset />
                </ns:document>
        </ns:documentset>
        <ns:dataset>
                <ns:data name="punnetKeyA" type="String">
                        <ns:value>valueA</ns:value>
                </ns:data>
        </ns:dataset>
        <folderSet />
</ns:punnet>
```

The first content should then remain as is, while the second one is expected to be removed by the script of our task.

Speaking of it, the strategy will be to iterate through all the documents of the punnet; and for each document, iterate through all its contents. Finally, a simple condition will evaluate whether the content is to be kept.

```
punnet.getDocuments().forEach(function (document) {
    var duplicate = [].concat(document.getContentSet());

    duplicate.forEach(function (content) {
        if (content.getProperty("contentKey") !=
punnet.getDataSet().getDataValue("punnetKey"))
            document.getContentSet().remove(content);
    });
});
```

Once the task is completed (in other terms, once the script has been executed), the output punnet new structure is lightened:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<ns:punnet xmlns:ns="http://www.arondor.com/xml/document"
punnetId="doc_0_0#1">
        <ns:documentset>
                <ns:document documentId="doc_0_0">
                        <ns:contentset>
                                <ns:content>
                                        <ns:property
name="contentKeyA" value="valueA" />

<ns:url>path/of/first/content</ns:url>
                                </ns:content>
                        </ns:contentset>
                        <ns:dataset />
                        <ns:folderset />
                        <ns:annotationset />
                </ns:document>
        </ns:documentset>
        <ns:dataset>
                <ns:data name="punnetKeyA" type="String">
                        <ns:value>valueA</ns:value>
                </ns:data>
        </ns:dataset>
        <folderSet />
</ns:punnet>
```

As expected, the second content is gone. If the content to filter out was inside a parent content of this document (i.e. punnet > document > first level content > second level content), this JS code should have been adapted to add one deeper level of content scanning:

```
punnet.getDocuments().forEach(function (document) {
    document.getContentSet().forEach(function (firstLevelContent) {
        var duplicate =
[].concat(firstLevelContent.getSubContents());
        duplicate.forEach(function (secondLevelContent) {
            if (
                secondLevelContent.getProperty("contentKeyA") !=
                punnet.getDataSet().getDataValue("punnetKeyA")
            )

firstLevelContent.getSubContents().remove(secondLevelContent);
        });
    });
});
```

Example #3 Get content path

Another application of this task could be to reach values outside the scope of the document dataset, like for example the path of the associated content.

This value is stored tightly within the content, which is why passing via the manager component of the JSTransform task is required.

```
punnet.getDocuments().forEach(function (doc) {
    var path =
manager.getPunnetContentFactory().getContentAsFile(doc.getContentSet(
    doc.getDataSet().addData("pathCopy", "String", path);
});
```

The temporary variable `path`, which the value of the absolute path will be stored into, can thereafter be accessed as a regular metadata under the key `pathCopy` alongside the other document metadata.

We can envision further operations of this value (or any other from the dataset), such as truncating, hashing, comparison etc.

Bonus tip

When a Java class needs to be manipulated within the JavaScript code, a variable analogous to a Java class needs to be created:

```
var Id = Java.type("com.example.package.Id");

punnet.getDocuments().forEach(function (doc) {
  var id = Id.id("myStringId");

  ...
});
```

In the end, it's all just about you

As you can imagine, several other use-cases can be addressed by implementing on-the-fly JS scripts, just like the ones showed above. The capabilities of this task are endless: team with Fast2 on this fight floor, and tackle your migration with your sharpened coding skills !

Advanced / Custom module

WARNING

This page has been moved to the Knowledge-Base at 🔗Build a custom module in Fast2

Fast2 cookbooks

INFO

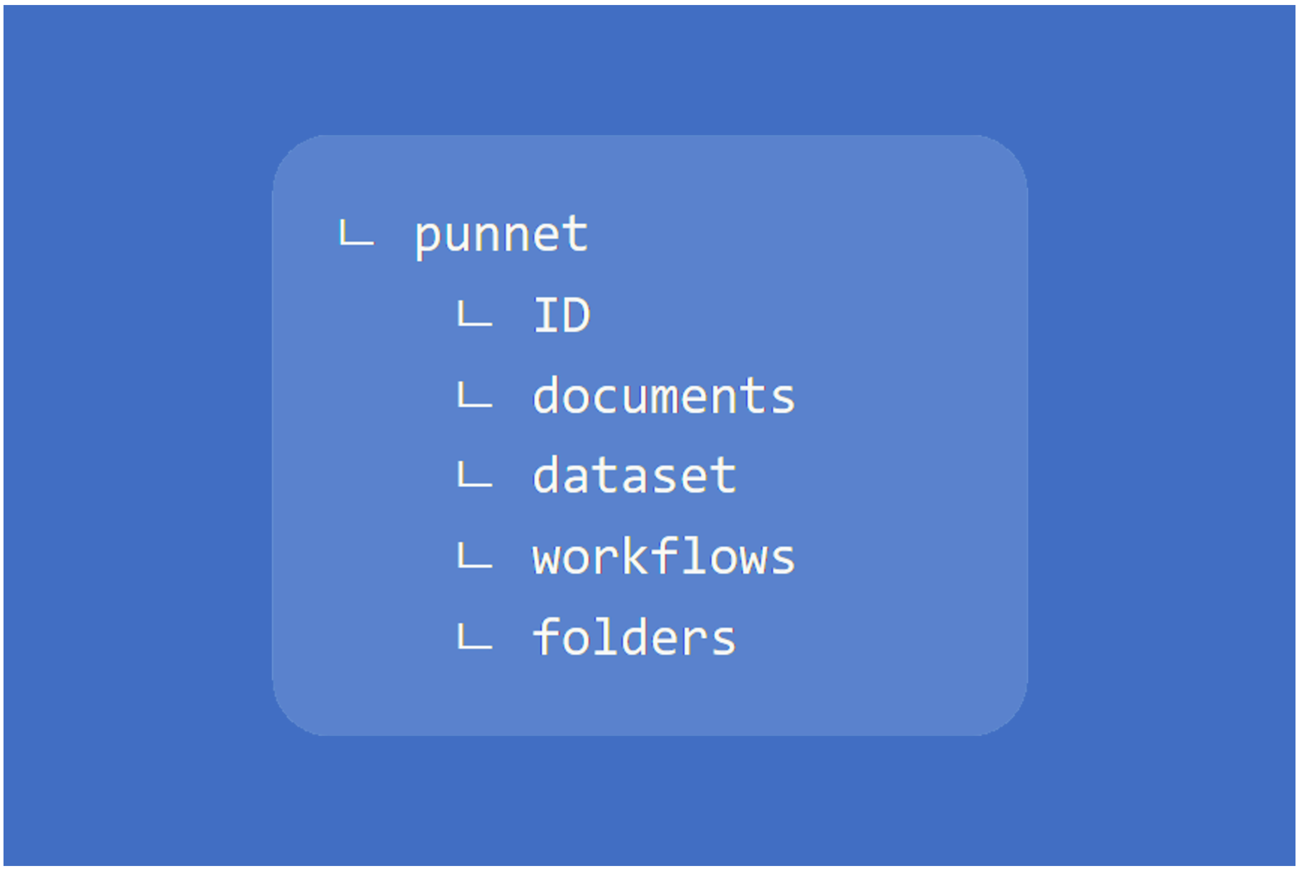
A cookbook in the programming context is collection of tiny programs that each demonstrate a particular programming concept. The Cookbook Method is the process of learning a programming language by building up a repository of small programs that implement specific programming concepts.

Here are some examples of short use-cases with Fast2, covering APIs management which might be required for configuring some of the tasks.

This section will particularly be insightful for custom module development.

You will find here boilerplate code snippets, as well as real-life examples of task configuration for challenges you might take up, one day.

Cookbooks / Learn how to deal with punnets



Since the punnet is the pivot format into Fast2, all documents, folders and metadata can only be manipulated through this object. Here are the basics for an appropriate understanding of the Java API of the punnet object.

For a better understanding of the following, the punnet structure needs to be clearly visualized. If required, go back to the definition of such an object in Fast2.

Creating a punnet

The creation of a punnet is an operation which should only take place in source tasks, as all other tasks only process punnets given to the as input.

To create a punnet:

```
Punnet myPunnet =
task.getManager().getPunnetFactory().createEmptyPunnet();
```

The punnet ID will be automatically computed by Fast2, following certain models based on the task producing the punnet. This ID can be useful to track the punnet over different tasks, as its value will remain unchanged.

ID

To force the ID of the punnet, a `String` value can be passed as argument.

```
myPunnet.setPunnetId(PunnetId.id("myId"));
```

Retrieving the ID of the punnet just goes as so:

```
PunnetId punnetId = myPunnet.getPunnetId();
```

Sending a punnet into the campaign

The source tasks are responsible for not only create the punnet, but also send it in the migration workflow.

Once the punnet is created and its components are correctly formed, the last step of the source will rely on the `Consumer` input parameter of the main method of the connector :

```
consumer.push(myPunnet);
```

Documents

Punnets may or may not embed document(s).

Several-documents cases could be like:

- the migration strategy is to group in a punnet all documents stored in a given folder
- the migration strategy is to group in a punnet all documents matching a criterion (depending of the configuration fields of the source)
- a punnet embed a released version of a document and all its past versions.

Whatever the reason, the documents can be added to the punnet in two ways:

1. The document was **already existing**:

   ```
   Document myDoc = myPunnet.addDocument(myDocument);
   ```

2. The document **needs to be created**, which can be done on the fly:

   ```
   Document myDoc = myPunnet.addDocument(DocumentId.id());
   ```

All documents can be access via the list of all documents stored in the punnet:

```
List<Document> myDocuments = myPunnet.getDocuments();
```

As any Java list, documents can be removed as long as the correct index is provided.

For more information concerning the documents, head out to the dedicated section.

DataSet

The purpose of the punnet dataset is to store metadata not closely related to any folder or document specifically.

This dataset can be access via an usual getter:

```
DataSet myDataset = myPunnet.getDataSet();
```

A punnet is built with an empty dataset by default.

For more information concerning the dataset, head out to the dedicated section.

Folders

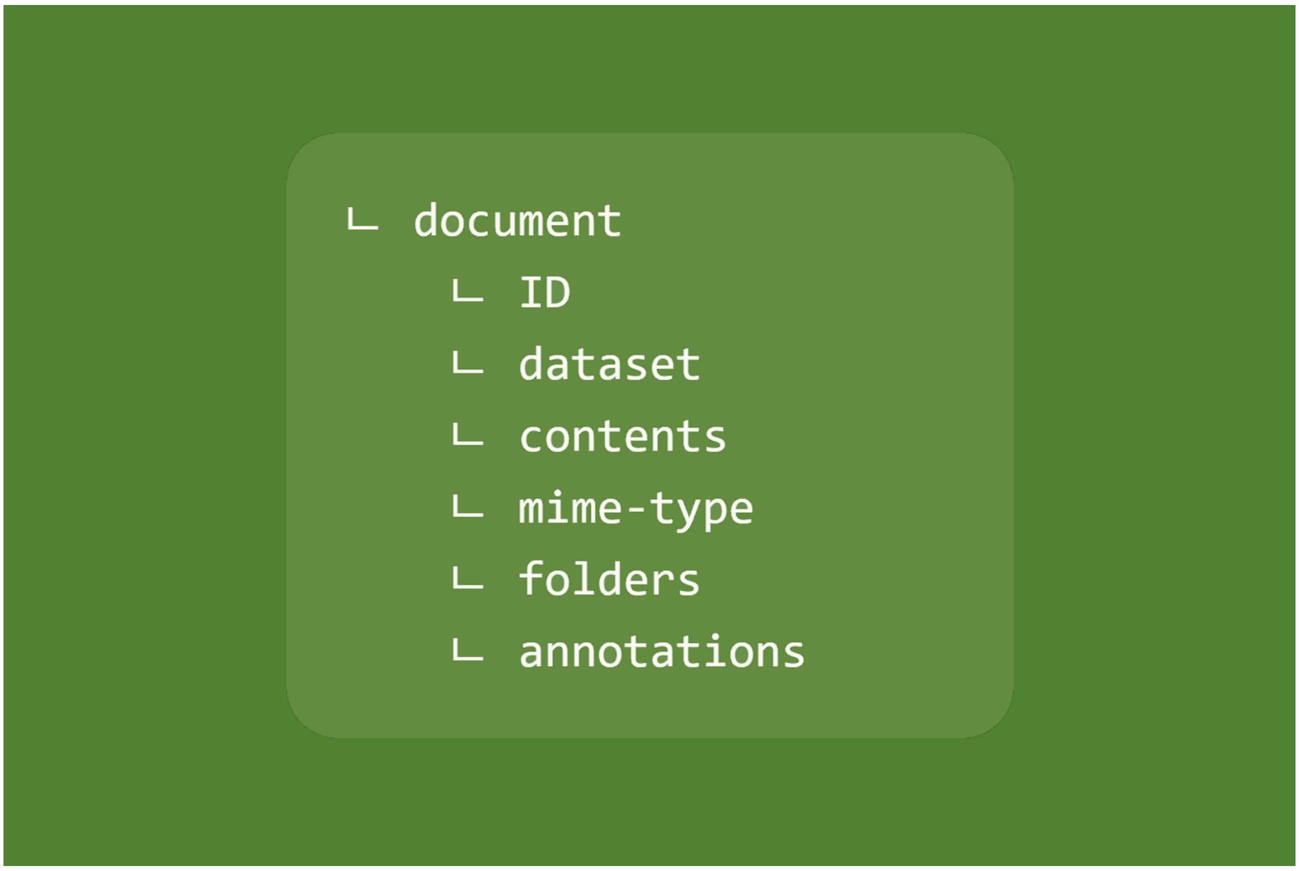
The punnet folderset can be used for folders-only migration and well as folder-as-a-whole ones, where a punnet will contain a folder reference and all the documents previously filed into this folder.

A folderset can contains one or several folder references, and is access as follows:

```
FolderSet myFolders = myPunnet.getFolders();
```

A punnet is built with an empty folderset by default.

Cookbooks / Learn how to deal with documents



The documents are a main part of any migration, if not the purpose of it. Here are the basics for an appropriate understanding of the Java API of the document object.

For a better understanding of the following, the document structure needs to be clearly visualized. If required, go back to the definition of such an object in Fast2.

Creation

As explained in the basics of punnet a document can be created on the fly:

```
Document myDoc = myPunnet.addDocument(DocumentId.id());
```

If required, the document ID can be force by adding a parameter into the document ID creation:

```
Document myDoc = myPunnet.addDocument(DocumentId.id("myDocId"));
```

However the ID can be forced after the document creation:

```
myDoc.setDocumentId(DocumentId.id("myDocId"));
```

Later on, its ID can be retrieved just like the punnet's:

```
DocumentId myId = myDoc.getDocumentId();
```

DataSet

The purpose of the document dataset is to store metadata closely related to its entity. When data are not too tighly related to the content of a document, chances are they will be stored as this dataset level. The mime-type does not follow this rule, though.

This dataset can be access via an usual getter:

```
DataSet myDataset = myDoc.getDataSet();
```

A document is built with an empty dataset by default.

For more information concerning the dataset, head out to the dedicated section.

Contents

In Fast2, a document can have no to several contents.

No-content cases could be like:

- the document does not have a content, originally
- the content has already been migrated
- the migration is just an update with only a few metadata to send to the destination

Several-contents cases could be like:

- the document has attachments (1 content per attachment)
- the document has content of different types (e.g. a PDF file alongside a TIFF file)

However the ratio 1-content-for-1-document is quite common.

Contents are accessed via the `ContentSet` which basically is a collection of `ContentContainer`s:

```
ContentSet myContents = myDoc.getContentSet();
```

A document is built with an empty contentset by default.

For more information concerning the contentset, head out to the dedicated section.

Mime-type

All documents provide a shortcut to their first content mime-type, under the `mimeType` data stored in the document dataset:

```
String myMimetype = myDoc.getMimeType();
```

This data can also be set from the document level:

```
myDoc.setMimeType("myMimetype");
```

As said earlier, this method is just a shortcut to add a mime-type data into the document dataset.

Folders

The document folderset can be used for folders-only migration and well as folder-as-a-whole ones.

A folderset can contains one or several folder references, and is access as follows:

```
FolderSet myFolders = myDoc.getFolders();
```

A document is built with an empty folderset by default.

For more information concerning the folders, head out to the dedicated section.

Annotations

The document can embed zero to several annotations alongside its contents or data.

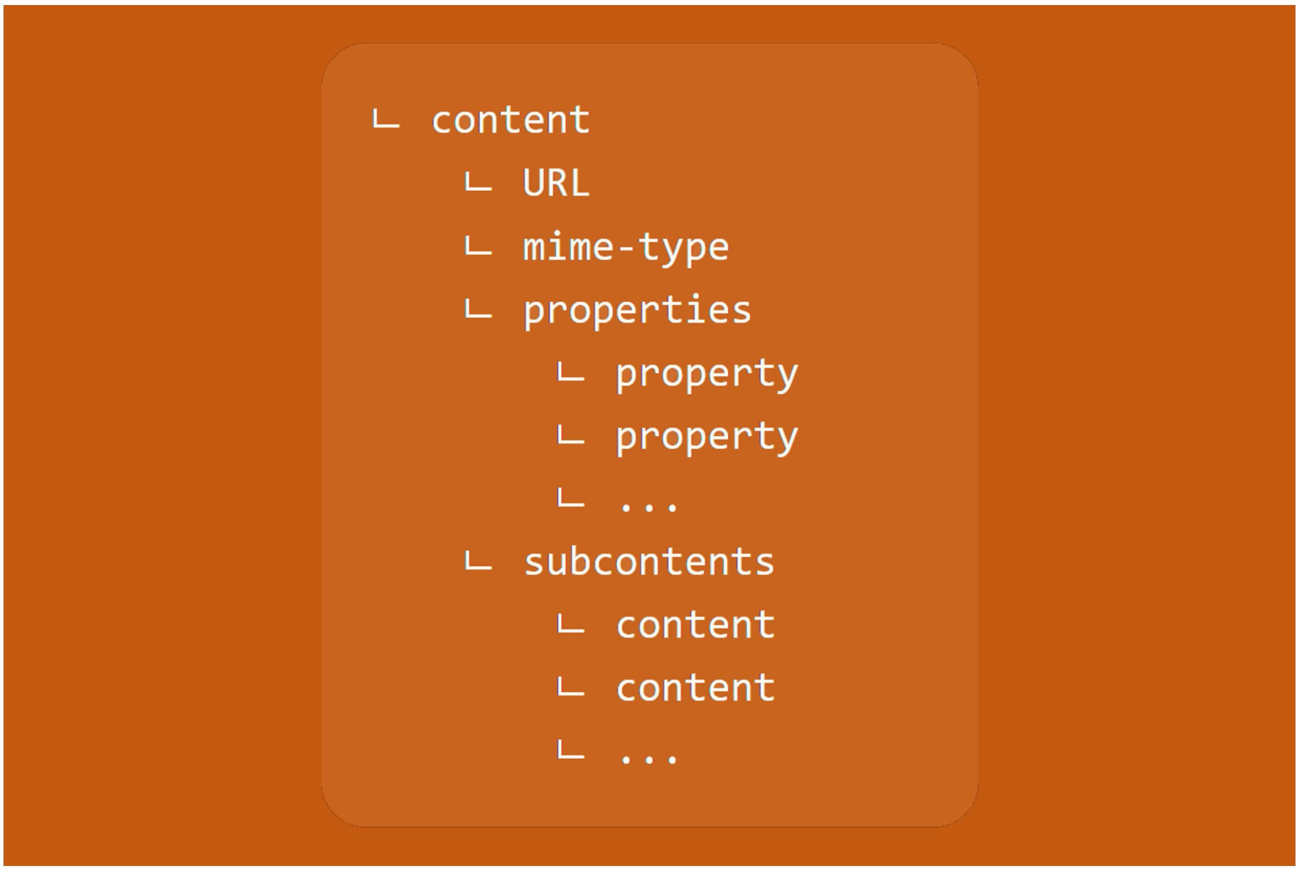
The collection of these annotations is called an `AnnotationSet`, and can be accessed as follows:

```
AnnotationSet myAnnotationSet = myDoc.getAnnotationSet();
List<Annotation> myAnnotions = myAnnotationSet.getAnnotationList();

myAnnotationSet.addAnnotation(myAnnotation);

Annotation newAnnotation =
myAnnotationSet.addAnnotation(myAnnotatioContent);
```

A document is built with an empty annotationset by default.

From a Fast2 standpoint, a annotation is just a object composed by an ID and a content.

Cookbooks / Learn how to deal with contents



```
∟ content
    ∟ URL
    ∟ mime-type
    ∟ properties
        ∟ property
        ∟ property
        ∟ ...
    ∟ subcontents
        ∟ content
        ∟ content
        ∟ ...
```

In Fast2, contents are objects embedding the "file" (= binary format) of the document. They can be found within either documents themselves or annotations, and can be accessed through different ways. Contents usually hold a mime-type property, alongside any other property closely related to the content itself.

Contents are often referred as **ContentContainers**.

How to create a content

This section relates of how to add a content from the code.

If you wish to add a content (or delete it), head out to the AlterDocumentContent task.

```
// From an URL or a path
ContentContainer myContent = task.getManager()
                    .getPunnetContentFactory()
                    .createContent(myDocument, myUrl);


// From an inputstream
ContentContainer myContent = task.getManager()
                    .getPunnetContentFactory()
                    .createContent(myPunnet, myDocument,
myInputStream);


// From a byte array
ContentContainer myContent = task.getManager()
                    .getPunnetContentFactory()
                    .createContent(myPunnet, myDocument,
myByteArray);
```

How to access a content

When digging into the structure of a punnet from the Explore place, you'll come across an URL pointing to the location of the binary file.

However there is quite a few ways of accessing a given content:

```
// As java file
File myFile = task.getManager()
                    .getPunnetContentFactory()
                    .getContentAsFile(myContent);



// As byte array
byte[] myBytes = task.getManager()
                    .getPunnetContentFactory()
                    .getContentAsByteArray(myContent);



// As URL
URL myURL = task.getManager()
                    .getPunnetContentFactory()
                    .getContentAsUrl(myContent);



// As RandomAccessInterface
RandomAccessInterface myRAI = task.getManager()
                    .getPunnetContentFactory()
                    .getContentAsRandomAccessInterface(myContent);
```

Mime-type

The content mime-type is a property usually added by the MimeTypeFinder task. However you could be willing to force it, which can be done like so:

```
myContent.setMimeType("the right mime-type");
```

This is basically what the MimeTypeFinder will do once the mime-type resolved from the content format.

To access this value, a regular java getter will do:

```
String myMimetype = myContent.getMimeType();
```

Properties

The contents in Fast2 also embed properties, for more closely related data.

```
Collection<Property> myProps = myContent.getProperties();

String myValue = myContent.getProperty(myName);

myContent.setProperty(myName,myValue);
```

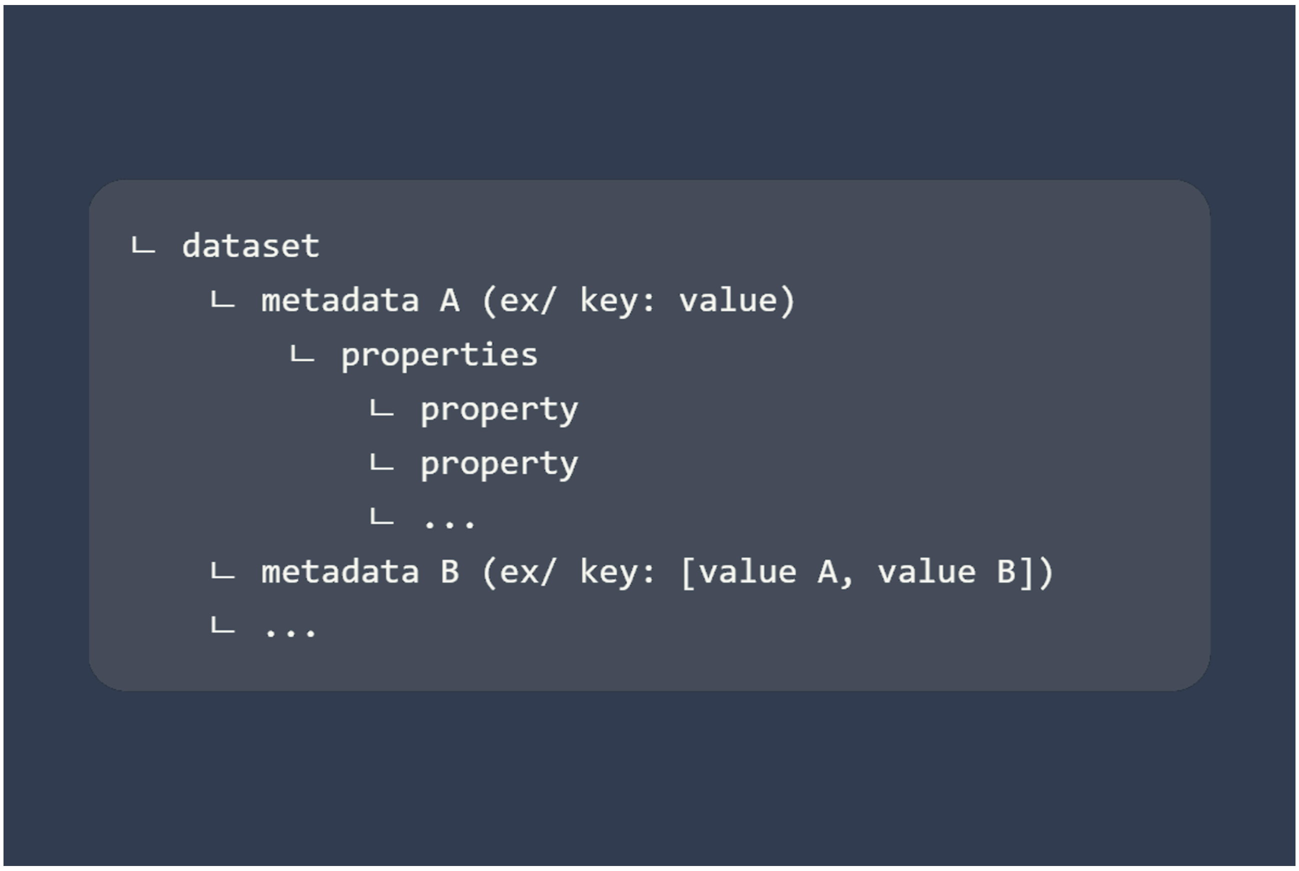
Sub-contents

Subcontents are just regular contents stored into a **ContentSet** attached to a content.

They can be both created/added and removed:

```
ContentSet subContents = myContent.getSubContents();

myContent.clearSubContents();
```

Cookbooks / Learn how to deal with datasets



```
∟ dataset
    ∟ metadata A (ex/ key: value)
        ∟ properties
            ∟ property
            ∟ property
            ∟ ...
    ∟ metadata B (ex/ key: [value A, value B])
    ∟ ...
```

Datasets are Fast2 objects which can be involved at different levels within the punnet.

They can be found on the

- punnets
- documents
- workflows

Datasets gather data which can be manipulated to store properties, and can be accessed as follows:

```
DataSet dataset = punnet.getDataSet();

DataSet dataset = document.getDataSet();

DataSet dataset = workflow.getDataSet();
```

Since datasets are just groups of data, understanding basic operations with data is primordial.

Data object

In Fast2, a data has 3 different informations:

- its name,
- its type (`String` or `int`)
- its value(s)

The following line retrieve the data as object :

```
Data data = dataset.getData(dataName);
```

Name

Getting the name of a data just goes like:

```
String dataName = data.getSymbolicName();
```

Type

If no type has been defined when the data has been created, the data type will be `null`.

However Fast2 will treat the value of the data as a regular `String`.

```
String dataType = data.getType();
```

Value(s)

When dealing with data, some can be single-valued while others can be multi-valued.

The returned object will differ accordingly.

```
String dataValue = data.getValue();

List<String> dataValues = data.getValues();
```

Data values can be added along the way, even when the data has already been created with a given value to begin with:

```
data.addValue(value);
```

Properties

A data can be dealt with just like any other object with properties.

Therefore, adding a property, removing it or getting it are just as simple as you would think:

```
data.setProperty(name, value);

String value = data.getProperty(name);

data.removeProperty(name);
```

Add data

Several ways of adding data to the dataset are available, depending on the type of value you are willing to store:

```
Data myData = myDataset.addData(name, "String", value);  // String
Data myData = myDataset.addData(name, null, value);

Data myData = myDataset.addData(name, "boolean", true);  // boolean

Data myData = myDataset.addData(name, type, 10);         // long,
int

Data myData = myDataset.addData(name, "String");              //
list or arrays of String
myData.getValues().addAll(Arrays.asList("a", "b", "c"));

document.getDataSet().addData("multivalued",
"String").getValues().addAll(Arrays.asList("value #1", "value
#2"));
```

Adding a new data with the same name as an already stored one, will result in overwriting the existing value with the new one.

Iterating through all data

Data mapping often requires to cover all data, no matter their name. To do so, the easiest way is to get them as a list:

```
List<Data> allData = myDataset.getData();

for(Data data : allData){
    // ...
}
```

Retrieve data value(s)

The following line retrieve the data as object :

```
Data data = dataset.getData(dataName);
```

Single-valued data

The dataset offers a shortcut to get the value(s) of any data:

```
// 1st way : via data object
String value = myDataSet.getData(dataName).getValue();

// or

// 2nd way : dataset shortcut
String value = myDataSet.getDataValue(dataName);
```

Multi-valued data

```
// 1st way : via data object
List<String> value = myDataSet.getData(dataName).getValues();

// or

// 2nd way : dataset shortcut
List<String> value = myDataSet.getDataValues(dataName);
```

Remove data

If the data has been found and could successfully be removed, the following method will return `TRUE`:

```
boolean removedSuccessfully = myDataset.removeData(name);
```

Check if data exists

Rely on this method to make sure not to overwrite any existing data, nor having a `DataNotFoundException` exception.

```
boolean exists = myDataset.hasData(name);
```

DataNotFound exception

When operations are performed on non-existing data, an exception of type `DataNotFoundException` is thrown.

Cookbooks / From ZIP to punnet

Dealing with archive files was never close to seldom in the past times, and surely won't ever be !

Let's guide you here through the common process for retrieving data, content, parsing files withing archive binaries, with our ETL tool.

🧐 Where do we come from ?

For the educational aspect of this topic, let us consider a folder gathering all the different archives, matching the following structure:

```
├─ ZIP archive/
|       ├─ metadata.json
|       └─ content.pdf
├─  ZIP archive/
|       ├─ metadata.json
|       └─ content.pdf
└─ ...
```

Each ZIP archive embeds a PDF content as flat file, alongside a JSON listing the metadata which we'll have to attach to the PDF document (before any injection-or-else phase).

The metadata contained in the JSON files are simply arranged like so:

```json
{
    "agency": "Agency_name",
    "customerNumber": "658217041",
    "contractNumber": "0121443-01",
    "operationDate": "20170523000000",
    "docType": "Bill",
    "fiscalYear": "2016",
    "treatmentType": "Archiving"
}
```

🤔 Where to go ?

At a glance, we are just 3 (major) steps away from having a PDF content in our punnet, with a basic dataset populated from the JSON metadata :

1. First we need to import these ZIP archives into Fast2
2. Then we need to dive into the ZIP "documents" one after the other,
3. And finally focus on the JSON content to parse and map the embedded properties.

Since our purpose is to dive into the ZIP files, we first need to gather them all with the LocalSource task, providing the parent folder where all these archives are currently being stored. The only required parameter is the path of the parent folder(s).

The second step is now to open these files up, in order to provide access to both the PDF content and the JSON file, revealing at the same time the metadata we are looking for. Such exposure can be achieved by using the DispatchingArchive task.

Once accessible, the JSON file can be parsed by the JSTransform task. Choice is yours regarding where to store the data found in the JSON content of our ZIP archive, here they will be added to the dataset of the document.

Not a big deal, right ? Let's then tackle this challenge right away, shall we !!

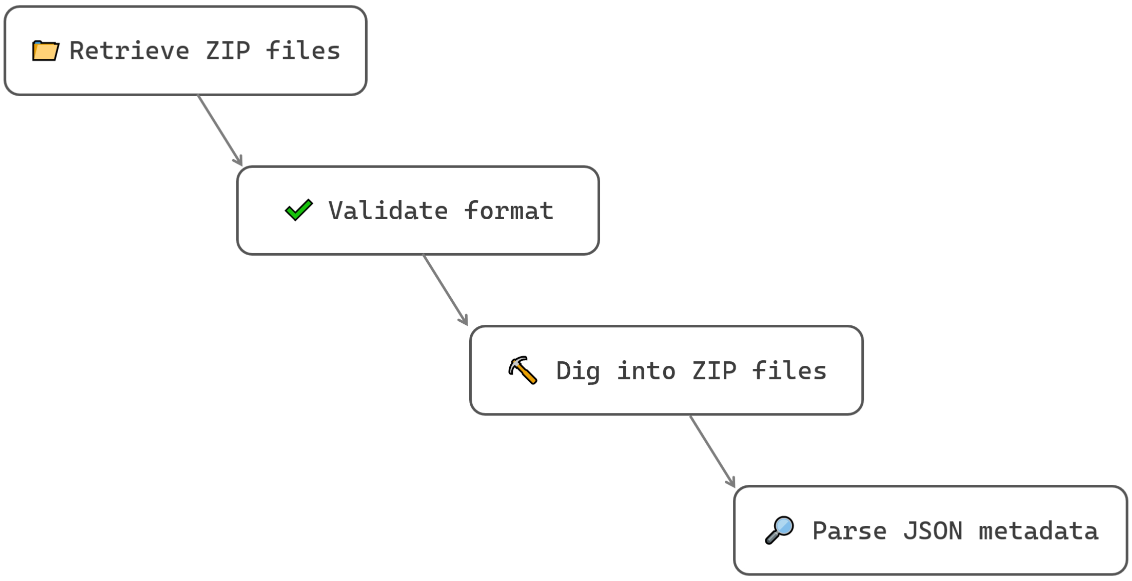
🚀 Way to go !

Inside Fast2, the map design is now pretty straightforward, given our ideas are rather clear in terms of ~~the mission~~ the overall order of the operations.

The map is even quite close to the 3 steps detailed earlier. The DispatchingArchive task just needs to be preceeded by a MimeTypeFinder task to highlight the archive format (here the ZIP extension is correct, but you could deal with archives without any extension, or mis-identified format).

That way, we end up with 4 tasks :

- LocalSource, to collect the documents from local storage,
- MimeTypeFinder, to assert the archive file format,
- DispatchingArchive, to open up the ZIPs,
- JSTransform, to focus on the JSONs and parse its content.



Although the configuration of the 3 first tasks can be easily guessed, the JSTransform may need some extra consideration: the focus on the JSON content of the ZIP content of the document of the punnet (see where we are heading, here ? 👀) plus the parsing phase all happen here.

The base script for this task (as it is presented here almost suits our need, except a minor tweaking to reach down the subcontent:

```
// First, get the document
var doc = punnet.getDocumentList().get(0);

// get JSON sub-content of ZIP-content of the document
var zipContent = doc.getContentSet().getContent().get(0);
var jsonContent = zipContent.getSubContents().get(1); // empirical
decision, PDF comes first
var bytes =
manager.getPunnetContentFactory().getContentAsByteArray(jsonContent);

var String = Java.type("java.lang.String");
var lines = new String(bytes);

// Then, parse this fragment as JSON
var jsonObject = JSON.parse(lines);

// Fetch all properties from existing fragment and create them as
Document data
for (pty in jsonObject) doc.getDataSet().addData(pty, "String",
jsonObject[pty]);
```

You might even bring it further, with content deletion or architectural changed of the punnet (e.g. bringing the PDF content as direct content in the punnet, while deleting the ZIP details as they are no longer required).

Head out now to the Run screen, start your campaign and just... enjoy !

At the latest stage of your workflow, the document dataset is filled with the properties found in the JSON and integrated as metadata.

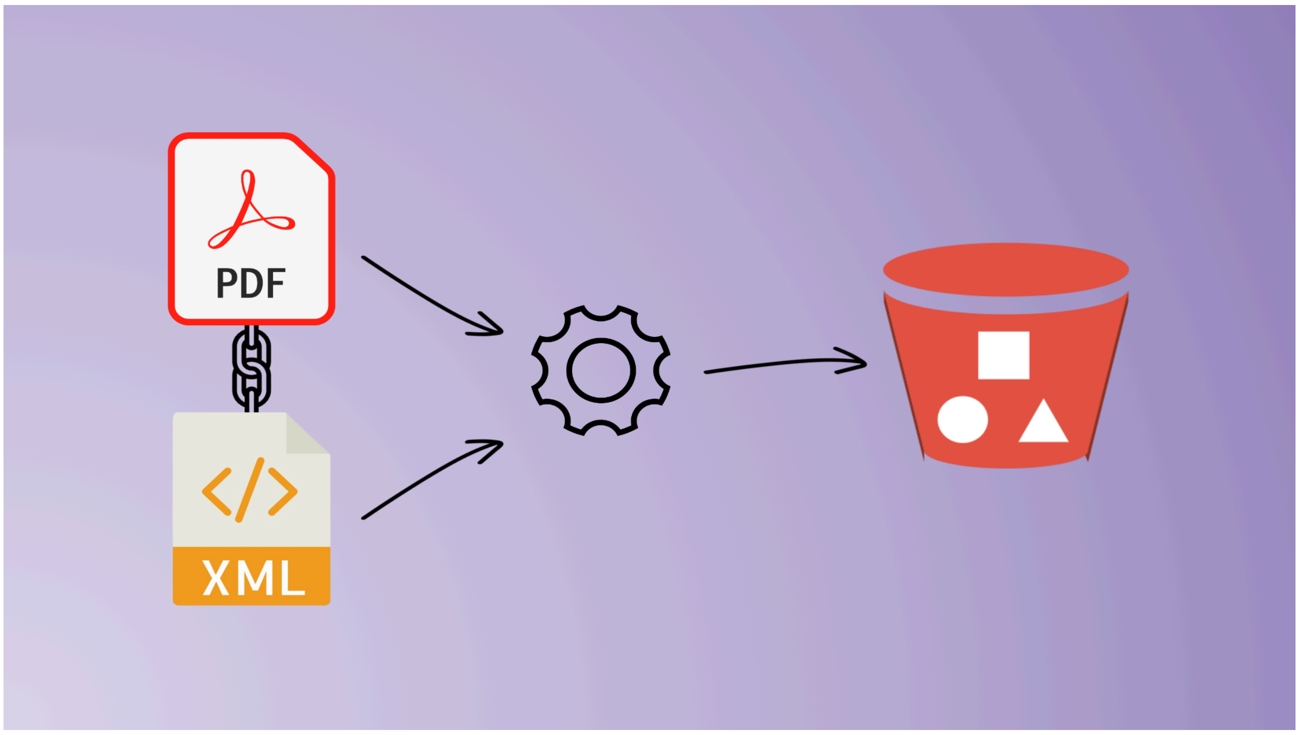
```
"data": {
    "absoluteParentPath": "G:/path/to/folder",
    "absolutePath": "G:/path/to/folder/ZIP_archive.zip",
    "agency": "Agency_name",
    "canExecute": true,
    "canRead": true,
    "canWrite": true,
    "contractNumber": "0121443-01",
    "customerNumber": "658217041",
    "docType": "Bill",
    "fileName": "ZIP_archive.zip",
    "fiscalYear": "2016",
    "lastModified": {
        "type": "Date",
        "value": "Tue Apr 05 11:50:08 CEST 2022"
    },
    "length": {
        "value": "82009"
    },
    "mimeType": "application/zip",
    "operationDate": "20170523000000",
    "treatmentType": "Archiving"
},
"documentId": "ZIP_archive.zip",
"folders": [...]
```

👏 Fast2: 1, ZIP: 0

Congrats, you've made it ! From ZIP archives as input, you now end up with a usable PDF file, ready for OCR or conversion, and its metadata.

If this use-case echoes your early needs, other tasks can be tied to this map to reach a higher level of complexity characteristic of real-world migration projects.

Cookbooks / Upload content and metadata in a S3 bucket



Injecting metadata into an S3 bucket needs to be done differently that what could be done with a regular content management system where a document is a set of contents and metadata. This constraint is even enforced when we upload documents into a SnowBall drive.

Let's quickly review here how storing both content and metadata in a S3 bucket can be achieved with Fast2.

🧐 Where do we come from ?

Let's suppose just extracted a document from a well known CMS solution, which created a regular punnet with a set of metadata and one content (ex/ a PDF file).

Injecting this document directly into a S3 bucket would just create a new binary file with the ID of the document as the name of the file in the bucket, and that would be all. Each and every metadata would have been lost in the way.

🤔 Where to go ?

To counteract this loss, we need to get Fast2 to add these metadata as a content too.

This can easily be done with the off-the-shelf tasks of Fast2, namely the PunnetSerializer task and the AlterDocumentContent task. Respectively, these tasks will create a new binary file with the metadata as XML inside, according to the Fast2 data model which you can find here.

In the end, we expect the bucket to have 2 contents for 1 document :

- 1 content in PDF, the original content of our document, whose name is the ID of the document with the correct extension (`.pdf`)
- 1 content in XML, filled with the metadata of the original document, whose name is the ID of the document with the correct extension (`.xml`)

🚀 Way to go !

Let's first create an XML file out of the metadata of the punnet, attach this created file to the document, and inject them later into our bucket.

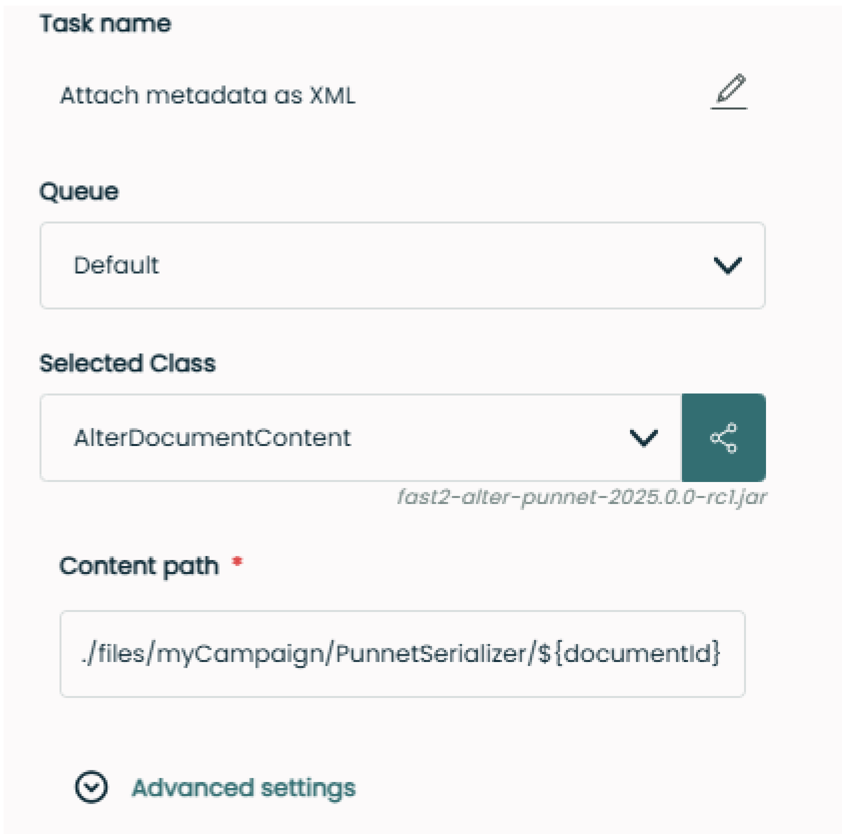
⚗️ From metadata to XML

Once our document fully extracted from the source CMS (via the tasks Source and ContentExtractor), we have the content and the metadata in a punnet.

The PunnetSerializer will convert the in-memory dataset record into XML format, in the default storage architecture (namely `$FAST2_HOME/files/<campaign>/<task>/<documentId>`). This path is the one we will have to provide in the next step of the workflow, which is the AlterDocumentContent task.

In our case, the pattern for the content to add is :

```
./files/myCampaign/PunnetSerializer/${documentId}
```



Task name

Attach metadata as XML

Queue

Default

Selected Class

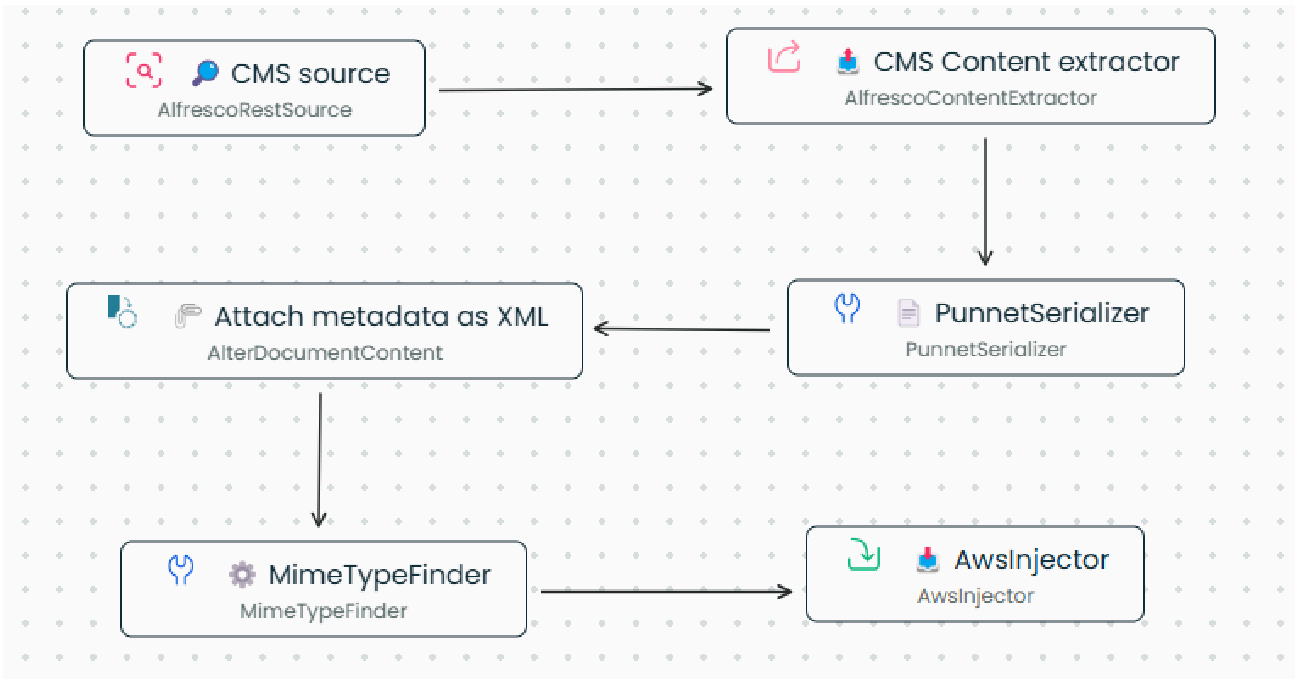
AlterDocumentContent

fast2-alter-punnet-2025.0.0-rc1.jar

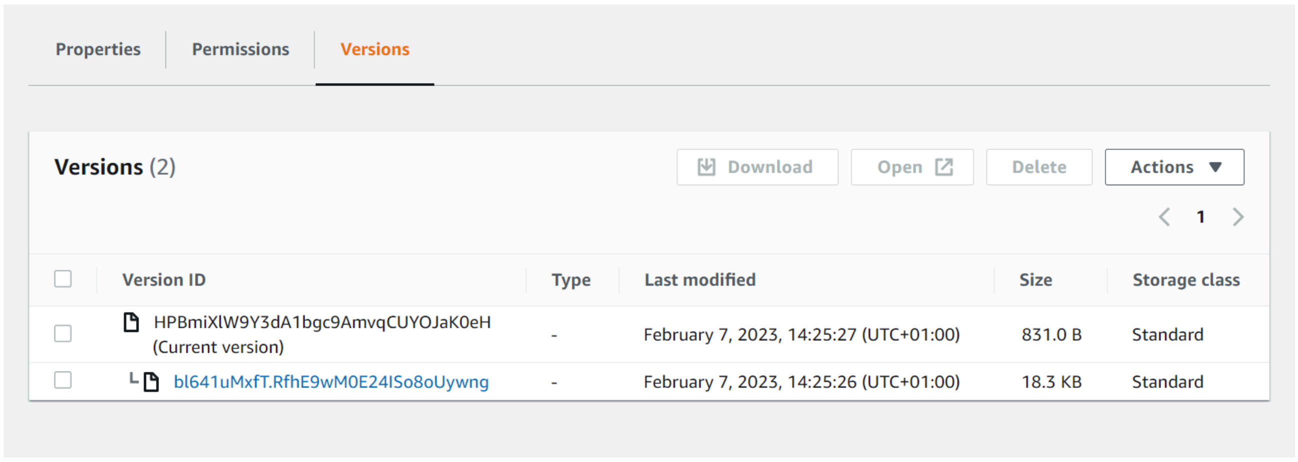
Content path *

./files/myCampaign/PunnetSerializer/${documentId}

Eventually, the migration workflow will end up looking like this :



However, if we run it, we see the following result in the destination S3 bucket :



Versions (2)

	Version ID	Type	Last modified	Size	Storage class
	HPBmiXlW9Y3dA1bgc9AmvqCUYOJaK0eH (Current version)	-	February 7, 2023, 14:25:27 (UTC+01:00)	831.0 B	Standard
	bl641uMxfT.RfhE9wM0E24ISo8oUywng	-	February 7, 2023, 14:25:26 (UTC+01:00)	18.3 KB	Standard

This can be explained by the fact that, when injecting, the S3 connector will upload both contents with the same name (which happens to be the documentId of the document). And as you might guess, 2 different documents with the same name induces the oldest one to be overwritten by the second.

We are just a tweak away of having these 2 contents alongside though, and that will need to happen in the AWSInjector connector.

✂️ Differentiate the 2 contents

Here, the tricky part is to identify the type of content we are dealing with.

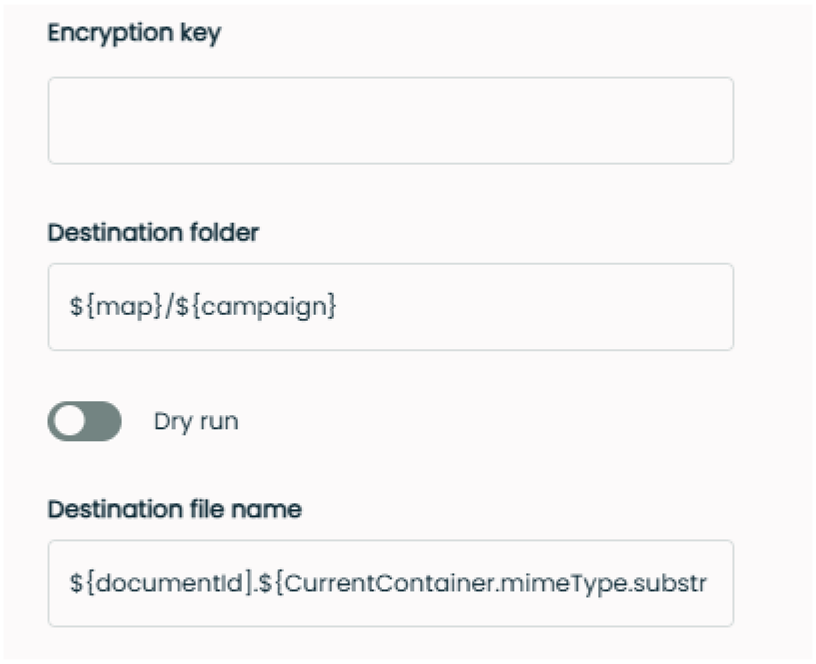
We know that the original document is a PDF and the PunnetSerializer generates an XML. So let's know Fast2 that information by creating the mimetype metadata on each content (as shown on the map screenshot earlier).

From now on, we can use a pattern to append the extension based on the content type value, accessing the `${CurrentContainer}` object.

The final pattern to use as "Destination file name" from the injector configuration is the following :

```
${documentId}.${CurrentContainer.mimeType.substring(CurrentContainer.
```

In the task configuration, the field to update is the "Destination file name" :



Encryption key

Destination folder

${map}/${campaign}

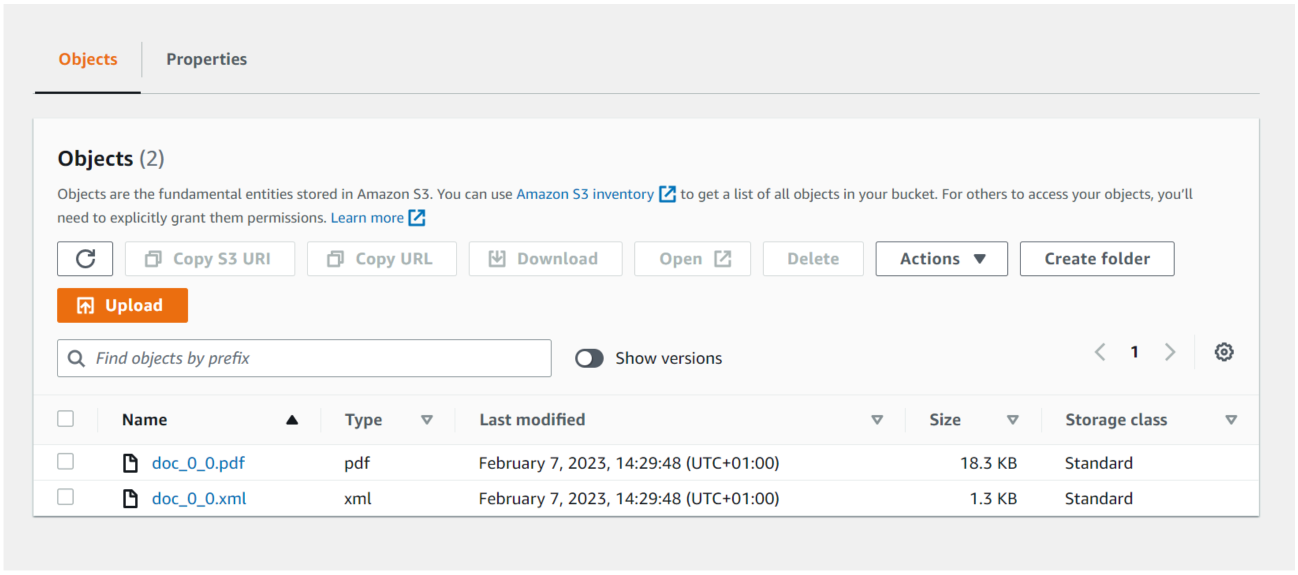
Dry run

Destination file name

${documentId}.${CurrentContainer.mimeType.substr

🏁 Result

This little edit will get Fast2 to build the final name on the fly for each content, and this is exactly what we needed to get the final result in our bucket :



Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

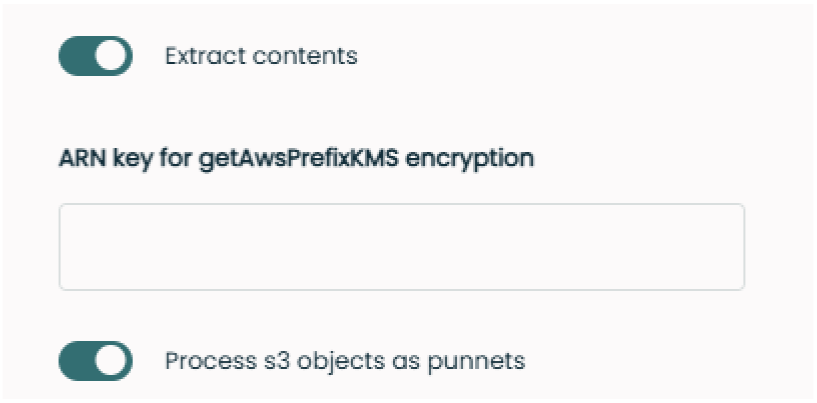
Name	Type	Last modified	Size	Storage class
doc_0_0.pdf	pdf	February 7, 2023, 14:29:48 (UTC+01:00)	18.3 KB	Standard
doc_0_0.xml	xml	February 7, 2023, 14:29:48 (UTC+01:00)	1.3 KB	Standard

👏 Let's wrap up

So now we have both contents and metadata with same name but different extension in our destination bucket, we could extract them easily with the AWSContentSource task by enabling the 2 options for such :

Process S3 objects as punnets (ie. metadata as XML and associated content)

Extract punnet contents (if required)

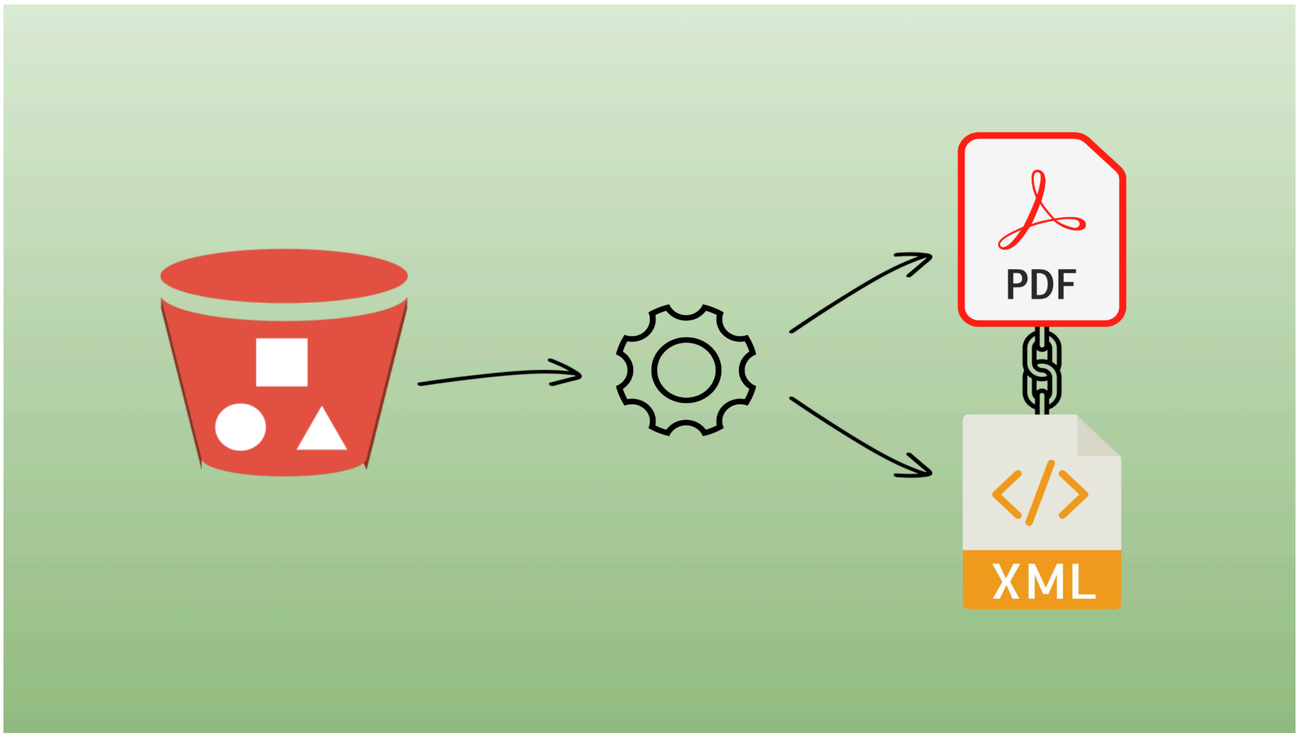


Extract contents

ARN key for getAwsPrefixKMS encryption

Process s3 objects as punnets

Cookbooks / Retrieve content and metadata from an S3 bucket



Extracting metadata from a S3 bucket needs to be done differently than what could be done with a regular content management system, because it is a storage space and not an ECM.

WARNING

"Prior to v2.10, Fast2 needed a few steps to add “manually” (with AlterDocumentProperties key and bucket information in the XML file, to then get corresponding PDF files."

Extracting metadata from a S3 bucket needs to be done differently than what could be done with a regular content management system, where a document is a set of contents and metadata. Indeed, S3 bucket is a storage space and not an ECM (we’ll get into that a little bit later).

Let’s quickly review here how extracting both content and metadata in a S3 bucket can be achieved with Fast2.

🧐 Where do we come from ?

For this case, let's supposed our documents have been injected in S3 bucket (direct sequel of the Upload content and metadata in a S3 bucket cookbook). This action splits them into pairs of individual files : contents (PDF) and matching metadata (XML) files, each sharing the same file name.

```
├─ Bucket S3/
│       ├─ folder A
│       │     ├─ document_A_1.xml
│       │     ├─ document_A_1.pdf
│       │     └─ ...
│       │
│       ├─ folder B
│       │     ├─ document_B_1.xml
│       │     ├─ document_B_1.pdf
│       │     └─ ...
│       └─...
└─ ...
```

Another constraint is that the PDF path information has been set into the XML file during the serialisation prior to the injection. So, once stored in the S3 bucket, it is not up-to-date.

🤔 Where to go ?

We want to end up with a regular Fast2 document composed with a content (PDF) and its metadata (parsed from the matching XML). Because of the old content path information (explained earlier), the content (PDF) will not be found in the XML information, inducing an incomplete document to be created.

🚀 Way to go !

First, we identify XML files. They contain all metadata, namely the PDF content we need to attach.

Then, we update the file extension : PDF and corresponding XML files have the same name.

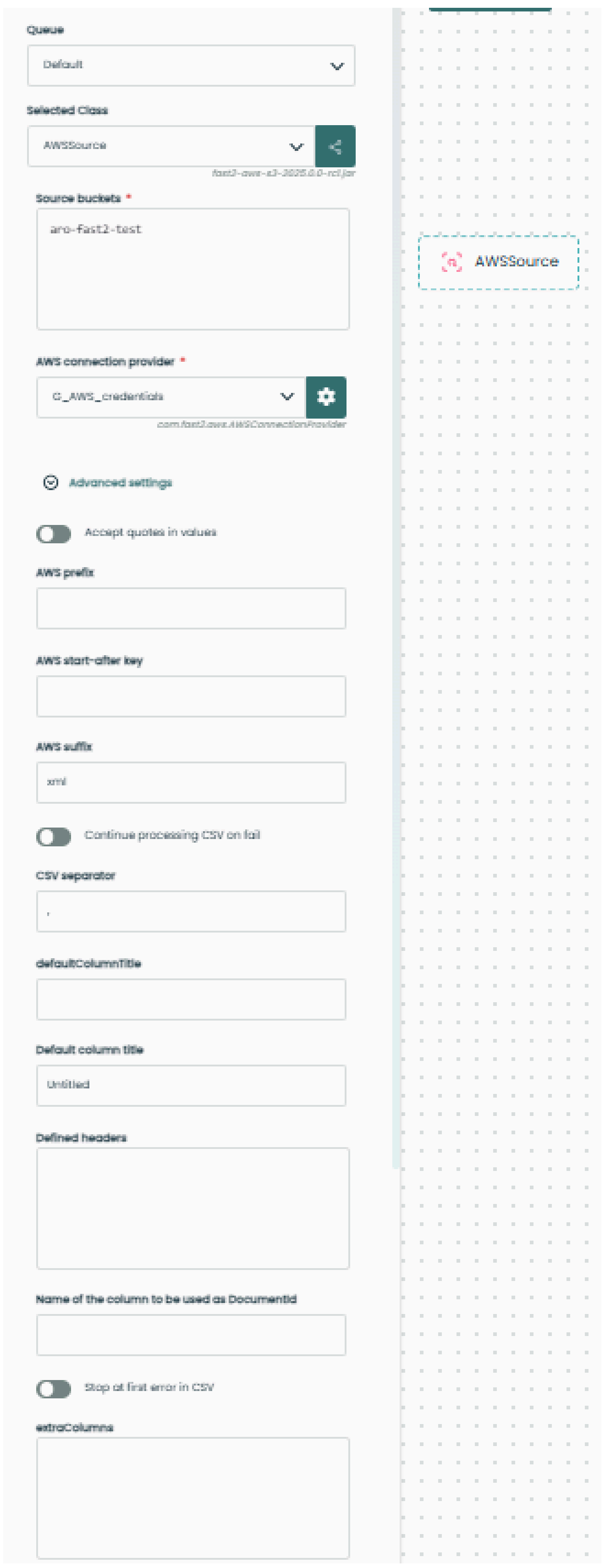
And finally, thanks to the source information, and XML metadata, we resolve the matching PDF content path, extract it from the bucket, and tie it to the Fast2 document.

🔎 Find content from metadata

In the AWSSource task , we extract only interested XML files because they contain metadata :

To only select punnet-formated XML correponding to the punnets, you will need to fill the AWS suffix field with : `xml`.

Optionally, you can also provide the concerned folder(s) in the Source folders if relevant.



Queue

Default

Selected Class

AWSSource

fast2-aws-s3-2025.0.0-rc1.jar

Source buckets *

aro-fast2-test

AWS connection provider *

G_AWS_credentials

com.fast2.aws.AWSConnectionProvider

Advanced settings

Accept quotes in values

AWS prefix

AWS start-after key

AWS suffix

xml

Continue processing CSV on fail

CSV separator

,

defaultColumnTitle

Default column title

Untitled

Defined headers

Name of the column to be used as DocumentId

Stop at first error in CSV

extraColumns

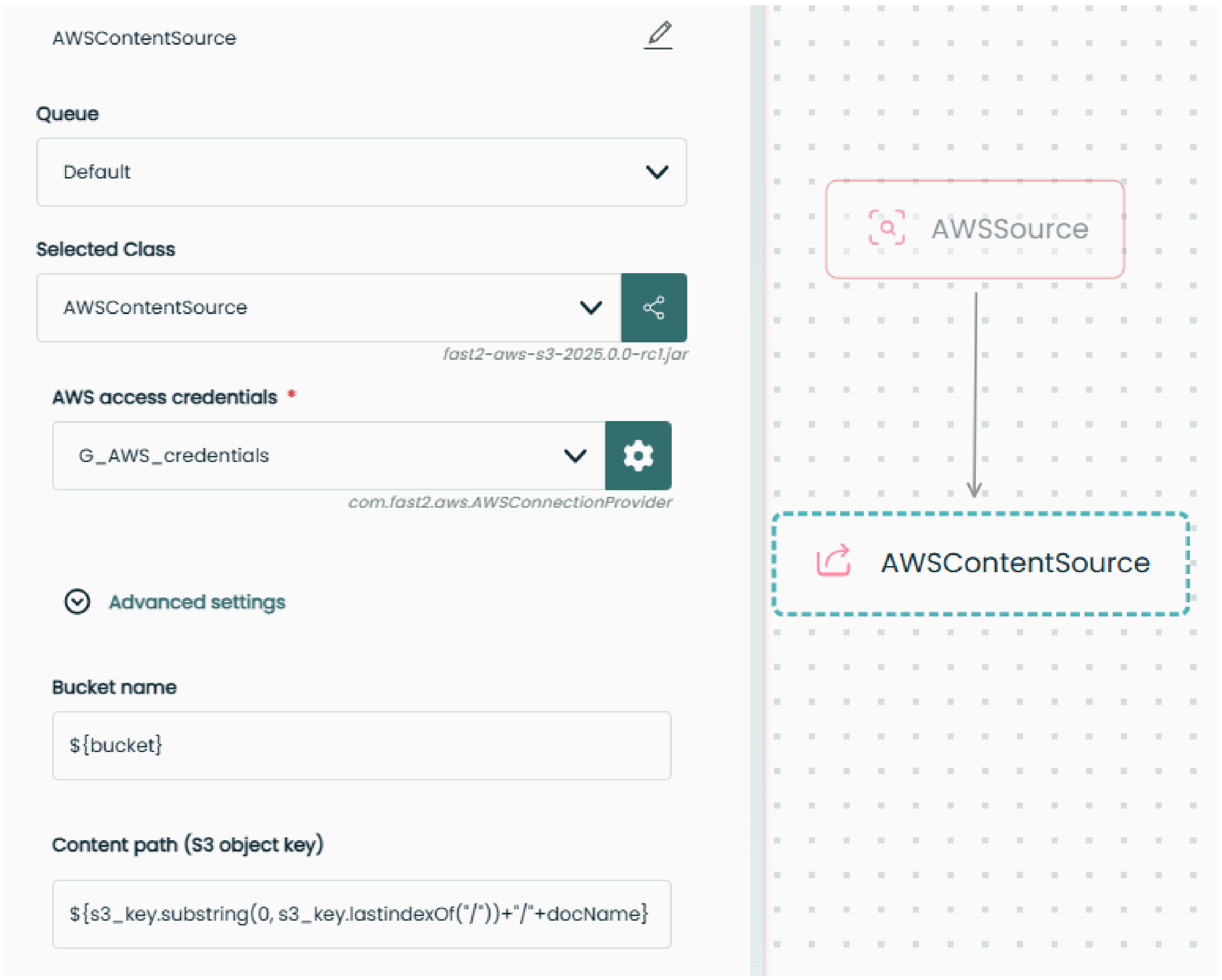
In the AWSContentSource task configuration, fields to fill are :

- Bucket name : `${bucket}`
- Content path (S3 object key) :

```
${s3_key.substring(0, s3_key.lastIndexOf(“/”)) + ”/” + docName}
```

By filling the S3 objecy key, the connector will on-the-fly build up the correct path where to look for the related content, and tie it to the punnet.

Fast2 extracts all metadata files present in our S3 bucket, following the key of the XML, as a punnet, and names them correctly.


AWSContentSource

Queue: Default

Selected Class: AWSContentSource
fast2-aws-s3-2025.0.0-rc1.jar

AWS access credentials *: G_AWS_credentials
com.fast2.aws.AWSConnectionProvider

Advanced settings

Bucket name: ${bucket}

Content path (S3 object key): ${s3_key.substring(0, s3_key.lastIndexOf("/"))+"/"+docName}

👏 Let's wrap up

We updated path information into XML files, which contain the XML-structured punnet. Then, we can attached corresponding PDF files.

We have now punnets containing linked content and metadata, which were sharing the same name in the bucket.

Next, we could process them through additional conversion or data transformation steps, or inject them into a secondary repository, you name it.

Cookbooks / CSV source : a step further

Ever wondered how Fast2 could help you dive into archives and come out with burried content and metadata? It actually is quite simple, as long as you have the right tools in your hands.

The CSVSource task has been designed to receive a CSV file as input.

Basic usage

With little to no configuration, each line represents one document with different values matching the column header. From a Fast2 standpoint, a CSV with following content :

```
header1,header2
value1_A,value2_A
value1_B,value2_B
```

will generate 2 documents :

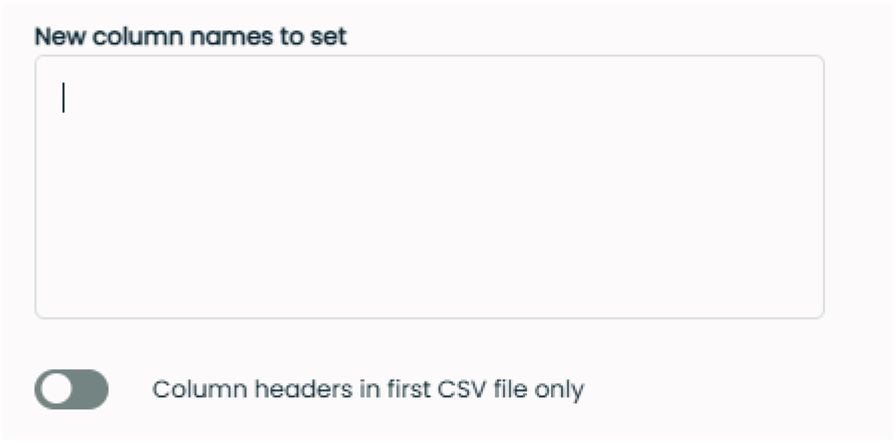
- The first document will have 2 data, `header1: value1_A` and `header2: value2_A`
- The second document will have 2 data, `header1: value1_B` and `header2: value2_B`

Although the default before resolved the data names from the 1st row (column headers), these names can be overwritten by the user, or even enriched.

Change data names

As for the first option (overwriting data names), the configuration needs to focus on the "New column names to set".

Enter each new header on a new line, making sure your input covers all the columns found in the CSV file.



With such a setting, Fast2 will map the data retrieved from the CSV directly under those new data names.

Example

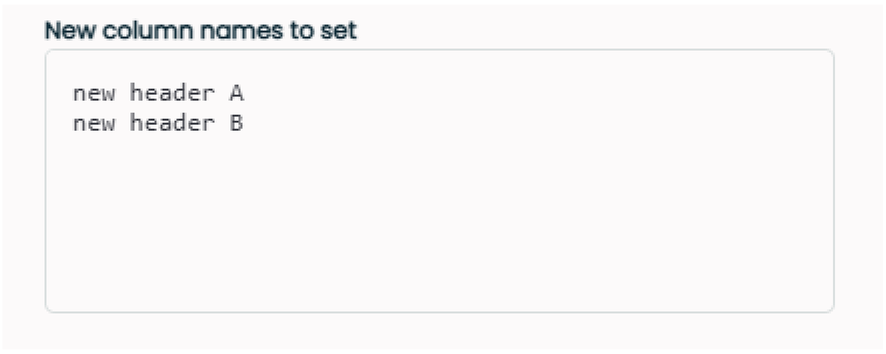
Let's consider processing a CSV file with the following content:

```
header1,header2
value1 ,value2
```

With the default settings, the document in Fast2 would have such dataset:

```
{
    "header1": "value1",
    "header2": "value2"
}
```

If the CSVSource task is configured as shown below,



```
new header A
new header B
```

the created document will only have a dataset looking like:

```
{
    "new header A": "value1",
    "new header B": "value2"
}
```

Fast2 will keep no trace of the old header names, generating a document with a dataset populated from the CSV file alongside new data names.

Of course this data name mapping could have been handled by an additional task, such as Drools or JSTransform (just to name a few).

But this CSV task here combines these 2 steps (of parsing and mapping) into a single one, lowering room for error and freeing the document of unnecessary information you'd not even have used.

Create extra columns based on existing data

This feature requires the configuration of the `extracolumns` option.

Syntaxe Enter one line per new data you intend to create.

Syntax goes as follows :

```
[variable]=[function]:[param1]:[param2]:[param3]:[param4]...
```

Rules

1. The separator is the character `:` (semi-colon).
2. Parameters have to striclty match the format `$<data_name>`. A data with the name "key" will be accessed under `$key`.
3. Parameters can use other params

   ```
   param1=stringLength:$keyA
   param2=substring:$param1:3:5
   ```

Supported functions

Function	Description
stringLength	length of param1
substring	substring of param1, from param2, during param3 characters
concat	concatenation of all params
ifeq	if param1 equals param2, then takes value of param3, otherwise param4

Example

We consider the following CSV content as input :

```
header1,header2,header3,header4
value1 ,value2 ,value3 ,this-is-the-value4
```

In the following examples, a new data with the name 'var_name' will be created with the value depending on the chosen option.

How to write it	Value of the
`var_name=stringLength:$header1`	Length (as in 'value1' unde
`var_name=substring:$header1:0:4`	First 4 chara under key `he`
`var_name=substring:$header4:4:7`	The 7 conse starting from (counting fro
`var_name=concat:OK/:$header1:/:$header2:_:$header3:.pdf`	Output : **ok/value1/v**
`var_name=concat:KO/:$header1:.pdf`	Output : **KO/**
`var_name=ifeq:$header1:18:$header2:$header3`	Value of 'hea to '18', so th

Cookbooks / Sort documents in a punnet

As we have seen before in the punnet structure, punnets can be composed with several different documents, each one of them embedding its own data and values.

Whether it be for the purpose of processing order of these documents or any other operation requiring these same documents to be sorted in a different order, the catalog provides sufficient tooling to tackle this challenge.

For the matter, we will consider having to sort the documents based on a `Date` data, which will go by the name of "creation_date".

🧐 Where do we come from ?

For the educational aspect of this topic, let us consider a punnet gathering several documents, all with the same data : `creation_date`, currently `String`-typed.

Our punnet would look like this:

```
├─ Punnet
|      ├─ document_1
|      |       └─ dataset
|      |              ├─ creation_date=03/3/2020 3:03:03 PM
|      |              └─ ...
|      ├─ document_2
|      |       └─ dataset
|      |              ├─ creation_date=01/1/2020 1:01:01 PM
|      |              └─ ...
|      ├─ document_3
|      |       └─ dataset
|      |              ├─ creation_date=02/2/2020 2:02:02 PM
|      |              └─ ...
|      └─ ...
└─ ...
```

As we can see, the correct order should be **document_2**, then **document_3** and finally **document_1**.

🤔 Where to go ?

At a glance, we are just couple steps away from sorting our documents : we need to go over all documents, dig into their dataset and retrieve the value of the sorting criteria.

However we need to go through a String-to-Date conversion so the sorting will be done correctly over datetimes values, instead of alphabetical values.

> **TIP**
>
> We will just reorganise all the documents within the punnet, they will be considered like tiles to rearrange, but we need not to mix their own data up (for obvious data integrity reasons).

🚀 Way to go !

The JSTransform task will be our hobbyhorse here since it offers the ability to handle the punnet at a pretty low cost in terms of performance and setup.

⚗️ JavaScript elaboration

You will need to add a new JSTransform task right after any task in your workflow have the punnet with all the documents (ex/ a ContentExtractor with all versions of a document extracted).

This JavaScript-ish task will be configured with the following script :

```
// Java types required for Java objects
var SimpleDateFormat = Java.type("java.text.SimpleDateFormat");
var Document = Java.type("com.fast2.model.punnet.Document");
var Collections = Java.type("java.util.Collections");

// Global parameters
var dataToSort = "creation_date";
var dateFormat = "MM/d/yyyy h:mm:ss aa";

var formatter = new SimpleDateFormat(dateFormat);

var compareByDate = function (doc1, doc2) {
    try {
        return formatter
            .parse(doc1.getDataSet().getDataValue(dataToSort))

.compareTo(formatter.parse(doc2.getDataSet().getDataValue(dataToSort)
    } catch (e) {}
    return 0;
};

Collections.sort(punnet.getDocuments(), compareByDate);
```

Explanations :

- L7 : the data which you want to sort, in our case the `creation_date`
- L8 : the date format we deduced from the `String` value of the previous data
- L12-L17 : we need to parse the value as date, to compare the date as so instead of regular `String` values (which could be too approximative)

The output of this task will be the same documents in the punnet, just ordered by creation date ascending.

Head out now to the Run screen, and start your campaign.

🏁 Result

At the latest stage of your workflow, the document dataset is filled with the properties found in the JSON and integrated as metadata.

```
├─ Punnet
|      ├─ document_2
|      |       └─ dataset
|      |             ├─ creation_date=01/1/2020 1:01:01 PM
|      |             └─ ...
|      ├─ document_3
|      |       └─ dataset
|      |             ├─ creation_date=02/2/2020 2:02:02 PM
|      |             └─ ...
|      ├─ document_4
|      |       └─ dataset
|      |             ├─ creation_date=03/3/2020 3:03:03 PM
|      |             └─ ...
|      └─ ...
└─ ...
```

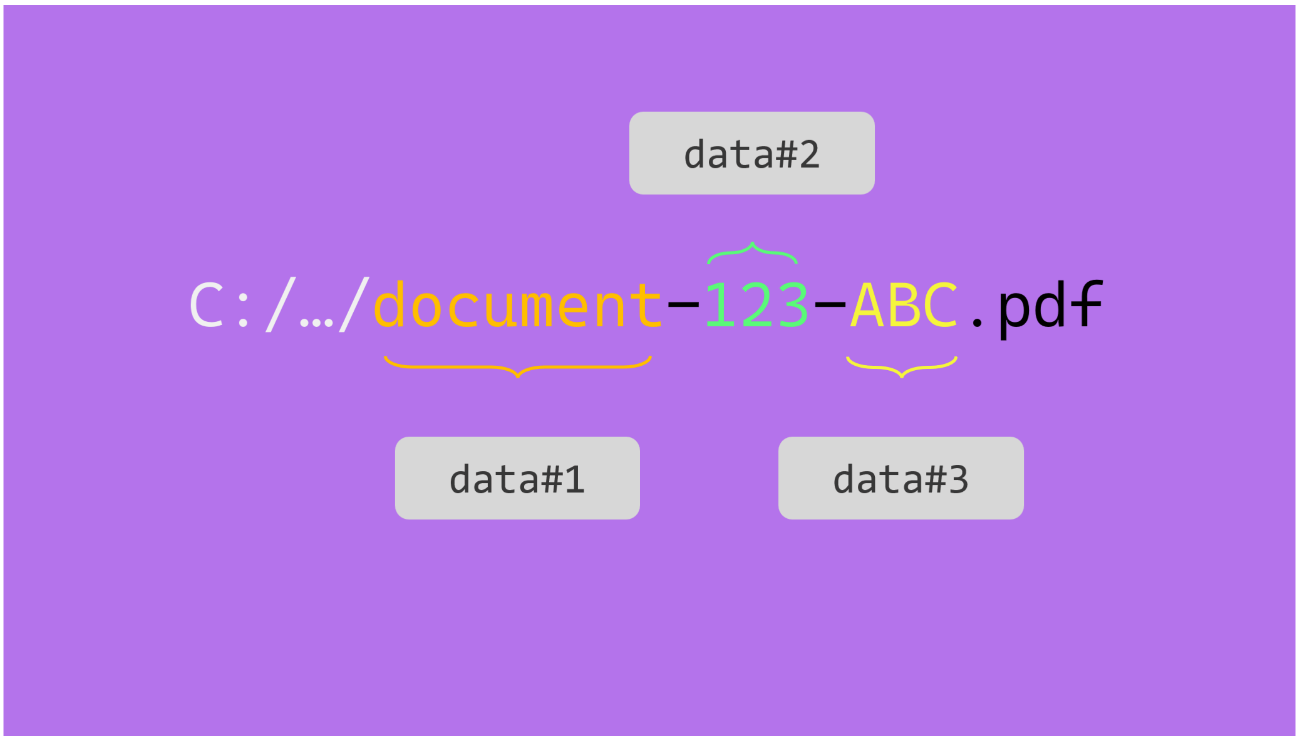
👏 Let's sum up

We can bring this scenario further by sorting based on 2 or more data, regular `int` values or else.

Since its just sorting, you might want to sort the documents in the reverse order. In this example, we left the default behavior (ascending), but a minor tweak to the previous script and you'll be good to go !

If this use-case echoes your early needs, other tasks can be tied to this map to reach a higher level of complexity characteristic of real-world migration projects.

Cookbooks / Add data from file name



Times will happen when you will not be able to rely on side-file metadata documents to map onto the documents you are migrating. The data will be concatenated into the file name.

Fortunatelly with Fast2, there is still a possibility to parse this file name and pull out the required metadata. Last step would be to tight them down into the document dataset.

:face_with_raised_eyebrow: Where do we come from ?

For the educational aspect of this topic, let us consider a folder gathering several documents, all with the same format : `<document-type>-<data1>-<data2>`.

Our folder looks like this:

```
├─ folder-to-extract
|       ├─ contract-123-ABC.pdf
|       ├─ contract-346-DEF.pdf
|       |
|       ├─ bill-123-ABC.pdf
|       ├─ bill-346-DEF.pdf
|       |
|       ├─ contract-123-ABC.pdf
|       ├─ contract-346-DEF.pdf
|       |
|       ├─ draft-123-ABC.pdf
|       ├─ draft-346-DEF.pdf
|       └─ ...
└─ ...
```

🤔 Where to go ?

At a glance, we are just 3 (major) steps away from having a PDF content in our punnet, with a basic dataset populated from the JSON metadata :

1. Scan the parent folder and list all the documents with names to map,
2. Get the document path, and isolate the file name
3. Parse the file name and attach the metadata to the dataset. For this example, data will be mapped onto the document dataset.

🚀 Way to go !

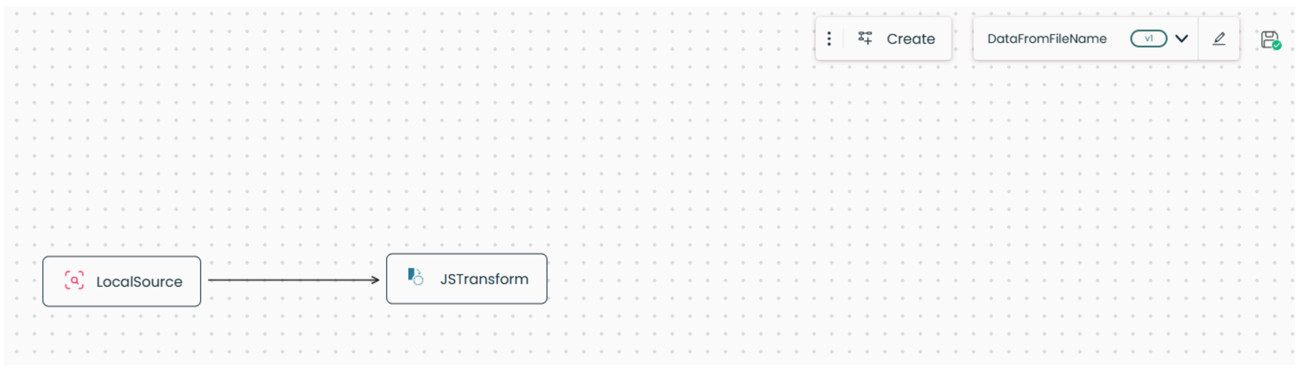
Inside Fast2, the map design is now pretty straightforward, given our ideas are rather clear in terms of the overall order of the operations.

The map is even quite close to the 3 steps detailed earlier. The LocalSource task just needs to be given the path of the folders to deal with. This task will also identify the file name and attach the metadata to the document dataset.

Then the JSTranform will retrieve the corresponding document path, and carry on with the data mapping.

That way, we end up with 4 tasks :

- LocalSource, to collect the documents from local storage,
- JSTranform, whose role will be to :
  1️⃣ parse the file name
  2️⃣ add the data to the dataset



🧪 JavaScript elaboration

Although the configuration of the first task can be easily guessed, the JSTranform final resulting script should look something like this :

```
punnet.getDocuments().forEach(function (doc) {
    // (1)
    var filenameWithoutExtension =
doc.getDataSet().getData("fileName").getValue().split(".")[0];

    // (2)
    var data = filenameWithoutExtension.split("-");

    // (3)
    doc.getDataSet().addData("document-type", "String", data[0]);
    doc.getDataSet().addData("data1", "String", data[1]);
    doc.getDataSet().addData("data2", "String", data[2]);
});
```

1. 1️⃣ Get the filename, and remove the extension
2. 2️⃣ Parse the filename with the separator character
3. 3️⃣ Attach the data

Head out now to the Run screen, start your campaign and just... enjoy !

🏁 Result

At the latest stage of your workflow, the document dataset is filled with the properties found in the JSON and integrated as metadata.

```json
{
        "punnetId": "document-123-ABC.pdf#1",
        "documents": [
                {
                        "documentId": "document-123-ABC.pdf",
                        "data": {
                                "absolutePath": "C:\\samples\\document-123-ABC.pdf",
                                "fileName": "document-123-ABC.pdf",
                                "absoluteParentPath": "C:\\samples",
                                "length": {
                                        "value": "18700"
                                },
                                "lastModified": {
                                        "value": "Mon Dec 27 14:10:47 CET 2021",

                                        "type": "Date"
                                },
                                "document-type": "document",
                                "data1": "123",
                                "data2": "ABC"
                        },
                        "contents": {
                                "url": "C:\\samples\\document-123-ABC.pdf"
                        },
                        "folders": [...]
                }
        ]
}
```

👏 Let's sum up

We can bring this scenario further by mapping data from the parent folder(s). We would just need the document path, which can be retrieved easily, as explained in the advanced section of how to handle the JS Tranform task.

For a OS-proofed script (Linux or Windows have their own subtleties when it comes to paths), you may need to make sure the parsing is done correctly, by standardizing the folder-architecture-related special characters from the Windows `\` to a regular `/`.

If this use-case echoes your early needs, other tasks can be tied to this map to reach a higher level of complexity characteristic of real-world migration projects.

Cookbooks / Interact with a SQL database

WARNING

This page has been moved to the Knowledge-Base at 🔗JDBC : How to link Fast2 and SQL DB

API documentation

Welcome to the **Fast2 API documentation**.
This section provides a complete reference for interacting programmatically with Fast2 using its RESTful web services.

The API allows you to:

- Trigger and monitor migrations
- Manage configurations, workers, and migration jobs
- Query results and punnets data
- And many more...

All endpoints are documented with:

- Path and query parameters
- Request and response schemas
- Example calls in **cURL**

Before using the API, make sure you have:

- A valid authentication token (see Authentication)
- Network access to the Fast2 server

Base URL: `http://localhost:1789`

Note: The interactive Swagger UI for Fast2 is available at:
http://localhost:1789/swagger-ui/index.html

Authentication

- **Bearer JWT** via `Authorization: Bearer <token>` (Bearer Token)

> All examples include cURL requests. Replace placeholders with real values.

Tags

- **Catalog API** — Endpoint to retrieve catalog tasks
- **Job API** — Endpoint for managing jobs
- **Worker API** — API for managing workers
- **User API** — Endpoint for managing users
- **Shared Objects API** — Endpoint for managing shared objects
- **Email API** — Endpoint for managing emails
- **Map API** — Endpoint for managing maps
- **Queue API** — Endpoint for managing queues
- **Punnet API** — Endpoint to retrieve punnets
- **Broker API** — Endpoint for broker-worker communication
- **Campaign API** — Endpoint for managing campaigns
- **Authentication API** — Endpoint for managing authentications

---

Authentication API

Endpoint for managing authentications

changePassword

`POST /auth/change-password`

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`currentPassword`	string	no	
`newPassword`	string	no	
`newPasswordConfirmation`	string	no	

Body format:

```
{
  "currentPassword": "string",
  "newPassword": "string",
  "newPasswordConfirmation": "string"
}
```

Responses

Status	Content-Type	Schema	Description
200	/	object	OK

cURL example

```
curl -X POST 'http://localhost:1789/auth/change-password' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "currentPassword": "string",
  "newPassword": "string",
  "newPasswordConfirmation": "string"
}'
```

isAuthenticated

`GET /auth/is-authenticated`

Responses

Status	Content-Type	Schema	Description
200	/	boolean	OK

cURL example

```
curl -X GET 'http://localhost:1789/auth/is-authenticated' \
  -H 'Authorization: Bearer <token>' \
```

getLockTimeDuration

`GET /auth/lock-time-duration`

Responses

Status	Content-Type	Schema	Description
200	/	integer(int32)	OK

cURL example

```
curl -X GET 'http://localhost:1789/auth/lock-time-duration' \
  -H 'Authorization: Bearer <token>' \
```

authenticate

`POST /auth/login`

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`email`	string	no	
`password`	string	no	

Body format:

```
{
  "email": "string",
  "password": "string"
}
```

Responses

Status	Content-Type	Schema	Description
200	/	object	OK

cURL example

```
curl -X POST 'http://localhost:1789/auth/login' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "string",
  "password": "string"
}'
```

getMaxFailedAttempts

`GET /auth/max-failed-attempts`

Responses

Status	Content-Type	Schema	Description
200	/	integer(int32)	OK

cURL example

```
curl -X GET 'http://localhost:1789/auth/max-failed-attempts' \
  -H 'Authorization: Bearer <token>' \
```

getPublicKey

`GET /auth/public-key`

Responses

Status	Content-Type	Schema	Description
200	/	string	OK

cURL example

```
curl -X GET 'http://localhost:1789/auth/public-key' \
  -H 'Authorization: Bearer <token>' \
```

refreshToken

`POST /auth/refresh-token`

Responses

Status	Content-Type	Schema	Description
200	/	object	OK

cURL example

```
curl -X POST 'http://localhost:1789/auth/refresh-token' \
  -H 'Authorization: Bearer <token>' \
```

getRemainingAttempts

`GET /auth/remaining-attempts`

Parameters

Name	In	Required	Type	Description
`email`	query	yes	string	

Responses

Status	Content-Type	Schema	Description
200	/	integer(int32)	OK

cURL example

```
curl -X GET 'http://localhost:1789/auth/remaining-attempts' \
  -H 'Authorization: Bearer <token>' \
```

getRemainingLockTime

`GET /auth/remaining-lock-time`

Parameters

Name	In	Required	Type	Description
`email`	query	yes	string	

Responses

Status	Content-Type	Schema	Description
200	/	integer(int64)	OK

cURL example

```
curl -X GET 'http://localhost:1789/auth/remaining-lock-time' \
  -H 'Authorization: Bearer <token>' \
```

resetPassword

`POST /auth/reset-password`

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`targetUser`	string	no	
`newPassword`	string	no	
`newPasswordConfirmation`	string	no	

Body format:

```
{
  "targetUser": "string",
  "newPassword": "string",
  "newPasswordConfirmation": "string"
}
```

Responses

Status	Content-Type	Schema	Description
200	/	object	OK

cURL example

```
curl -X POST 'http://localhost:1789/auth/reset-password' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "targetUser": "string",
  "newPassword": "string",
  "newPasswordConfirmation": "string"
}'
```

Broker API

Endpoint for broker-worker communication

Delete any content set in broker files directory

`DELETE /broker/contents`

Parameters

Name	In	Required	Type	Description
`path`	query	yes	string	

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/broker/contents' \
  -H 'Authorization: Bearer <token>' \
```

Download any content set in broker files directory

`GET /broker/contents`

Parameters

Name	In	Required	Type	Description
`path`	query	yes	string	

Responses

Status	Content-Type	Schema	Description
200	application/json	string(binary)	OK

cURL example

```
curl -X GET 'http://localhost:1789/broker/contents' \
  -H 'Authorization: Bearer <token>' \
```

Download logs produced by the broker

`GET /broker/download-broker-logs`

Responses

Status	Content-Type	Schema	Description
200	application/json	array[string(byte)]	OK

cURL example

```
curl -X GET 'http://localhost:1789/broker/download-broker-logs' \
  -H 'Authorization: Bearer <token>' \
```

Campaign API

Endpoint for managing campaigns

Delete campaigns by names

`DELETE /campaigns/delete-by-names`

Deletes campaigns that match the list of specified campaign names and map version. Returns a multi-status response indicating the success or failure of deleting each campaign

Parameters

Name	In	Required	Type	Description
`campaigns`	query	yes	array[object]	Campaign names to retrieve
`mapVersionSerieId`	query	no	string	Map version series id to filter campaigns

Responses

Status	Content-Type	Schema	Description
207	application/json	object	Server failed to delete campaigns
200	application/json	object	Successfully deleted campaigns

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters

cURL example

```
curl -X DELETE 'http://localhost:1789/campaigns/delete-by-names' \
  -H 'Authorization: Bearer <token>' \
```

Delete campaigns by pattern

`DELETE /campaigns/delete-by-pattern`

Deletes campaigns that match the specified name pattern and map version. If no name pattern is provided, all campaigns will be selected. Returns a multi-status response indicating the success or failure of deleting each campaign

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter campaign names
`mapVersionSerieId`	query	no	string	Map version series id to filter campaigns

Responses

Status	Content-Type	Schema	Description
207	application/json	object	Server failed to delete campaigns
200	application/json	object	Successfully deleted campaigns
400	—	—	Invalid request parameters

cURL example

```
curl -X DELETE 'http://localhost:1789/campaigns/delete-by-pattern'
\
  -H 'Authorization: Bearer <token>' \
```

Download campaign exceptions

`GET /campaigns/download-exceptions`

Downloads all exceptions thrown during a specific campaign

Parameters

Name	In	Required	Type	Description
`campaigns`	query	yes	array[string]	
`mapIds`	query	yes	array[string]	

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully downloaded exceptions
500	—	—	Server failed to download exceptions
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/download-exceptions' \
  -H 'Authorization: Bearer <token>' \
```

Get campaigns dto information by names

`GET /campaigns/dto/search-by-names`

Retrieves campaigns that match the list of specified campaign names, map version and map id

Parameters

Name	In	Required	Type	Description
`campaigns`	query	yes	array[object]	Campaign names to retrieve
`mapVersionSerieId`	query	no	string	Map version series id to

Name	In	Required	Type	Description
				filter campaigns
`mapId`	query	no	string	Map id to filter campaigns
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
500	—	—	Failed to retrieve campaigns
200	application/json	object	Successfully retrieved campaigns
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/dto/search-by-names' \
  -H 'Authorization: Bearer <token>' \
```

Get campaigns dto information by pattern

`GET /campaigns/dto/search-by-pattern`

Retrieves a list of campaigns dto that match the specified name pattern, map version and map id. If no name pattern is provided, all campaigns will be selected

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter campaign names
`mapVersionSerieId`	query	no	string	Map version series id to filter campaigns
`mapId`	query	no	string	Map id to filter campaigns
`status`	query	no	string	Campaign status to filter campaigns
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve campaigns
200	application/json	object	Successfully retrieved campaigns
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/dto/search-by-pattern'
\
  -H 'Authorization: Bearer <token>' \
```

Get campaign dto information

`GET /campaigns/dto/{campaign}`

Retrieves a campaign dto from its name

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name to retrieve

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully retrieved campaign
400	—	—	Invalid request parameters
404	—	—	Campaign not found
500	—	—	Server failed to retrieve campaign

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/dto/<campaign>' \
  -H 'Authorization: Bearer <token>' \
```

Get campaigns information by name

`GET /campaigns/search-by-names`

Retrieves campaigns that match the list of specified campaign names, map version and map id

Parameters

Name	In	Required	Type	Description
`campaigns`	query	yes	array[object]	Campaign names to stop
`mapVersionSerieId`	query	no	string	Map version series id to filter campaigns
`mapId`	query	no	string	Map id to filter campaigns
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully retrieved campaigns
500	—	—	Server failed to retrieve campaigns
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/search-by-names' \
  -H 'Authorization: Bearer <token>' \
```

Get campaigns information by pattern

`GET /campaigns/search-by-pattern`

Retrieves campaigns that match the specified name pattern, map version and map id. If no name pattern is provided, all campaigns will be selected

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter campaign names
`mapVersionSerieId`	query	no	string	Map version series id to filter campaigns

Name	In	Required	Type	Description
`mapId`	query	no	string	Map id to filter campaigns
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully retrieved campaigns
500	—	—	Server failed to retrieve campaigns
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/search-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

Stop campaigns by names

`POST /campaigns/stop-by-names`

Stops campaigns that match the specified list of campaign names and map version. Returns a multi-status response indicating the success or failure of stopping each campaign

Parameters

Name	In	Required	Type	Description
`campaigns`	query	yes	array[object]	Campaign names to stop
`mapVersionSerieId`	query	no	string	Map version series id to filter campaigns

Responses

Status	Content-Type	Schema	Description
207	application/json	object	Server failed to stop campaign
200	application/json	object	Successfully stopped campaign
400	—	—	Invalid request parameters

cURL example

```
curl -X POST 'http://localhost:1789/campaigns/stop-by-names' \
  -H 'Authorization: Bearer <token>' \
```

Stop campaigns by pattern

`POST /campaigns/stop-by-pattern`

Stops campaigns that match the specified name pattern and map version. If no name pattern is provided, all campaigns will be selected. Returns a multi-status response indicating the success or failure of stopping each campaign

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter campaign names
`mapVersionSerieId`	query	no	string	Map version series id to filter campaigns

Responses

Status	Content-Type	Schema	Description
207	application/json	object	Server failed to stop campaign
200	application/json	object	Successfully stopped campaign
400	—	—	Invalid request parameters

cURL example

```
curl -X POST 'http://localhost:1789/campaigns/stop-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

Delete a campaign

`DELETE /campaigns/{campaign}`

Deletes a campaign from its name

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name to delete

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully deleted campaign
400	—	—	Invalid request parameters
404	—	—	Campaign not found
500	—	—	Server failed to delete the campaign

cURL example

```
curl -X DELETE 'http://localhost:1789/campaigns/<campaign>' \
  -H 'Authorization: Bearer <token>' \
```

deleteCampaignParameter

`DELETE /campaigns/{campaign}/parameter/{campaignParameter}`

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name
`campaignParameter`	path	yes	string	Campaign parameter key

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X DELETE
'http://localhost:1789/campaigns/<campaign>/parameter/<campaignParame
\
  -H 'Authorization: Bearer <token>' \
```

deleteAllCampaignParameters

`DELETE /campaigns/{campaign}/parameters`

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X DELETE
'http://localhost:1789/campaigns/<campaign>/parameters' \
  -H 'Authorization: Bearer <token>' \
```

getCampaignParameters

`GET /campaigns/{campaign}/parameters`

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/<campaign>/parameters'
\
  -H 'Authorization: Bearer <token>' \
```

createCampaignParameters

`POST /campaigns/{campaign}/parameters`

Parameters

Name	In	Required	Type	Description
`mapId`	query	yes	object	Map Id
`campaign`	path	yes	object	Campaign name

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`empty`	boolean	no	

Body format:

```
{
  "empty": true
}
```

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X POST
'http://localhost:1789/campaigns/<campaign>/parameters' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "empty": true
}'
```

Resume a campaign

`POST /campaigns/{campaign}/resume`

Resumes a campaign from its name. Only stopped campaigns can be resumed

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name

Responses

Status	Content-Type	Schema	Description
400	—	—	Campaign not stopped
404	—	—	Campaign not found
500	—	—	Server failed to resume campaign
200	application/json	object	Successfully resumed campaign

cURL example

```
curl -X POST 'http://localhost:1789/campaigns/<campaign>/resume' \
  -H 'Authorization: Bearer <token>' \
```

Retry punnets

`POST /campaigns/{campaign}/retry-punnets`

Retry punnets in any step for a specific campaign. You can filter the punnets to retry by status and metadata values

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name of the punnets to retry
`mapId`	query	yes	object	Map id related to the campaign
`stepId`	query	yes	object	Step id containing the punnets to retry
`status`	query	no	array[string]	Status to filter punnets to retry

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retry punnets
400	—	—	Invalid request parameters
200	—	—	Successfully retried campaign

cURL example

```
curl -X POST 'http://localhost:1789/campaigns/<campaign>/retry-
punnets' \
  -H 'Authorization: Bearer <token>' \
```

Start a campaign

`POST /campaigns/{campaign}/start`

Starts a campaign from its name and a map id. Can be a new campaign or a rerun

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name
`mapId`	query	yes	object	Map Id
`newCampaign`	query	no	boolean	New campaign

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
404	—	—	Map not found
200	application/json	object	Successfully started campaign
500	—	—	Server failed to start campaign

cURL example

```
curl -X POST 'http://localhost:1789/campaigns/<campaign>/start' \
  -H 'Authorization: Bearer <token>' \
```

Get campaign stats

`GET /campaigns/{campaign}/stats`

Retrieves stats of specified campaign

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name to retrieve

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve campaign stats
200	application/json	object	Successfully retrieved campaign stats
400	—	—	Invalid request parameters
404	—	—	Campaign or stats not found

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/<campaign>/stats' \
  -H 'Authorization: Bearer <token>' \
```

Get campaign status

`GET /campaigns/{campaign}/status`

Retrieves status of specified campaign

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name to retrieve

Responses

Status	Content-Type	Schema	Description
404	—	—	Campaign or status not found
200	application/json	string	Successfully retrieved campaign status
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve campaign status

cURL example

```
curl -X GET 'http://localhost:1789/campaigns/<campaign>/status' \
  -H 'Authorization: Bearer <token>' \
```

Download step result

`GET /campaigns/{campaign}/step/{stepId}/download-result`

Downloads step result for a specific campaign. You can filter the punnets to download by status and metadata values

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name of the step
`mapId`	query	yes	object	Map id related to the campaign
`stepId`	path	yes	object	Step id to download result
`status`	query	no	array[string]	Status to filter punnets to download

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully downloaded step result

Status	Content-Type	Schema	Description
500	—	—	Server failed to download step result
400	—	—	Invalid request parameters

cURL example

```
curl -X GET
'http://localhost:1789/campaigns/<campaign>/step/<stepId>/download-
result' \
  -H 'Authorization: Bearer <token>' \
```

Pause a step

`POST /campaigns/{campaign}/step/{stepId}/pause`

Pause a step by providing the campaign name, map ID, and step ID. These parameters define the exact context of the task to pause.

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	The campaign name to identify the task context
`mapId`	query	yes	object	The map Id related to the campaign

Name	In	Required	Type	Description
`stepId`	path	yes	object	The step Id defining the specific task to pause

Responses

Status	Content-Type	Schema	Description
204	application/json	object	Successfully paused step
400	—	—	Invalid request parameters
404	—	—	Campaign not found
500	—	—	Failed to pause step

cURL example

```
curl -X POST
'http://localhost:1789/campaigns/<campaign>/step/<stepId>/pause' \
  -H 'Authorization: Bearer <token>' \
```

Resume a step

`POST /campaigns/{campaign}/step/{stepId}/resume`

Resume a step by providing the campaign name, map ID, and step ID. These parameters define the exact context of the task to resume.

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	The campaign name to identify the task context
`mapId`	query	yes	object	The map Id related to the campaign
`stepId`	path	yes	object	The step Id defining the specific task to resume

Responses

Status	Content-Type	Schema	Description
500	—	—	Failed to resume step
400	—	—	Invalid request parameters
404	—	—	Campaign not found
204	application/json	object	Successfully resumed step

cURL example

```
curl -X POST
'http://localhost:1789/campaigns/<campaign>/step/<stepId>/resume' \
  -H 'Authorization: Bearer <token>' \
```

Stop a campaign

`POST /campaigns/{campaign}/stop`

Stop a campaign from its name

Parameters

Name	In	Required	Type	Description
`campaign`	path	yes	object	Campaign name

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully stopped campaign
400	—	—	Campaign not started
500	—	—	Failed to resume campaign
404	—	—	Campaign not found

cURL example

```
curl -X POST 'http://localhost:1789/campaigns/<campaign>/stop' \
  -H 'Authorization: Bearer <token>' \
```

Catalog API

Endpoint to retrieve catalog tasks

getCatalog

`GET /catalog`

Parameters

Name	In	Required	Type	Description
`name`	query	no	string	
`classNames`	query	no	array[string]	
`allTask`	query	no	boolean	

Responses

Status	Content-Type	Schema	Description
200	application/json	array[object]	OK

cURL example

```
curl -X GET 'http://localhost:1789/catalog' \
  -H 'Authorization: Bearer <token>' \
```

getCatalogDto

`GET /catalog/dto`

Parameters

Name	In	Required	Type	Description
`name`	query	no	string	
`classNames`	query	no	array[string]	
`allTask`	query	no	boolean	

Responses

Status	Content-Type	Schema	Description
200	application/json	array[object]	OK

cURL example

```
curl -X GET 'http://localhost:1789/catalog/dto' \
  -H 'Authorization: Bearer <token>' \
```

Email API

Endpoint for managing emails

createEmail

`POST /emails`

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`email`	string	no	
`active`	boolean	no	
`emailId`	object	no	

Body format:

```
{
  "email": "string",
  "active": true,
  "emailId": {}
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X POST 'http://localhost:1789/emails' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "string",
  "active": true,
  "emailId": {}
}'
```

updateEmail

`PUT /emails`

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`email`	string	no	
`active`	boolean	no	
`emailId`	object	no	

Body format:

```
{
  "email": "string",
  "active": true,
  "emailId": {}
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X PUT 'http://localhost:1789/emails' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "string",
  "active": true,
  "emailId": {}
}'
```

deleteEmails_byNames

`DELETE /emails/delete-by-names`

Parameters

Name	In	Required	Type	Description
`emails`	query	yes	array[string]	

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`from`	integer(int32)	no	
`size`	integer(int32)	no	
`orderBy`	string	no	
`ascending`	boolean	no	

Body format:

```
{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/emails/delete-by-names' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}'
```

deleteEmails_byPattern

`DELETE /emails/delete-by-pattern`

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`from`	integer(int32)	no	
`size`	integer(int32)	no	
`orderBy`	string	no	

Field	Type	Required	Description
`ascending`	boolean	no	

Body format:

```
{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/emails/delete-by-pattern' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}'
```

getEmailsByNames

`GET /emails/search-by-names`

Parameters

Name	In	Required	Type	Description
`emails`	query	yes	array[string]	
`paginateParams`	query	yes	object	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/emails/search-by-names' \
  -H 'Authorization: Bearer <token>' \
```

getEmails

`GET /emails/search-by-pattern`

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	
`paginateParams`	query	yes	object	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/emails/search-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

deleteEmail

`DELETE /emails/{email}`

Parameters

Name	In	Required	Type	Description
`email`	path	yes	string	

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/emails/<email>' \
  -H 'Authorization: Bearer <token>' \
```

getEmail

`GET /emails/{email}`

Parameters

Name	In	Required	Type	Description
`email`	path	yes	string	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/emails/<email>' \
  -H 'Authorization: Bearer <token>' \
```

Job API

Endpoint for managing jobs

Create a job

`POST /jobs`

Creates a new job for a map with its cron expression

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`jobId`	object	no	
`jobName`	string	no	
`campaign`	object	no	
`taskFlowMapRef`	object	no	
`maxNumberExecutions`	integer(int32)	no	
`action`	string	no	
`active`	boolean	no	

Field	Type	Required	Description
`cronExpression`	string	no	

Body format:

```
{
  "jobId": {},
  "jobName": "string",
  "campaign": {
    "name": "string",
    "index": "string",
    "prefix": "string",
    "suffix": 0
  },
  "taskFlowMapRef": {},
  "maxNumberExecutions": 0,
  "action": "STOP",
  "active": true,
  "cronExpression": "string"
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully created job
400	—	—	Invalid request parameters
409	—	—	Job name already taken

Status	Content-Type	Schema	Description
500	—	—	Server failed to create job

cURL example

```
curl -X POST 'http://localhost:1789/jobs' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "jobId": {},
  "jobName": "string",
  "campaign": {
    "name": "string",
    "index": "string",
    "prefix": "string",
    "suffix": 0
  },
  "taskFlowMapRef": {},
  "maxNumberExecutions": 0,
  "action": "STOP",
  "active": true,
  "cronExpression": "string"
}'
```

Update a job

`PUT /jobs`

Updates a job. The job must have a name and an Id are required

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`jobId`	object	no	
`jobName`	string	no	
`campaign`	object	no	
`taskFlowMapRef`	object	no	
`maxNumberExecutions`	integer(int32)	no	
`action`	string	no	
`active`	boolean	no	
`cronExpression`	string	no	

Body format:

```
{
  "jobId": {},
  "jobName": "string",
  "campaign": {
    "name": "string",
    "index": "string",
    "prefix": "string",
    "suffix": 0
  },
  "taskFlowMapRef": {},
  "maxNumberExecutions": 0,
  "action": "STOP",
  "active": true,
  "cronExpression": "string"
}
```

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to update job
404	—	—	Provided job not found
400	—	—	Invalid request parameters
409	—	—	Job name already taken
200	application/json	object	Successfully updated job

cURL example

```
curl -X PUT 'http://localhost:1789/jobs' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "jobId": {},
  "jobName": "string",
  "campaign": {
    "name": "string",
    "index": "string",
    "prefix": "string",
    "suffix": 0
  },
  "taskFlowMapRef": {},
  "maxNumberExecutions": 0,
  "action": "STOP",
  "active": true,
  "cronExpression": "string"
}'
```

Delete jobs by name

`DELETE /jobs/delete-by-name`

Deletes jobs that match the list of specified job names. Returns a multi-status response indicating the success or failure of deleting each job

Parameters

Name	In	Required	Type	Description
`jobNames`	query	yes	array[string]	Job names to delete

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully deleted jobs
400	—	—	Invalid request parameters
207	application/json	object	Server failed to delete campaigns

cURL example

```
curl -X DELETE 'http://localhost:1789/jobs/delete-by-name' \
  -H 'Authorization: Bearer <token>' \
```

Delete jobs by pattern

`DELETE /jobs/delete-by-pattern`

Deletes jobs that match the specified name pattern. If no name pattern is provided, all jobs will be selected. Returns a multi-status response indicating the success or failure of deleting each job

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter job names

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully deleted jobs
400	—	—	Invalid request parameters
207	application/json	object	Server failed to delete campaigns

cURL example

```
curl -X DELETE 'http://localhost:1789/jobs/delete-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

Get jobs by name

`GET /jobs/search-by-names`

Retrieves jobs that match the list of specified job names

Parameters

Name	In	Required	Type	Description
`jobNames`	query	yes	array[string]	Job names to retrieve
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve jobs
200	application/json	object	Successfully retrieved jobs

cURL example

```
curl -X GET 'http://localhost:1789/jobs/search-by-names' \
  -H 'Authorization: Bearer <token>' \
```

Get jobs by pattern

`GET /jobs/search-by-pattern`

Retrieves jobs that match the specified name pattern. If no name pattern is provided, all jobs will be selected

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter job names
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve jobs
200	application/json	object	Successfully retrieved jobs

cURL example

```
curl -X GET 'http://localhost:1789/jobs/search-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

isCronValidFromString

`GET /jobs/validateCron`

Parameters

Name	In	Required	Type	Description
`cronExpression`	query	yes	string	

Responses

Status	Content-Type	Schema	Description
200	application/json	boolean	OK

cURL example

```
curl -X GET 'http://localhost:1789/jobs/validateCron' \
  -H 'Authorization: Bearer <token>' \
```

Delete a job

`DELETE /jobs/{jobName}`

Deletes a job from its name

Parameters

Name	In	Required	Type	Description
`jobName`	path	yes	string	The name of the job

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
404	—	—	Job not found
200	—	—	Successfully deleted job
500	—	—	Server failed to delete job

cURL example

```
curl -X DELETE 'http://localhost:1789/jobs/<jobName>' \
  -H 'Authorization: Bearer <token>' \
```

Get job from name

`GET /jobs/{jobName}`

Retrieves a job from its name

Parameters

Name	In	Required	Type	Description
`jobName`	path	yes	string	The name of the job

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve job
400	—	—	Invalid request parameters
404	—	—	Job not found
200	application/json	object	Successfully retrieved job

cURL example

```
curl -X GET 'http://localhost:1789/jobs/<jobName>' \
  -H 'Authorization: Bearer <token>' \
```

Map API

Endpoint for managing maps

Create a map

`POST /maps`

Creates a new map with an id and a unique name

Request Body

Content-Type: `application/json`

Responses

Status	Content-Type	Schema	Description
409	—	—	Map name already taken
500	—	—	Server failed to create map
400	—	—	Invalid request parameters
200	application/json	string	Successfully created map

cURL example

```
curl -X POST 'http://localhost:1789/maps' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '"string"'
```

Save a map

`PUT /maps`

Saves a whole map including step configurations

Request Body

Content-Type: `application/json`

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to save map
200	application/json	string	Successfully saved maps
409	—	—	New map name already taken
400	—	—	Invalid request parameters
404	—	—	Map not found from provided mapId or map versions weren't matching between provided and found map

cURL example

```
curl -X PUT 'http://localhost:1789/maps' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '"string"'
```

Delete maps by Ids

`DELETE /maps/delete-by-ids`

Deletes maps that match the list of specified map Ids

Parameters

Name	In	Required	Type	Description
`mapIds`	query	yes	array[object]	Map ids of the maps to delete

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully deleted maps
500	—	—	Server failed to delete maps
400	—	—	Invalid request parameters

cURL example

```
curl -X DELETE 'http://localhost:1789/maps/delete-by-ids' \
  -H 'Authorization: Bearer <token>' \
```

Delete maps by pattern

`DELETE /maps/delete-by-pattern`

Deletes maps that match the specified name pattern. If no name pattern is provided, all maps will be selected

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter map names

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully deleted maps
500	—	—	Server failed to delete maps
400	—	—	Invalid request parameters

cURL example

```
curl -X DELETE 'http://localhost:1789/maps/delete-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

Delete maps by version

`DELETE /maps/delete-by-version`

Deletes maps summary that match the specified map version Ids

Parameters

Name	In	Required	Type	Description
`mapVersionSerieId`	query	yes	object	The mapVersionSerieId attached to the map

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`from`	integer(int32)	no	
`size`	integer(int32)	no	
`orderBy`	string	no	
`ascending`	boolean	no	

Body format:

```
{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully deleted maps
500	—	—	Server failed to delete maps
400	—	—	Invalid request parameters

cURL example

```
curl -X DELETE 'http://localhost:1789/maps/delete-by-version' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}'
```

Download map

`GET /maps/download/{mapId}`

Downloads map file from provided map id

Parameters

Name	In	Required	Type	Description
`mapId`	path	yes	object	The map Id of the map to download

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to download maps
200	application/json	object	Successfully downloaded maps
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/maps/download/<mapId>' \
  -H 'Authorization: Bearer <token>' \
```

Assert map name availability

`GET /maps/name-availability`

Checks that provided map name is available

Parameters

Name	In	Required	Type	Description
`mapName`	query	yes	string	The map name to check availability

Responses

Status	Content-Type	Schema	Description
200	application/json	boolean	Successfully checked map name availability
500	—	—	Server failed to check map name
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/maps/name-availability' \
  -H 'Authorization: Bearer <token>' \
```

Get maps by Ids

`GET /maps/search-by-ids`

Retrieves maps that match the list of specified map Ids

Parameters

Name	In	Required	Type	Description
`mapIds`	query	yes	array[object]	Map ids of the maps to retrieve
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
200	application/json	string	Successfully retrieved maps
500	—	—	Server failed to retrieve maps

cURL example

```
curl -X GET 'http://localhost:1789/maps/search-by-ids' \
  -H 'Authorization: Bearer <token>' \
```

Get maps by pattern

`GET /maps/search-by-pattern`

Retrieves maps that match the specified name pattern. If no name pattern is provided, all maps will be selected

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter map names
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
200	application/json	string	Successfully retrieved maps
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve maps

cURL example

```
curl -X GET 'http://localhost:1789/maps/search-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

Get maps by version Id

`GET /maps/search-by-version`

Retrieves maps summary that match the list of specified map Ids

Parameters

Name	In	Required	Type	Description
`mapVersionSerieId`	query	yes	object	The mapVersionSerieId attached to the map
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
200	application/json	string	Successfully retrieved maps
404	—	—	Provided mapVersionSerieId not found
500	—	—	Server failed to retrieve maps

cURL example

```
curl -X GET 'http://localhost:1789/maps/search-by-version' \
  -H 'Authorization: Bearer <token>' \
```

Get maps summaries by Id

`GET /maps/summary/search-by-ids`

Retrieves maps summaries that match the list of specified map Ids

Parameters

Name	In	Required	Type	Description
`mapIds`	query	yes	array[object]	Map ids of the maps to retrieve
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully retrieved maps
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve maps

cURL example

```
curl -X GET 'http://localhost:1789/maps/summary/search-by-ids' \
  -H 'Authorization: Bearer <token>' \
```

Get maps summaries by pattern

`GET /maps/summary/search-by-pattern`

Retrieves maps summaries that match the specified name pattern. If no name pattern is provided, all maps will be selected

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter map names
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully retrieved maps
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve maps

cURL example

```
curl -X GET 'http://localhost:1789/maps/summary/search-by-pattern'
\
  -H 'Authorization: Bearer <token>' \
```

Upload a map

`POST /maps/upload/{mapName}`

Created a new map from file and associates it with the given map name

Parameters

Name	In	Required	Type	Description
`mapName`	path	yes	string	The map name to associates

Request Body

Content-Type: `multipart/form-data`

Field	Type	Required	Description
`file`	string(binary)	yes	The file to be uploaded

Body format:

```
{
  "file": "string"
}
```

Responses

Status	Content-Type	Schema	Description
409	—	—	Map name already taken
400	—	—	Invalid request parameters
200	application/json	string	Successfully uploaded map

Status	Content-Type	Schema	Description
500	—	—	Server failed to upload maps

cURL example

```
curl -X POST 'http://localhost:1789/maps/upload/<mapName>' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: multipart/form-data' \
  -d '{
  "file": "string"
}'
```

Delete a map from its Id

`DELETE /maps/{mapId}`

Deletes one map from provided map Id

Parameters

Name	In	Required	Type	Description
`mapId`	path	yes	object	Map Id attached to the map to delete

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to delete maps

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
404	—	—	Map not found
200	—	—	Successfully deleted maps

cURL example

```
curl -X DELETE 'http://localhost:1789/maps/<mapId>' \
  -H 'Authorization: Bearer <token>' \
```

Get a map from its id

`GET /maps/{mapId}`

Retrieves one map from its map Id

Parameters

Name	In	Required	Type	Description
`mapId`	path	yes	object	The map Id attached to the map

Responses

Status	Content-Type	Schema	Description
404	—	—	Map not found from provided mapId

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve map
200	application/json	string	Successfully retrieved map

cURL example

```
curl -X GET 'http://localhost:1789/maps/<mapId>' \
  -H 'Authorization: Bearer <token>' \
```

Punnet API

Endpoint to retrieve punnets

Get index mapping campaign

`GET /punnets/mapping`

Retrieves all field mappings (structure of documents) for the OpenSearch index related to the specified campaign

Parameters

Name	In	Required	Type	Description
`campaign`	query	yes	object	Campaign name

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve mapping
200	application/json	object	Successfully retrieved fields mapping
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/punnets/mapping' \
  -H 'Authorization: Bearer <token>' \
```

Get all punnet contexts

`GET /punnets/punnet-contexts`

Retrieves all punnet contexts from a campaign and a stepId. You can filter results by punnet metadata

Parameters

Name	In	Required	Type	Description
`campaign`	query	yes	object	Campaign name
`stepId`	query	yes	object	The step Id that processed the punnet

Name	In	Required	Type	Description
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve punnet contexts
200	application/json	object	Successfully retrieve punnet contexts
400	—	—	Invalid request parameters

cURL example

```
curl -X GET 'http://localhost:1789/punnets/punnet-contexts' \
  -H 'Authorization: Bearer <token>' \
```

Get values of a metadata

`GET /punnets/values`

Retrieves all values of a given metadata

Parameters

Name	In	Required	Type	Description
`campaign`	query	yes	object	Campaign name
`mapId`	query	yes	object	Map Id reference
`stepId`	query	yes	object	The step Id that processed the punnet
`field`	query	yes	string	The metadata key to retrieve values

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve values
400	—	—	Invalid request parameters
200	application/json	object	Successfully retrieved values

cURL example

```
curl -X GET 'http://localhost:1789/punnets/values' \
  -H 'Authorization: Bearer <token>' \
```

Get punnet context from its id

`GET /punnets/{punnetContextId}`

Retrieves PunnetContext information from its PunnetContextId

Parameters

Name	In	Required	Type	Description
`punnetContextId`	path	yes	object	PunnetContextId reference

Responses

Status	Content-Type	Schema	Description
404	—	—	PunnetContextId did not exist
200	application/xml	object	Successfully retrieve punnet
400	—	—	Invalid request parameters
500	—	—	Server failed to transform punnet as xml

cURL example

```
curl -X GET 'http://localhost:1789/punnets/<punnetContextId>' \
  -H 'Authorization: Bearer <token>' \
```

Get punnet exception

`GET /punnets/{punnetContextId}/exception`

Retrieves exception of any punnet from its PunnetContextId

Parameters

Name	In	Required	Type	Description
`punnetContextId`	path	yes	object	PunnetContextId reference

Responses

Status	Content-Type	Schema	Description
404	—	—	PunnetContextId not found
200	application/json	object	Successfully retrieved exception
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve exception

cURL example

```
curl -X GET
'http://localhost:1789/punnets/<punnetContextId>/exception' \
  -H 'Authorization: Bearer <token>' \
```

Get next punnets

`GET /punnets/{punnetContextId}/history/next`

Retrieves next PunnetContext information from a PunnetContextId. Finding several punnets comes from a duplication of punnets in the map. Can be empty if punnet context does not have any next punnet context

Parameters

Name	In	Required	Type	Description
`punnetContextId`	path	yes	object	PunnetContextId reference

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve punnet contexts
200	application/json	object	Successfully retrieved punnet contexts
400	—	—	Invalid request parameters

cURL example

```
curl -X GET
'http://localhost:1789/punnets/<punnetContextId>/history/next' \
  -H 'Authorization: Bearer <token>' \
```

Get previous punnet

`GET /punnets/{punnetContextId}/history/previous`

Retrieves previous PunnetContext information from a PunnetContextId. Unlike the next route, it is only possible to find a single punnet context. Can be null if punnet context does not have any previous punnet context

Parameters

Name	In	Required	Type	Description
`punnetContextId`	path	yes	object	PunnetContextId reference

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully retrieved the punnet context
400	—	—	Invalid request parameters
500	—	—	Server failed to retrieve the punnet context

cURL example

```
curl -X GET
'http://localhost:1789/punnets/<punnetContextId>/history/previous'
\
  -H 'Authorization: Bearer <token>' \
```

Get punnet logs

`GET /punnets/{punnetContextId}/logs`

Retrieves logs of any punnet from its PunnetContextId

Parameters

Name	In	Required	Type	Description
`punnetContextId`	path	yes	object	PunnetContextId reference

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve logs
400	—	—	Invalid request parameters
200	application/json	object	Successfully retrieved logs

cURL example

```
curl -X GET 'http://localhost:1789/punnets/<punnetContextId>/logs'
\
  -H 'Authorization: Bearer <token>' \
```

Get punnet as xml

`GET /punnets/{punnetContextId}/xml`

Retrieves a punnet from its PunnetContextId and prints it as an XML file

Parameters

Name	In	Required	Type	Description
`punnetContextId`	path	yes	object	PunnetContextId reference

Responses

Status	Content-Type	Schema	Description
404	—	—	PunnetContextId not found
400	—	—	Invalid request parameters
200	application/xml	object	Successfully retrieve punnet
500	—	—	Server failed to transform punnet as xml

cURL example

```
curl -X GET 'http://localhost:1789/punnets/<punnetContextId>/xml' \
  -H 'Authorization: Bearer <token>' \
```

Queue API

Endpoint for managing queues

getQueues

`GET /queues`

Parameters

Name	In	Required	Type	Description
`paginateParams`	query	yes	object	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/queues' \
  -H 'Authorization: Bearer <token>' \
```

createQueue

`POST /queues`

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`queueId`	object	no	
`numberOfSourceThreads`	integer(int32)	no	
`numberOfTaskThreads`	integer(int32)	no	

Body format:

```
{
  "queueId": {},
  "numberOfSourceThreads": 0,
  "numberOfTaskThreads": 0
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X POST 'http://localhost:1789/queues' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "queueId": {},
  "numberOfSourceThreads": 0,
  "numberOfTaskThreads": 0
}'
```

updateQueue

`PUT /queues`

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`queueId`	object	no	
`numberOfSourceThreads`	integer(int32)	no	
`numberOfTaskThreads`	integer(int32)	no	

Body format:

```
{
  "queueId": {},
  "numberOfSourceThreads": 0,
  "numberOfTaskThreads": 0
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X PUT 'http://localhost:1789/queues' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "queueId": {},
  "numberOfSourceThreads": 0,
  "numberOfTaskThreads": 0
}'
```

deleteQueues_byIds

`DELETE /queues/delete-by-ids`

Parameters

Name	In	Required	Type	Description
`queueIds`	query	yes	array[object]	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/queues/delete-by-ids' \
  -H 'Authorization: Bearer <token>' \
```

deleteQueues_byPattern

`DELETE /queues/delete-by-pattern`

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/queues/delete-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

deleteQueue

`DELETE /queues/{queueId}`

Parameters

Name	In	Required	Type	Description
`queueId`	path	yes	object	

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/queues/<queueId>' \
  -H 'Authorization: Bearer <token>' \
```

getQueue

`GET /queues/{queueId}`

Parameters

Name	In	Required	Type	Description
`queueId`	path	yes	object	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/queues/<queueId>' \
  -H 'Authorization: Bearer <token>' \
```

Shared Objects API

Endpoint for managing shared objects

Delete shared objects by names

`DELETE /shared-objects/delete-by-names`

Deletes shared objects that match the list of specified shared objects names. Returns a multi-status response indicating the success or failure of deleting each shared objects

Parameters

Name	In	Required	Type	Description
`sharedObjectNames`	query	yes	array[string]	Shared object names to delete

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`from`	integer(int32)	no	
`size`	integer(int32)	no	
`orderBy`	string	no	
`ascending`	boolean	no	

Body format:

```
{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}
```

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
207	application/json	object	Server failed to delete shared objects

Status	Content-Type	Schema	Description
200	application/json	object	Successfully deleted shared objects

cURL example

```
curl -X DELETE 'http://localhost:1789/shared-objects/delete-by-
names' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}'
```

Delete shared objects by pattern

`DELETE /shared-objects/delete-by-pattern`

Deletes shared objects that match the specified name pattern. If no name pattern is provided, all shared objects will be selected. Returns a multi-status response indicating the success or failure of deleting each shared object

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter shared objects

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`from`	integer(int32)	no	
`size`	integer(int32)	no	
`orderBy`	string	no	
`ascending`	boolean	no	

Body format:

```
{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}
```

Responses

Status	Content-Type	Schema	Description
207	application/json	object	Server failed to delete shared objects
200	application/json	object	Successfully deleted shared objects

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters

cURL example

```
curl -X DELETE 'http://localhost:1789/shared-objects/delete-by-
pattern' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "from": 0,
  "size": 0,
  "orderBy": "string",
  "ascending": true
}'
```

Get shared objects by names

`GET /shared-objects/search-by-names`

Retrieves shared objects that match the list of specified shared object names

Parameters

Name	In	Required	Type	Description
`sharedObjectNames`	query	yes	array[string]	Shared object names to retrieve

Name	In	Required	Type	Description
`mapId`	query	no	object	Identifier of the map from which to retrieve shared objects
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to retrieve shared object
400	—	—	Invalid request parameters
200	application/json	object	Successfully retrieved shared object

cURL example

```
curl -X GET 'http://localhost:1789/shared-objects/search-by-names' \
  -H 'Authorization: Bearer <token>' \
```

Get shared objects by pattern

`GET /shared-objects/search-by-pattern`

Retrieves shared objects that match the specified name pattern. If no name pattern is provided, all shared objects will be selected

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	Pattern to filter shared objects
`mapId`	query	no	object	Identifier of the map from which to retrieve shared objects
`paginateParams`	query	yes	object	Pagination parameters

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
500	—	—	Server failed to update shared object
200	application/json	object	Successfully retrieved shared object

cURL example

```
curl -X GET 'http://localhost:1789/shared-objects/search-by-
pattern' \
  -H 'Authorization: Bearer <token>' \
```

Delete a shared object

`DELETE /shared-objects/{sharedObjectName}`

Deletes a shared object from its name

Parameters

Name	In	Required	Type	Description
`sharedObjectName`	path	yes	string	Shared object name

Responses

Status	Content-Type	Schema	Description
400	—	—	Invalid request parameters
500	—	—	Server failed to delete the shared object
200	—	—	Successfully deleted the shared object
404	—	—	Shared object not found

cURL example

```
curl -X DELETE 'http://localhost:1789/shared-
objects/<sharedObjectName>' \
  -H 'Authorization: Bearer <token>' \
```

Get specific shared object

`GET /shared-objects/{sharedObjectName}`

Retrieve one shared object configuration from its name

Parameters

Name	In	Required	Type	Description
`sharedObjectName`	path	yes	string	Shared object name

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully retrieved shared object
500	—	—	Server failed to retrieve shared object
400	—	—	Invalid request parameters
404	—	—	Shared object not found

cURL example

```
curl -X GET 'http://localhost:1789/shared-
objects/<sharedObjectName>' \
  -H 'Authorization: Bearer <token>' \
```

Create a shared object

`POST /shared-objects/{sharedObjectName}`

Creates a shared object from its object configuration and a provided name

Parameters

Name	In	Required	Type	Description
`sharedObjectName`	path	yes	string	Shared object name

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`fields`	object	no	
`className`	string	no	
`scope`	string	no	
`fullyConfigured`	boolean	no	

Field	Type	Required	Description
`singleton`	boolean	no	
`objectName`	string	no	
`objectConfigurationId`	string	no	
`constructorArguments`	array[object]	no	
`fieldConfigurationType`	string	no	

Body format:

```
{
  "fields": {},
  "className": "string",
  "scope": "MAP",
  "fullyConfigured": true,
  "singleton": true,
  "objectName": "string",
  "objectConfigurationId": "string",
  "constructorArguments": [
    {
      "fieldConfigurationType": "Primitive"
    }
  ],
  "fieldConfigurationType": "Primitive"
}
```

Responses

Status	Content-Type	Schema	Description
500	—	—	Server failed to create shared object
400	—	—	Invalid request parameters
201	application/json	object	Successfully created shared object

cURL example

```
curl -X POST 'http://localhost:1789/shared-
objects/<sharedObjectName>' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "fields": {},
  "className": "string",
  "scope": "MAP",
  "fullyConfigured": true,
  "singleton": true,
  "objectName": "string",
  "objectConfigurationId": "string",
  "constructorArguments": [
    {
      "fieldConfigurationType": "Primitive"
    }
  ],
  "fieldConfigurationType": "Primitive"
}'
```

Update a shared object

`PUT /shared-objects/{sharedObjectName}`

Updates the configuration or the name of a shared object

Parameters

Name	In	Required	Type	Description
`sharedObjectName`	path	yes	string	Shared object name

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`fields`	object	no	
`className`	string	no	
`scope`	string	no	
`fullyConfigured`	boolean	no	
`singleton`	boolean	no	
`objectName`	string	no	
`objectConfigurationId`	string	no	
`constructorArguments`	array[object]	no	

Field	Type	Required	Description
`fieldConfigurationType`	string	no	

Body format:

```
{
  "fields": {},
  "className": "string",
  "scope": "MAP",
  "fullyConfigured": true,
  "singleton": true,
  "objectName": "string",
  "objectConfigurationId": "string",
  "constructorArguments": [
    {
      "fieldConfigurationType": "Primitive"
    }
  ],
  "fieldConfigurationType": "Primitive"
}
```

Responses

Status	Content-Type	Schema	Description
200	application/json	object	Successfully updated shared object
400	—	—	Invalid request parameters
500	—	—	Server failed to update shared object

cURL example

```
curl -X PUT 'http://localhost:1789/shared-
objects/<sharedObjectName>' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "fields": {},
  "className": "string",
  "scope": "MAP",
  "fullyConfigured": true,
  "singleton": true,
  "objectName": "string",
  "objectConfigurationId": "string",
  "constructorArguments": [
    {
      "fieldConfigurationType": "Primitive"
    }
  ],
  "fieldConfigurationType": "Primitive"
}'
```

User API

Endpoint for managing users

Delete all users

`DELETE /users`

Deletes all users

Parameters

Name	In	Required	Type	Description
`Authorization`	header	yes	string	

Request Body

Content-Type: `application/json`

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully created user

cURL example

```
curl -X DELETE 'http://localhost:1789/users' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '[
  "string"
]'
```

Get number of users per role

`GET /users/count-by-role`

Retrieves the number of users for each role

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully found number of users

cURL example

```
curl -X GET 'http://localhost:1789/users/count-by-role' \
  -H 'Authorization: Bearer <token>' \
```

Check if user exists

`GET /users/does-user-exist/{userEmail}`

Checks if a user with the given email exists

Parameters

Name	In	Required	Type	Description
`userEmail`	path	yes	string	

Responses

Status	Content-Type	Schema	Description
200	/	boolean	OK

cURL example

```
curl -X GET 'http://localhost:1789/users/does-user-
exist/<userEmail>' \
  -H 'Authorization: Bearer <token>' \
```

Update current user

`PATCH /users/me`

Updates information of the currently authenticated user

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`firstname`	string	no	
`lastname`	string	no	
`role`	string	no	

Body format:

```
{
  "firstname": "string",
  "lastname": "string",
  "role": "USER"
}
```

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully updated user

cURL example

```
curl -X PATCH 'http://localhost:1789/users/me' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "firstname": "string",
  "lastname": "string",
  "role": "USER"
}'
```

Create a user or an admin

`POST /users/register`

Creates a user from its password, firstname, lastname and email. You cannot create super admin with this endpoint

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`password`	string	no	
`firstname`	string	no	

Field	Type	Required	Description
`lastname`	string	no	
`email`	string	no	
`role`	string	no	

Body format:

```
{
  "password": "string",
  "firstname": "string",
  "lastname": "string",
  "email": "string",
  "role": "USER"
}
```

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully created user
403	—	—	User creation is restricted to a single user in the configuration
500	—	—	Server failed to create user

Status	Content-Type	Schema	Description
409	—	—	An account with this email already exists

cURL example

```
curl -X POST 'http://localhost:1789/users/register' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "password": "string",
  "firstname": "string",
  "lastname": "string",
  "email": "string",
  "role": "USER"
}'
```

Create a super admin

`POST /users/register-super-admin`

Creates a user from its password, firstname, lastname and email

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`password`	string	no	

Field	Type	Required	Description
`firstname`	string	no	
`lastname`	string	no	
`email`	string	no	
`role`	string	no	

Body format:

```
{
  "password": "string",
  "firstname": "string",
  "lastname": "string",
  "email": "string",
  "role": "USER"
}
```

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully created user
403	—	—	User creation is restricted to a single user in the configuration
500	—	—	Server failed to create user

Status	Content-Type	Schema	Description
409	—	—	An account with this email already exists

cURL example

```
curl -X POST 'http://localhost:1789/users/register-super-admin' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "password": "string",
  "firstname": "string",
  "lastname": "string",
  "email": "string",
  "role": "USER"
}'
```

Register a worker

`POST /users/register-worker`

Creates a worker as member to allow broker communication

Parameters

Name	In	Required	Type	Description
`X-Register-Token`	header	yes	string	

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`password`	string	no	
`firstname`	string	no	
`lastname`	string	no	
`email`	string	no	
`role`	string	no	

Body format:

```
{
  "password": "string",
  "firstname": "string",
  "lastname": "string",
  "email": "string",
  "role": "USER"
}
```

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully created user

Status	Content-Type	Schema	Description
403	—	—	User creation is restricted to a single user in the configuration
500	—	—	Server failed to create user
409	—	—	An account with this email already exists

cURL example

```
curl -X POST 'http://localhost:1789/users/register-worker' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "password": "string",
  "firstname": "string",
  "lastname": "string",
  "email": "string",
  "role": "USER"
}'
```

getUsersByPattern

`GET /users/search-by-pattern`

Parameters

Name	In	Required	Type	Description
`namePattern`	query	no	string	
`paginateParams`	query	yes	object	

Responses

Status	Content-Type	Schema	Description
200	/	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/users/search-by-pattern' \
  -H 'Authorization: Bearer <token>' \
```

Check admin existence

`GET /users/super-admin-exists`

Checks if super admin is registered

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully found admin

cURL example

```
curl -X GET 'http://localhost:1789/users/super-admin-exists' \
  -H 'Authorization: Bearer <token>' \
```

Delete a user

`DELETE /users/{userEmail}`

Deletes a user from its email

Parameters

Name	In	Required	Type	Description
`userEmail`	path	yes	string	

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully deleted user

cURL example

```
curl -X DELETE 'http://localhost:1789/users/<userEmail>' \
  -H 'Authorization: Bearer <token>' \
```

Get a user

`GET /users/{userEmail}`

Retrieves user information from its email

Parameters

Name	In	Required	Type	Description
`userEmail`	path	yes	string	

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully retrieved user

cURL example

```
curl -X GET 'http://localhost:1789/users/<userEmail>' \
  -H 'Authorization: Bearer <token>' \
```

Update another user

`PATCH /users/{userEmail}`

Admin updates any user's information

Parameters

Name	In	Required	Type	Description
`userEmail`	path	yes	string	

Request Body

Content-Type: `application/json`

Field	Type	Required	Description
`firstname`	string	no	
`lastname`	string	no	
`role`	string	no	

Body format:

```
{
  "firstname": "string",
  "lastname": "string",
  "role": "USER"
}
```

Responses

Status	Content-Type	Schema	Description
200	—	—	Successfully updated user

cURL example

```
curl -X PATCH 'http://localhost:1789/users/<userEmail>' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "firstname": "string",
  "lastname": "string",
  "role": "USER"
}'
```

Worker API

API for managing workers

deleteWorkers

`DELETE /workers`

Parameters

Name	In	Required	Type	Description
`workerIds`	query	no	array[object]	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/workers' \
  -H 'Authorization: Bearer <token>' \
```

getWorkers

`GET /workers`

Parameters

Name	In	Required	Type	Description
`workerIds`	query	no	array[object]	
`paginateParams`	query	yes	object	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/workers' \
  -H 'Authorization: Bearer <token>' \
```

spawnWorker

`POST /workers`

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X POST 'http://localhost:1789/workers' \
  -H 'Authorization: Bearer <token>' \
```

getLibraries

`GET /workers/libraries`

Parameters

Name	In	Required	Type	Description
`paginateParams`	query	yes	object	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/workers/libraries' \
  -H 'Authorization: Bearer <token>' \
```

getLibraryVersions

`GET /workers/library-versions/{libraryName}`

Parameters

Name	In	Required	Type	Description
`libraryName`	path	yes	string	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/workers/library-
versions/<libraryName>' \
  -H 'Authorization: Bearer <token>' \
```

restartWorkers

`POST /workers/restart`

Parameters

Name	In	Required	Type	Description
`workerIds`	query	no	array[object]	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X POST 'http://localhost:1789/workers/restart' \
  -H 'Authorization: Bearer <token>' \
```

restartWorkerById

`POST /workers/restart/{workerId}`

Parameters

Name	In	Required	Type	Description
`workerId`	path	yes	object	

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X POST 'http://localhost:1789/workers/restart/<workerId>' \
  -H 'Authorization: Bearer <token>' \
```

uploadLibrary

`POST /workers/upload-library`

Request Body

Content-Type: `multipart/form-data`

Field	Type	Required	Description
`file`	string(binary)	yes	

Body format:

```
{
  "file": "string"
}
```

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X POST 'http://localhost:1789/workers/upload-library' \
  -H 'Authorization: Bearer <token>' \
  -H 'Content-Type: multipart/form-data' \
  -d '{
  "file": "string"
}'
```

deleteWorkerById

`DELETE /workers/{workerId}`

Parameters

Name	In	Required	Type	Description
`workerId`	path	yes	object	

Responses

Status	Content-Type	Schema	Description
200	—	—	OK

cURL example

```
curl -X DELETE 'http://localhost:1789/workers/<workerId>' \
  -H 'Authorization: Bearer <token>' \
```

getWorkerById

`GET /workers/{workerId}`

Parameters

Name	In	Required	Type	Description
`workerId`	path	yes	object	

Responses

Status	Content-Type	Schema	Description
200	application/json	object	OK

cURL example

```
curl -X GET 'http://localhost:1789/workers/<workerId>' \
  -H 'Authorization: Bearer <token>' \
```

getWorkerLogs

`GET /workers/{workerId}/logs`

Parameters

Name	In	Required	Type	Description
`workerId`	path	yes	object	
`results`	query	no	integer(int32)	

Responses

Status	Content-Type	Schema	Description
200	application/json	array[object]	OK

cURL example

```
curl -X GET 'http://localhost:1789/workers/<workerId>/logs' \
  -H 'Authorization: Bearer <token>' \
```