Ngenea Hub

Ngenea Hub harnesses the power of Ngenea to provide global workflows, enabling your data to be where you need it, when you need it.

Installation

A deployment of Ngenea Hub comprises two main components:

- **Ngenea Hub** - the central management point, from which all tasks are managed
- **Ngenea Worker** - worker agents, installed on individual Ngenea servers, that execute the Ngenea tasks

Ngenea Worker are installed on one or more nodes in a Site, which typically represents a single Ngenea cluster or location.

Ngenea Hub needs to be accessible from all Ngenea Workers on the following ports:

- `6379/tcp`
- `5672/tcp`

Installing Ngenea Hub

Ngenea Hub can be installed in a number of ways, use one of the methods described below.

CentOS / Redhat - Online Installation

Configure Docker Authentication

> **Note:** This step is not required on PixStor systems

Configure Docker authentication for `eurepo.arcapix.com`.

```
docker login eurepo.arcapix.com
```

Installing Ngenea Hub

Transfer the `ngenea-hub` rpm to the target system.

Install the `ngenea-hub` package via yum.

```
yum install ngenea-hub-<version>.rpm
```

Optionally, create an initial Ngenea Hub configuration file at `/etc/sysconfig/ngeneahub`. This file contains the credentials which will be required for deploying

workers, and can also be edited to use external queue systems prior to starting the Ngenea Hub service. This file will be created automatically if it does not exist when the service is started.

```
ngeneahubctl createconfig
```

Enable and start the Ngenea Hub service.

```
systemctl enable --now ngeneahub
```

Check the status of the service with:

```
ngeneahubctl status
```

CentOS / Redhat - Offline installation

The `ngeneahub` service will attempt to pull the required docker images from the Ngenea software repository servers. In situations where this is not possible (due to network restrictions, for instance), the containers can be installed via additional RPM: `ngenea-hub-images`, available at the same location as the main RPM.

Once the RPMs are transferred to the target system, they can be installed using rpm.

```
rpm -ivh ngenea-hub-<version>.rpm ngenea-hub-images-<version>.rpm
```

Cloud Deployment

Coming soon: image-based deployment of Ngenea Hub in the cloud.

Container Native Deployment

It is possible to deploy Ngenea Hub using standard container management tools and processes.

Please contact us to discuss.

Ngenea Worker

Pre-requisite: Ngenea Server software installed and configured

Install the `ngenea-worker` package via rpm.

```
yum install ngenea-worker
```

Add the appropriate worker configuration.

Enable and start the `ngenea-worker` systemd unit.

```
systemctl enable --now ngenea-worker
```

Once installed, additional features such as Search may be set up as described in Feature Set-up

Upgrade

Before you start

Ideally, before upgrading, you should wait for any pending or active jobs to complete, otherwise they may be lost.

For additional safety, any scheduled workflows may be temporarily disabled to prevent new jobs being submitted during the upgrade process.

Backup Workflows

Ngenea Hub ships with some default workflows. New releases may make changes to these workflows, so any customisations to them may be lost during upgrade. For safety, workflows should be backed-up before upgrading.

The easiest way to backup workflows is using ngclient

```
ngclient workflows list > workflows_backup.json
```

See NGCLIENT-WORKFLOWS for more information.

Stop Services

First, shutdown Ngenea Worker on all nodes

```
systemctl stop ngenea-worker
```

Note that any Ngenea Worker packages pre-1.12.0 will use the older syntax:

```
systemctl stop ngenea-worker@SITENAME
```

Then shutdown Ngenea Hub

```
systemctl stop ngeneahub
```

Upgrade Packages

Download the latest RPMs from the Download page.

Upgrade Ngenea Hub

```
yum upgrade ngenea-hub-<version>.rpm
```

As with Installation, for offline situations, the Ngenea Hub base and image rpms can be upgraded with

```
rpm -Uvh ngenea-hub-<version>.rpm ngenea-hub-images-<version>.rpm
```

Upgrade Ngenea Worker on all nodes

```
yum upgrade ngenea-worker-<version>.rpm
```

Restart Services

First, startup Ngenea Hub

```
systemctl start ngeneahub
```

Then start Ngenea Worker on all nodes

```
systemctl start ngenea-worker
```

If any scheduled workflows were disabled, they can now safely be re-enabled.

Validation

Check the status of the Ngenea Hub service with:

```
ngeneahubctl status
```

Check the status of Ngenea Worker service with

```
systemctl status ngenea-worker
```

Restoring Workflows

Check the workflows post-update. If there are any issue or inconsistencies with the upgraded workflows, they can be restored from the backup created pre-upgrade. This is also done using `ngclient`.

Any workflow which is missing can be re-imported using

```
ngclient workflows import <workflow_file>
```

Any workflow which has changed can be restored using

```
ngclient workflows update <id> <workflow_file>
```

Note, `ngclient` only allows importing or updating single workflows at a time. The `workflow_file` passed to the above commands must only contain a single workflow definition.

Configuration

Hub Initial Configuration

Once all services are up, create an admin user with:

```
ngeneahubctl adduser
```

Register an initial site (replacing `SITENAME`) (see Worker Installation for enabling a site's worker agents).

```
ngeneahubctl  addsite SITENAME
```

Log into the UI at http://server.address:8000.

Hub Configuration

Settings

The main configuration file for Ngenea Hub is at `/etc/sysconfig/ngeneahub`. This is an environment file which holds the information required for connecting to the various backend services.

Mandatory Settings

Setting	Description
DJANGO_SECRET	Secret string used secure signed data within django
POSTGRES_DB	Internal database name
POSTGRES_USER	Internal database username
POSTGRES_PASSWORD	Internal database password
BROKER_USER	Queue username
BROKER_PASSWORD	Queue password

Optional settings

Setting	Description
RABBITMQ_USER	User to use when connecting to the rabbitmq broker. Overrides `BROKER_USER`.
RABBITMQ_PASSWORD	Password to use when connecting to the rabbitmq broker. Overrides `BROKER_PASSWORD`.
RABBITMQ_HOST	

Setting	Description
	Address of the RabbitMQ broker service. Defaults to the container service address.
REDIS_PASSWORD	User to connect to the rabbitmq broker. Overrides `BROKER_PASSWORD`.
REDIS_HOST	Address of the Redis queue results store. Defaults to the container service address.
RABBITMQ_DEFAULT_USER	Username to use when initializing the RabbitMQ broker. Not used after the service has been initialised.
RABBITMQ_DEFAULT_PASS	Password to use when initializing the RabbitMQ broker. Not used after the service has been initialised. Overrides `BROKER_PASSWORD`.
WORKERS	The number of gunicorn workers to spawn for serving API requests. Default to 8.
CONSUMER_TIMEOUT	The timeout for rabbitmq consumer delivery acknowledgement in seconds. Default: 10800000 (3 hours)
HUB_PORT	User configurable hub port

Server Configurations

Some settings are stored in the Ngenea Hub DB.

They can be viewed and changed via the REST API `/api/configurations/` endpoint.

See Configuration for more details.

Docker Compose configuration

The `docker-compose` file is stored in `/usr/share/ngeneahub/docker/docker-compose.yml`.

This can be extended by creating an override file at `/usr/share/ngeneahub/docker/docker-compose.override.yml`.

Worker Configuration

The Ngenea Worker configuration should be added to `/etc/ngenea/ngenea-worker.conf`. The configuration is in ini format. For example:

```
[settings]
broker_url = amqp://user:password@localhost
result_backend = redis://:password@localhost:6379
site = site1
```

The following is a list of available settings:

Option	Type	Default	Required	Description
threads	int	10	No	The number of concurrent tasks that can be run.
result_backend	string		Yes	The URI for the Ngenea Hub results backend
broker_url	string		Yes	The URI for the Ngenea Hub broker
site	string		Yes	The name of the queue to listen to

Note: The initial access credentials can be found in the `/etc/sysconfig/ngeneahub` configuration file on the Ngenea Hub server. By default, the `BROKER_PASSWORD` is used for both the `broker_url` and the `result_backend` passwords, and the `BROKER_USER` is used for the `broker_url` username.

Feature Set-up

To use certain features in Ngenea Hub, additional set-up is required as described in this section.

Search

The search feature provides the ability to search for files across one or more sites.

This page describes the steps required to set up the search feature.

For information on how to use the search feature, see Search

Prerequisites

The search feature is only supported on workers running on a PixStor.

There are two backends which can provide search functionality to Ngenea Hub: **PixStor Search** and **PixStor Analytics**.

This document assumes that the desired backend has already been deployed, as described in the PixStor deployment guide. Be aware that neither backend will have been deployed by default.

Configuration

Search Backend

By default, Ngenea Hub will use the Analytics backend. If you want to use the PixStor Search backend instead, change the `search_backend` to `pixstor_search`, as described in Global Configurations

```
$ curl -s -X PATCH 'http://example.com/api/configurations/' -H
'Accept: application/json' -H "Authorization: Bearer
```

```
$JWT_ACCESS_TOKEN" -H 'Content-Type: application/json' -d
'{"search_backend": "pixstor_search"}'
```

All sites **must** use the same backend - all analytics or all PixStor Search.

Elasticsearch URL

If you are using the Analytics backend and the worker is running on **PixStor 6**, you will need to change the `elasticsearch_url` for the site to `http://localhost:19200`, as described in Site-specific Configurations

```
$ curl -s -X PATCH 'http://example.com/api/sites/1/' -H 'Accept:
application/json' -H "Authorization: Api-Key $APIKEY" -H 'Content-
Type: application/json' -d '{"elasticsearch_url": "localhost:19200"}'
```

If not set, the `elasticsearch_url` will default to `localhost:9200`, which is correct for workers running on PixStor 5.

Different sites may configure a different `elasticsearch_url`, for example if one is PixStor 5 and another is PixStor 6.

Search UI

Once configured, search can be used via the REST API as described in Search

To use search via the Ngenea Hub UI, you must first enable the `searchui` feature flag, as described in Feature Flags

For example, using ngclient

```
ngclient features enable searchui
```

Usage

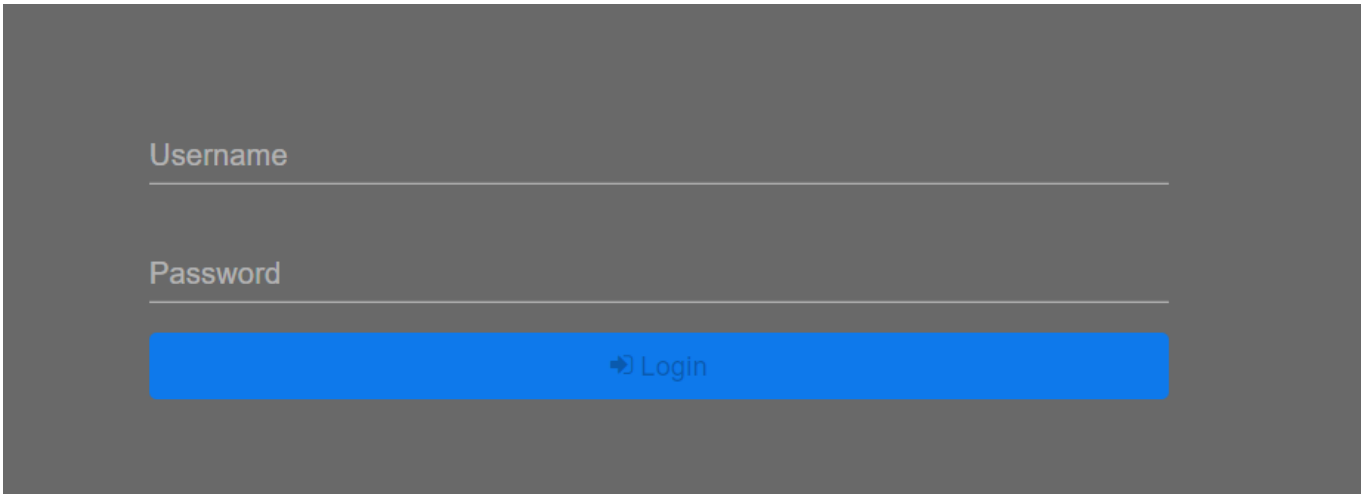
Web Interface

To access Ngenea Hub, go to http://example.com:8000/.

Authentication

Login

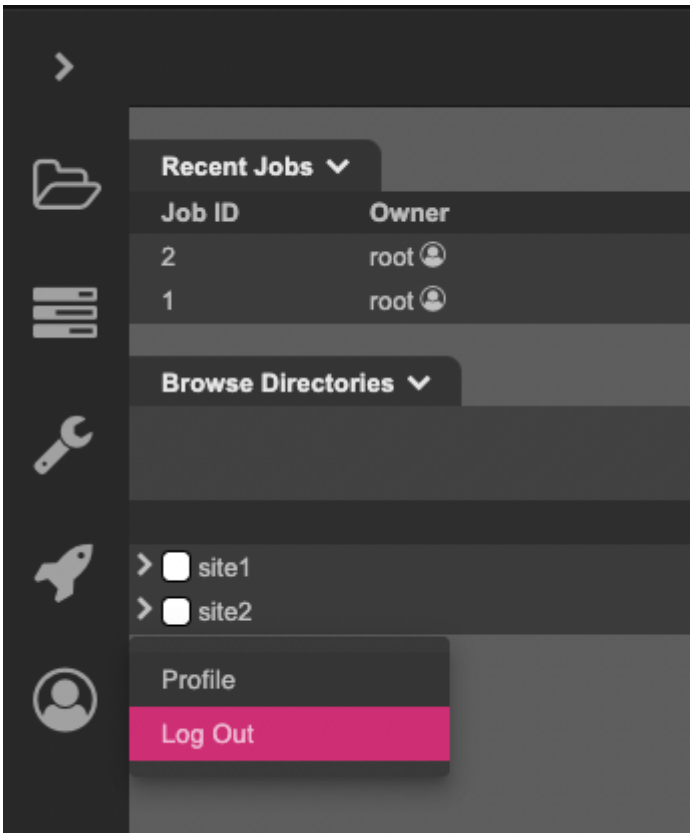
Upon navigating to the Ngenea Hub UI a login screen is presented.



Enter a valid username and password before pressing the `Login` button to authenticate.

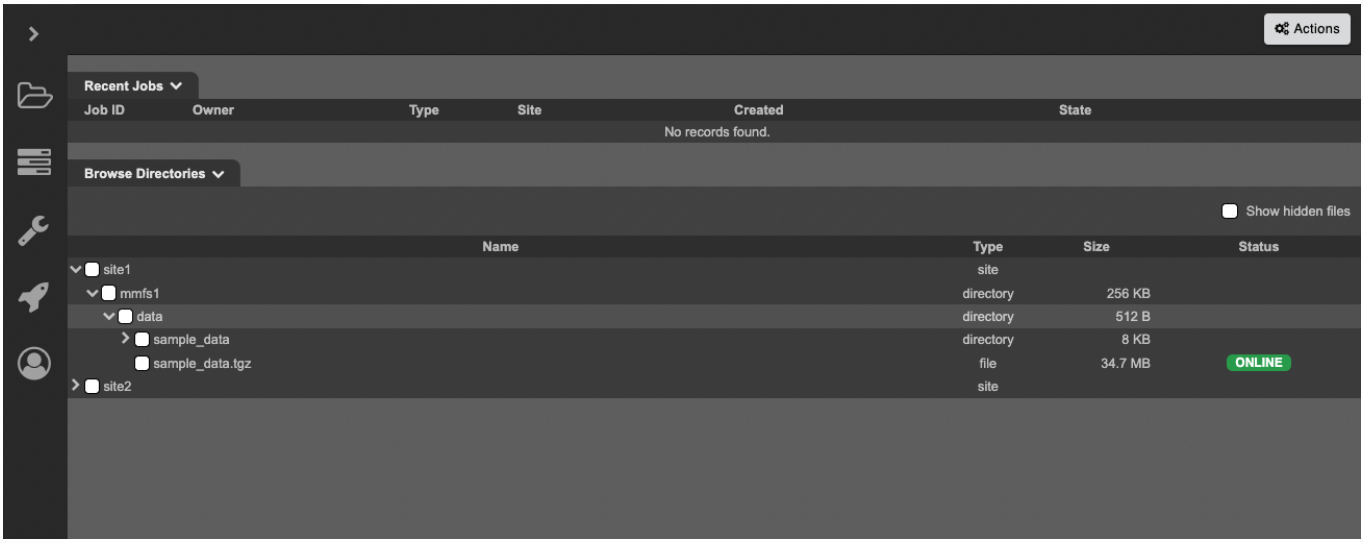
Logout

To end your session, select Logout from the `Man Icon` at the bottom left hand corner of the UI.



Browser

Upon logging in, select the directory icon on the left hand side of the UI. This takes to the Browser page.



The Browser page allows users to see a list of recent jobs as well as browse all available directories across all Sites for the purpose of either migrating, pre-migrating, recalling, or sending files.

Recent Jobs

The Recent Jobs section shows a list of the 5 most recent jobs that were initiated via the UI.

The view presents several columns:

Option	Description
Job ID	Shows the identification number associated with the job
Owner	Name of the user who created the job
Type	Shows the job's type, i.e. Migrate, Recall, Premigrate, or Send
Site	Name of the Site where files were migrated from
Created	Shows the job's creation time
State	Shows the job's state, i.e. success or failure

Search

> **Note:** Before you can use the search feature, additional set-up is necessary, as described in the search feature page.

Search section contains a search bar to discover the contents of the configured Sites.

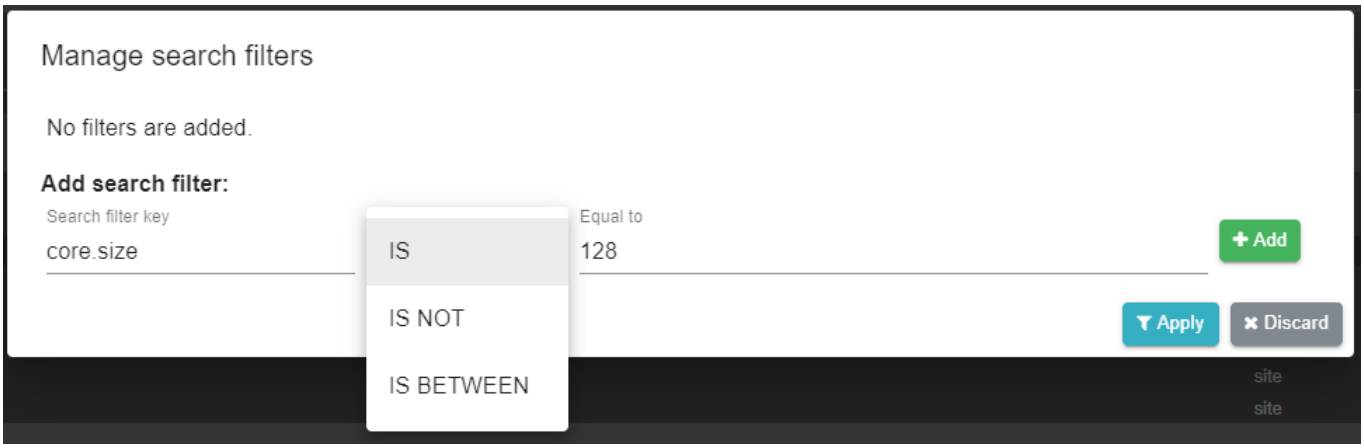
The view presents a search bar, a button for managing the search filters and another button for starting and stopping the search. The view also presents `(?)` button to get more instructions about starting a search.

To search for a term, type the search term in the search bar and click `Search` button (or hit `Enter` key) to start a search. The default search filter is "core.filename".

Managing the search filters from the dialog

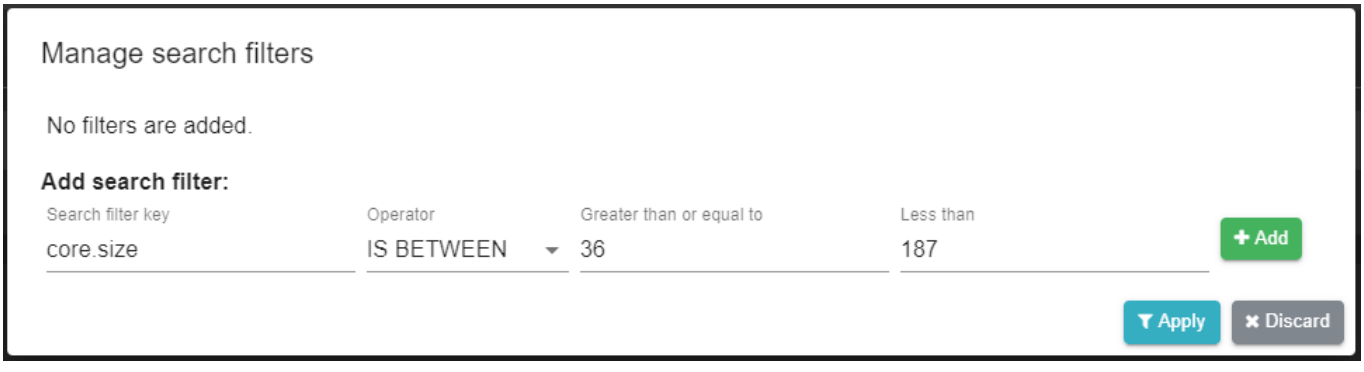
For more complex search operations, add more filters by clicking `Apply Filters` button and use the dialog opened after that. Type the search filter key in the input box with the label `Search filter key` and then pick an operator from the select box with the label `Operator`.

For the operator `IS`, the dialog is seen as in the image below:



For the operator `IS NOT`, the dialog is seen as in the image below:

For the operator `IS BETWEEN`, the dialog is seen as in the image below:



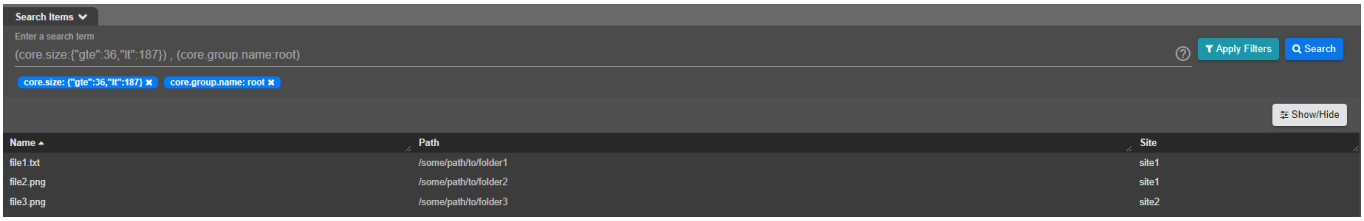
Click `Add` button to add the search filter. The dialog is seen as in the image below, when "core.size" is selected to be greater than or equal to 36 and less than 187, and "core.group.name" is selected to be "root".

Once the selection is finished, click `Apply` button to save & close the dialog (this action does not submit the search). See that filters are shown as a query in the search bar and they are shown with badges under the search bar. The search section is seen as in the image below, after the search filters are selected:

Submitting the search and viewing search results

Click `Search` button, or hit `Enter` key to start the search. The search section will contain search results from each configured site, matching with the search filters. To stop the search, click `Stop` button. The search section is seen as in the image below after the search is in progress.

The search section is seen as in the image below after some results are found.



Advanced usage: Managing the search filters from the search bar

You can also make complex search operations by typing the search term in the format below:

```
(<key1>:<value1>) , (<key2>:<value2>) , (<key3>:<value3>) ...
```

For multiple search filters, you need to separate them by paranthesis. For using various comparison operators, here are the filter formats:

Operation	Format
Equal to	`(<key1>: <value1>)`

Operation	Format
Less than	`(<key2>: {"lt": <value2> })`
Greater than or equal to	`(<key3>: {"gte": <value3> })`
Combined	`(<key4>: {"gte": <value4>, "lt": <value5> })`

Browse Directories

The Browse Directories section contains a list of configured Sites and correspondent directories under the Sites' filesystems.

The view presents several columns:

Option	Description
Name	Shows the name of the Site, directories, and files
Type	Shows whether the listed item is a Site, directory, or file
Size	Shows the directories and files size
Status	Shows whether files are online or offline

User can select one or multiple directories, or one or multiple files to migrate, premigrate, recall them or send them to a different Site.

Migrate

To Migrate a directory or file, expand the Site containing said directory and file. Select the directory or file you wish to Migrate by ticking their relevant boxes.

Click the "Actions" button at the top right hand side of the page and select "Migrate".

A new Job is created and it is shown at the top of the "Recent Jobs" list. Job's State will display a progress bar until completion.

Once job is complete, the State will either show as Success or Failed.

Premigrate

To Premigrate a directory or file, expand the Site containing said directory and file. Select the directory or file you wish to Premigrate by ticking their relevant boxes.

Click the "Actions" button at the top right hand side of the page and select "Premigrate".

A new Job is created and it is shown at the top of the "Recent Jobs" list. Job's State will display a progress bar until completion.

Once job is complete, the State will either show as Success or Failed.

Recall

To Recall a directory or file, expand the Site containing said directory and file. Select the directory or file you wish to Recall by ticking their relevant boxes.

Click the "Actions" button at the top right hand side of the page and select "Recall".

A new Job is created and it is shown at the top of the "Recent Jobs" list. Job's State will display a progress bar until completion.

Once job is complete, the State will either show as Success or Failed.

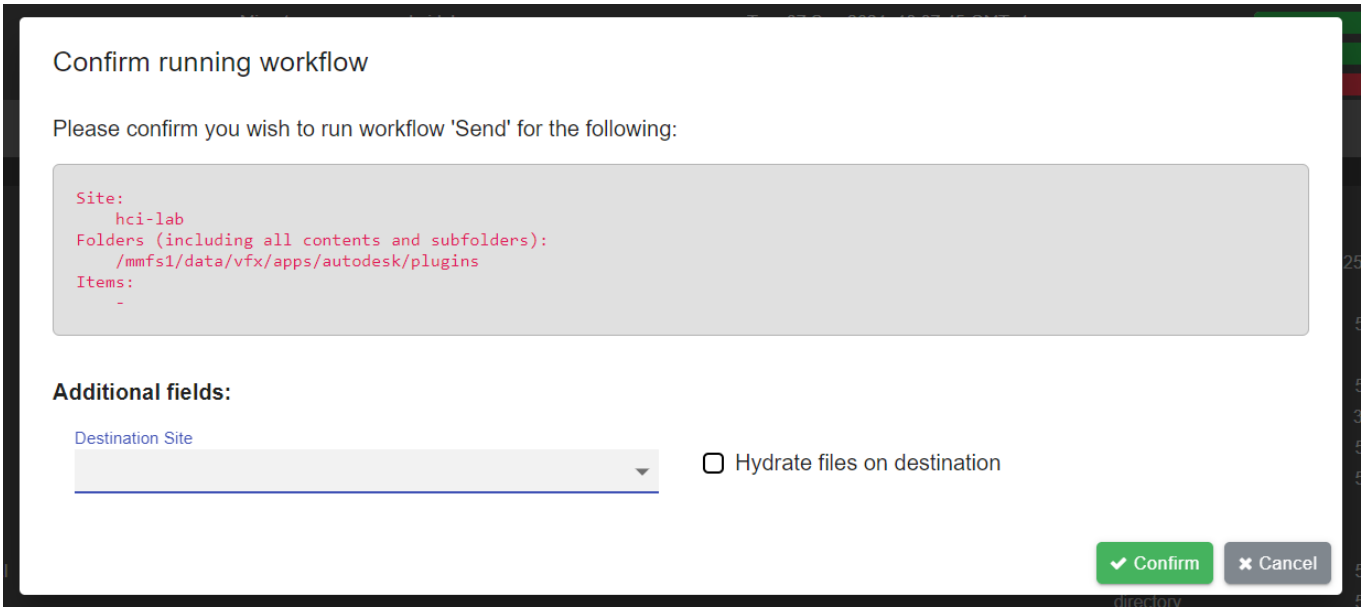
Send

Premigrate behaviour change: Prior to Ngenea Hub 1.8.0, the send workflow would migrate data on the source site. This has been changed to pre-migrate.

To Send a directory or file from one Site to another, expand the Site containing said directory and file. Select the directory or file you wish to Send by ticking their relevant boxes.

Click the "Actions" button at the top right hand side of the page and select "Send".

Select the Site you wish to send the directory and/or files to. Tick the "Hydrate files on destination" if required, and click "Confirm".



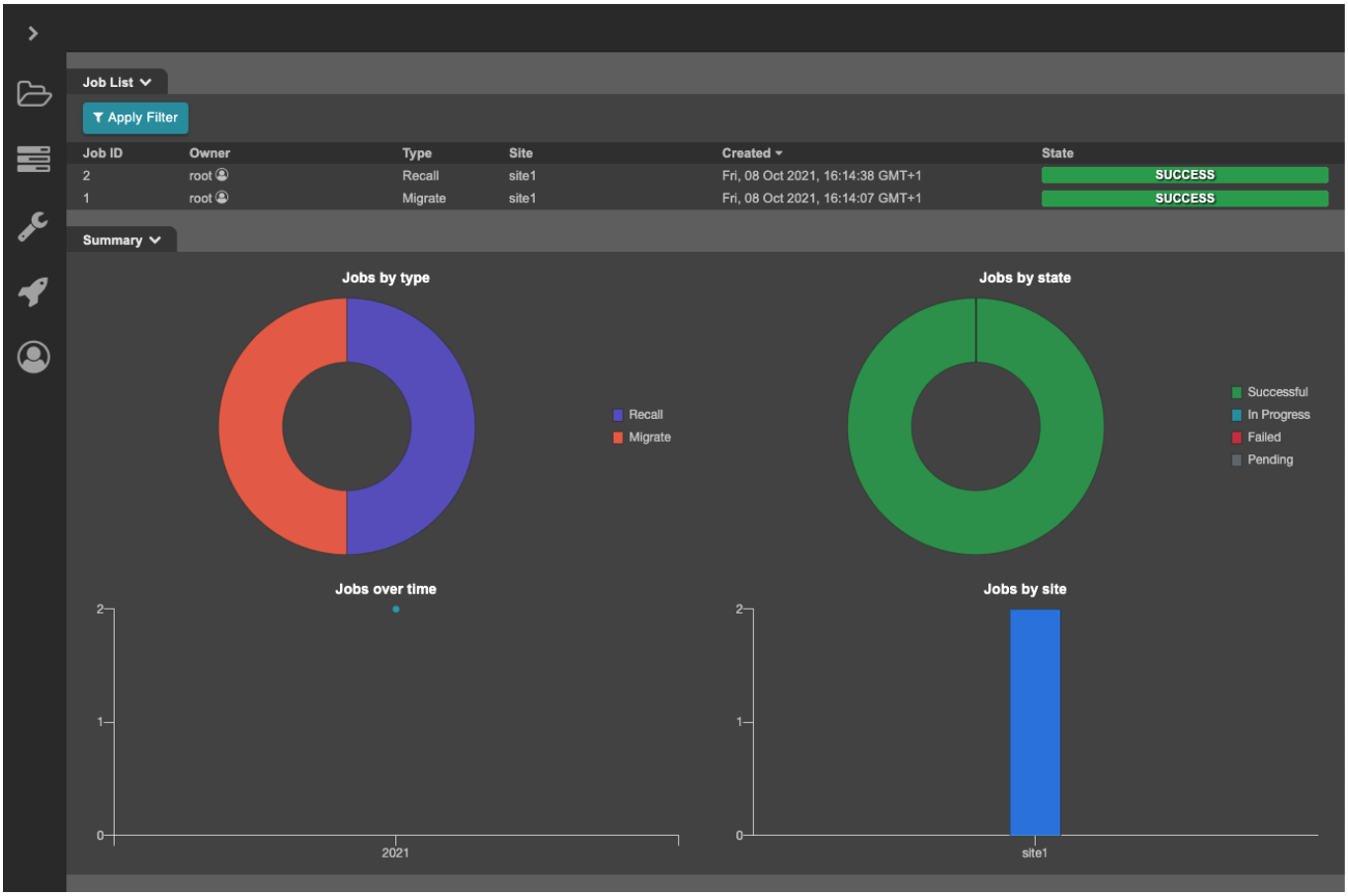
A new Job is created and it is shown at the top of the "Recent Jobs" list. Job's State will display a progress bar until completion.

Once job is complete, the State will either show as Success or Failed.

Expanding the receiving Site's Directories now shows the path that was replicated from the sending Site.

Jobs

The Jobs page shows a list of all the jobs that were initiated via the UI.



The view presents several columns:

Option	Description
Job ID	Shows the identification number associated with the job
Owner	Name of the user who created the job
Type	Shows the job's type, i.e. Migrate, Recall, Premigrate, or Send
Site	Name of the Site where files were migrated from
Created	Shows the job's creation time
State	Shows the job's state, i.e. success or failure

Each column can be sorted in ascending and descending order.

Pagination

To select whether to view 20, 50, or 100 Jobs at the time, choose the relevant option in the "Items per Page" dropdown.

Clicking on the right and left arrow next to "Items per Page" will take you to the next/previous pages.

Apply Filter

Jobs list can be filtered by time period, job type, and job state.

To filter the list, select the "Apply Filter" button on the top left hand side of the UI.

Select one or a combination of filters, and click "Apply".

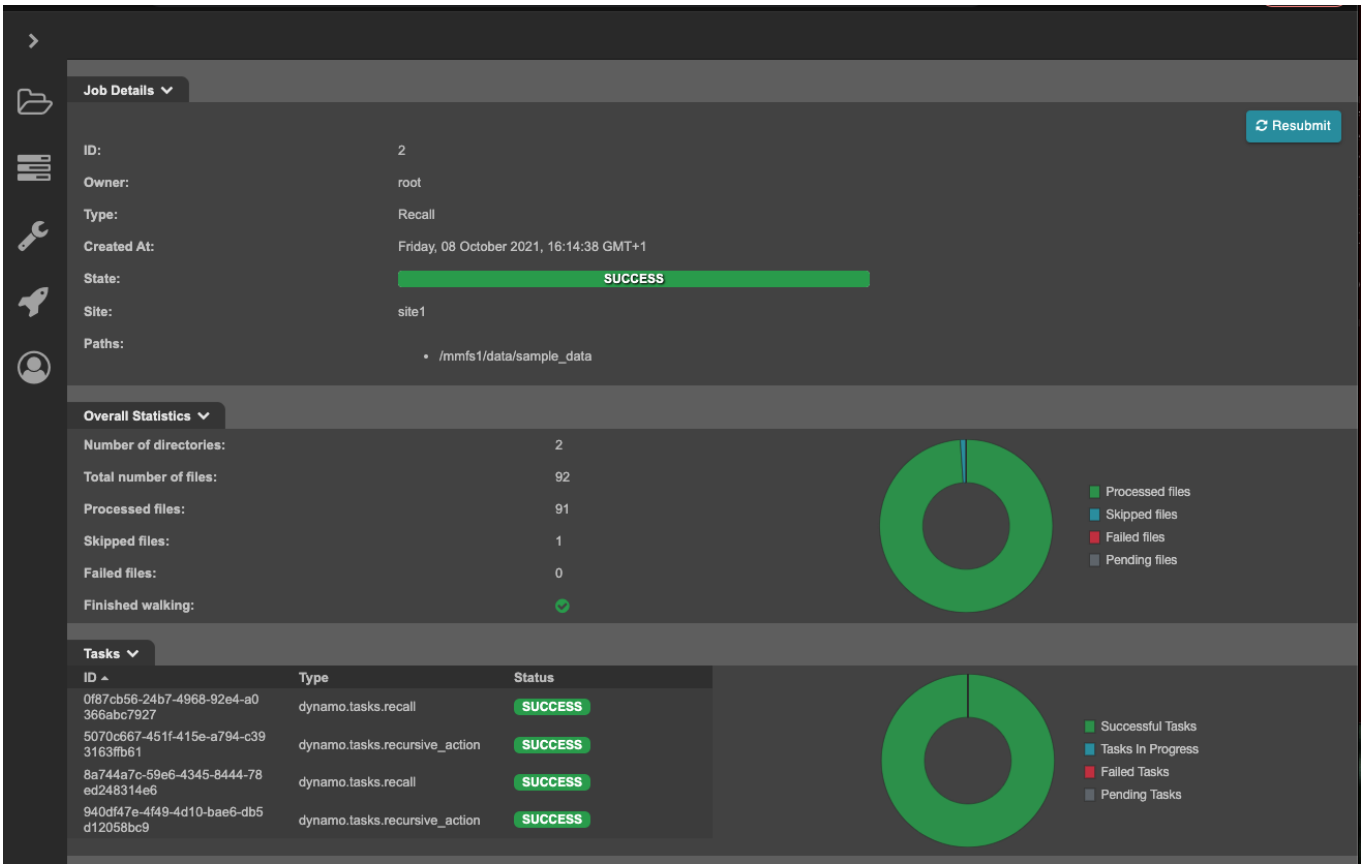
Job are now filtered as per your selection.

To remove a filter, simply select the "x" next to the applied filter.

Job ID

On the Jobs page, click on a Job ID to see additional information regarding the Job.

The Job Details, Overall Statistics, and Tasks tab are displayed.



Each tab shows specific Job details, some of which are clickable:

- Overall Statistics --> Total number of files, processed files, skipped files, and failed files.
- Tasks --> ID

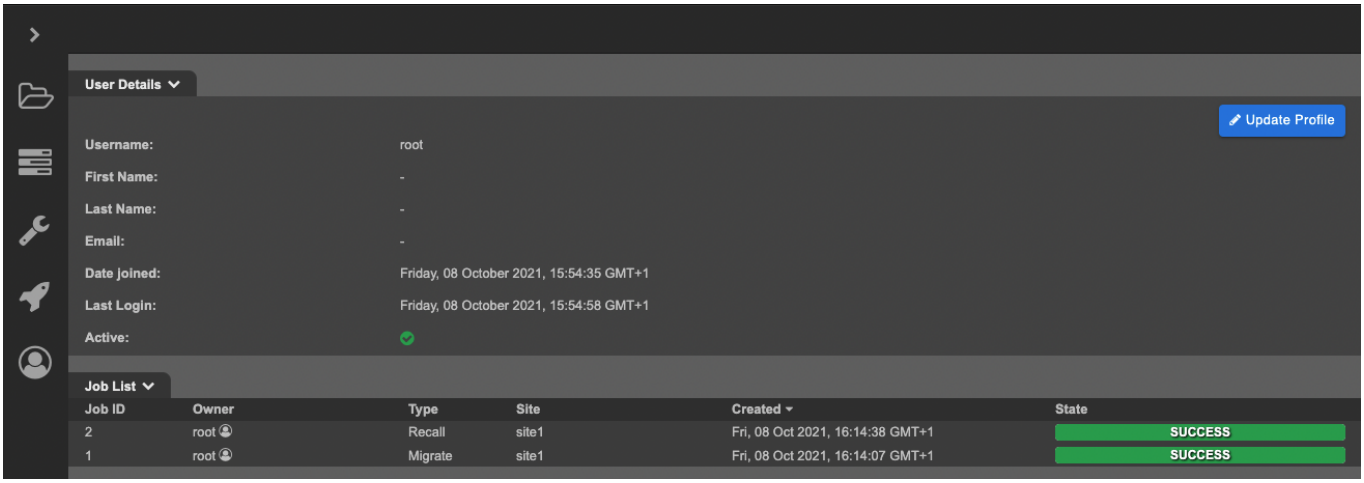
Selecting any of the clickable items opens a dialogue showing the relevant output.

Jobs can also be resubmitted by clicking the "Resubmit" button at the top right hand side of the page.

Owner

On the Jobs page, click on any Owner to see additional information regarding the user who initiated the Job.

Selecting an Owner takes to a page that shows details about the user, as well as a list of Jobs initiated by said user.



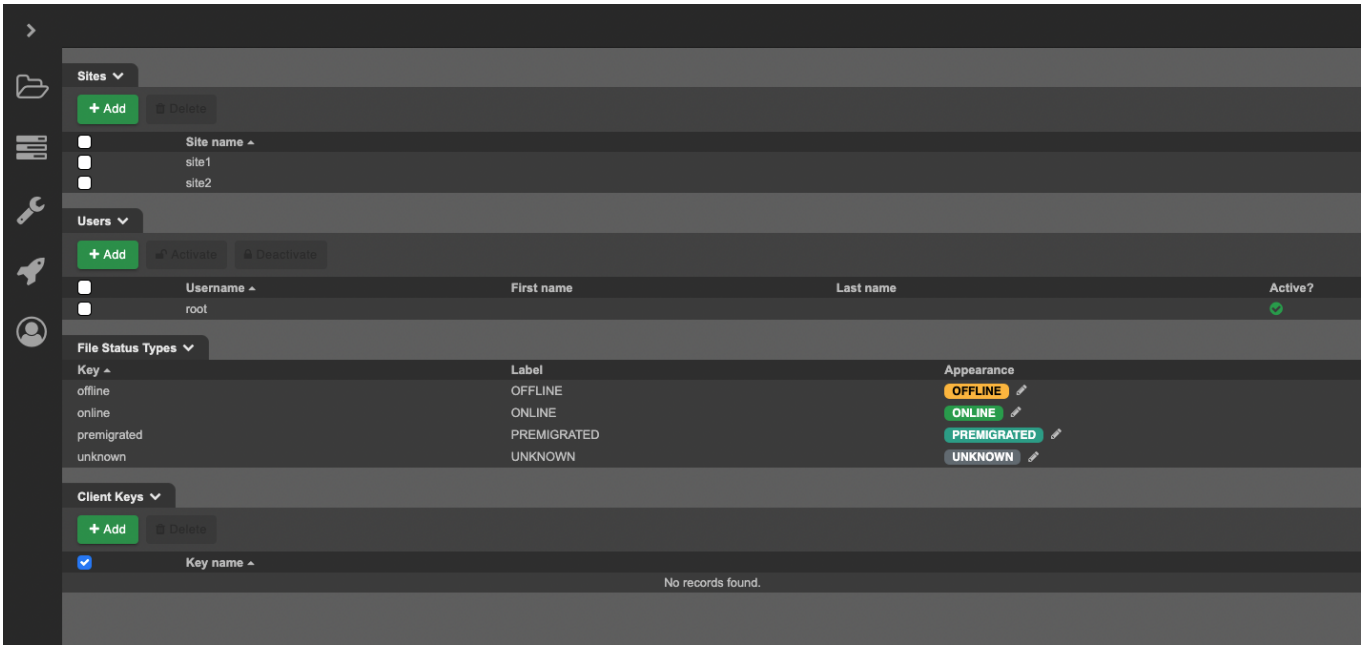
If you selected your own user, you will see an "Update Profile" on the top right hand side of the page.

This takes you to the "Update User" page where you can change your own password, email, first name and last name.

The Job List's layout is the same as the one shown in Jobs page and has the same functionalities.

Administration

The Administration page allows you to add and delete Sites, add, activate and deactivate Users, as well as changing the name and colours of the labels found in the Browse Directories section.



Sites

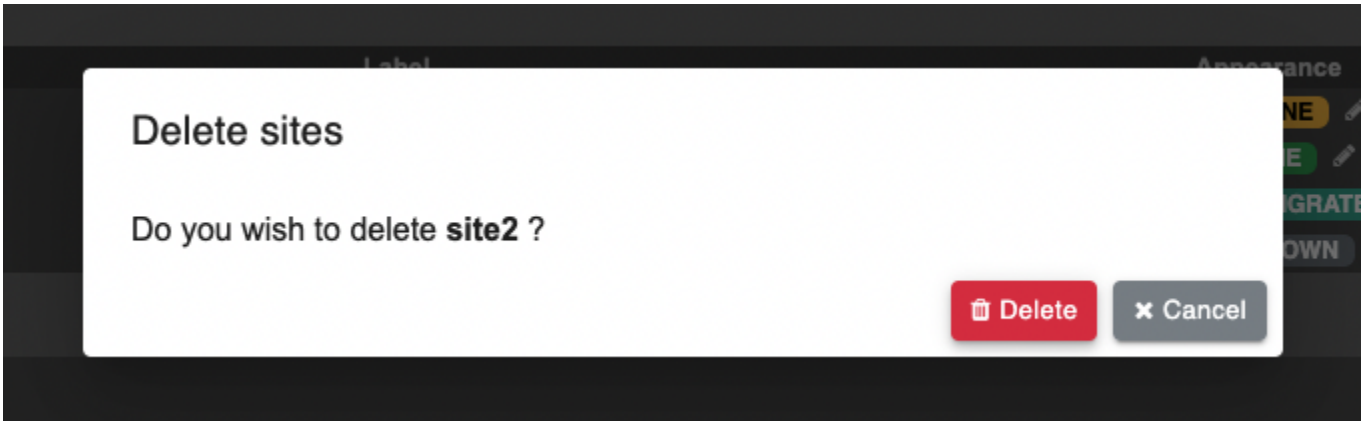
The Sites tab contains the list of Sites that have been configured. This list can be sorted by Site name in ascending and descending order.

Add Site

To add a Site, select the "Add" button, enter a Site name, and confirm by clicking "Add"

Delete Site

To delete a Site, select one or multiple Sites by ticking their correspondent boxes. Click "Delete" and then confirm deletion once the "Delete Site" dialogue is presented.

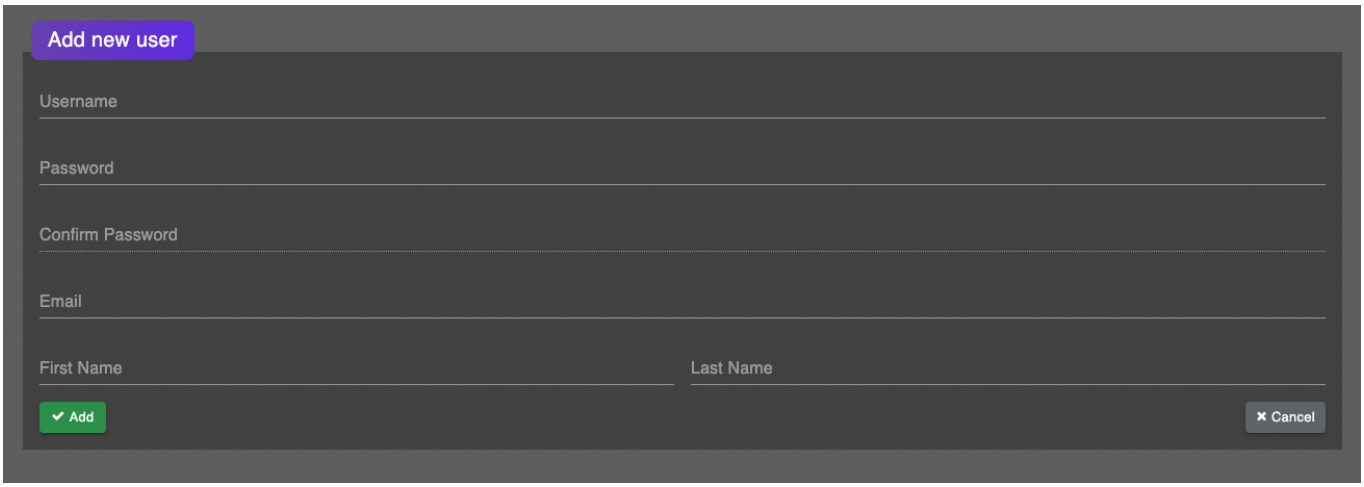


Users

The Users tab contains the list of Users who are allowed to use the UI. This list can be sorted by Username in ascending and descending order.

Add User

To add a new User, select the "Add" button, then enter a Username, Password, Email, First Name, and Last Name.



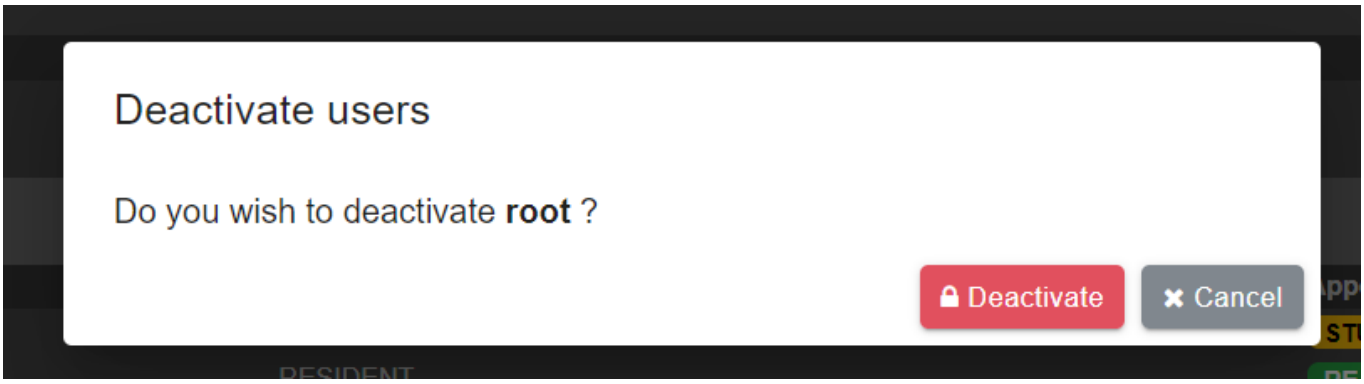
Confirm by clicking "Add".

Activate User

To activate an existing User, select the "Activate" button, and then confirm activation once the "Activate user" dialog is presented.

Deactivate User

To deactivate a User, select one or multiple Users by ticking their correspondent boxes. Click "Deactivate" and then confirm deactivation once the "Deactivate user" dialogue is presented.



User will no longer have access to the UI but still exists and can be reactivated at any time.

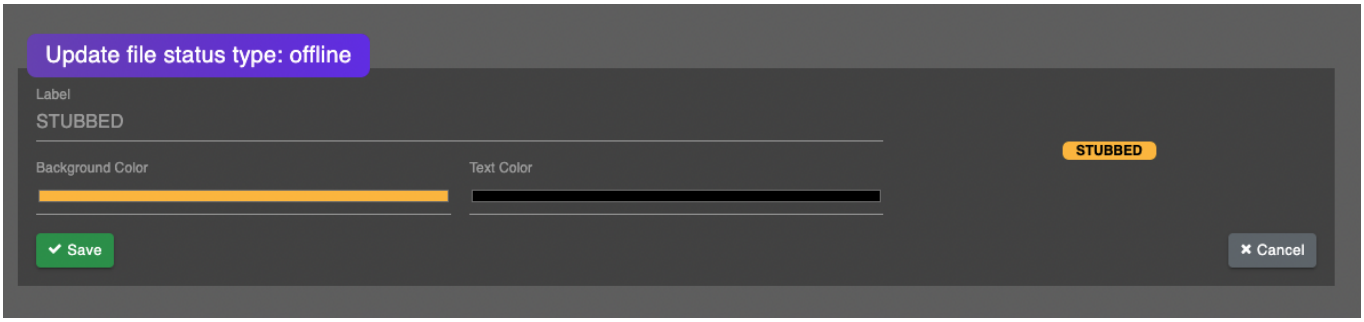
File Status Types

The File Status Types tab lists the labels that are used to indicate a file's status.

The labels are customisable as both colour and label name can be changed.

To change the appearance of a label, select the pencil symbol next to the label that needs updating.

"Update File Status Type" page is displayed:



Change label name, background colour, or text colour as per your preference.

Click "Save" to confirm the update.

API

Reference

The API reference can be found at your Ngenea Hub install at `/api/docs/`. This section does not attempt to duplicate the reference, but instead provide some usage examples.

Authentication

Authentication to the API can be performed in 2 ways

- JWT Authentication
- Client Keys

The first one is for interacting with the API interactively and is therefore most likely not suitable for building automated workflows. On the other hand, client keys are valid until they are revoked and are more suitable for automation.

JWT Authentication

To use the API directly, authentication tokens should be generated to prevent sending the username and password repeatedly. You need to generate these tokens by sending you username-password pair to the login endpoint: `/auth/token`

```
curl -s -X POST 'http://localhost:8000/api/auth/token/'  -H 'Accept:
application/json'  -H 'Content-Type: application/json' -d
'{"username": "dfoster",  "password": "******"}' | jq -r
{
  "access": <access_token>,
```

```
  "refresh": <refresh_token>,
}
```

There are 2 types of authentication tokens:

- Access token
- Refresh token

Access tokens are used for doing API requests. You need to include the token in the Authorization header to use any other endpoint:

```
curl -s -X GET 'http://localhost:8000/api/jobs/'  -H 'Accept:
application/json'  -H 'Content-Type: application/json'  -H
"Authorization: Bearer $JWT_ACCESS_TOKEN" | jq
{
  "count": 0,
  "next": null,
  "previous": null,
  "results": [],
  "stats": {
    "type": {
      "migrate": 0,
      "premigrate": 0,
      "recall": 0
    },
    "state": {
      "SUCCESS": 0,
      "FAILURE": 0,
      "STARTED": 0,
      "PENDING": 0,
      "ERROR": 0
    },
    "created": {},
    "site": {}
  }
}
```

On the other hand, refresh tokens are used for refreshing the access token. For security purposes, access tokens expire in 1 hour and refresh tokens expire in 1 day. When an expired token is used, one of `HTTP 401 Unauthorized` and `HTTP 403 Forbidden` errors is received. In that case, you need to refresh the access token with `/api/token/refresh/` endpoint:

```
curl -s -X POST 'http://localhost:8000/api/auth/token/refresh/'  -H
'Accept: application/json'  -H 'Content-Type: application/json' -d
'{"refresh": "<refresh_token>"}' | jq -r
{
  "access": <new_access_token>,
}
```

Refresh tokens can also be expired too. In that case, you need to send your credentials (username and password) again to obtain new token pair.

Client Keys

Creating Client Keys

> **Note:** The UI does not currently support creating client keys and therefore have to be done via the API directly

Before we can authenticate using the client key, we need to temporarily authenticate using JWT to be able to create a client key.

To get a valid JWT access token using curl and `jq`:

```
export JWT_TOKEN=$(curl -s -X POST 'http://localhost:8000/api/auth/
token/'  -H 'Accept: application/json'  -H 'Content-Type:
application/json' -d '{"username": "dfoster",  "password":
"******"}' | jq -r .access)
echo $JWT_TOKEN
<token>
```

This can now be used to create a client key:

```
curl -s -X POST 'http://localhost:8000/api/auth/clientkeys/'  -H
'Accept: application/json'  -H 'Content-Type: application/json'  -H
"Authorization: Bearer $JWT_TOKEN" -d '{"name": "my_automation_key"}'
| jq '.'
{
  "url": "http://localhost:8000/api/auth/clientkeys/1/",
  "id": 1,
  "name": "my_automation_key",
  "api_key": "YOUR_API_KEY"
}
```

```
{warning} This is the only time the client key will be visible, make
sure it is recorded.
```

Using Client Keys

The key created in the previous section can now be used by setting the header `Authorization: Api-Key YOUR_API_KEY` against an API endpoint. For example:

```
export API_KEY=YOUR_API_KEY

curl -s -X GET 'http://localhost:8000/api/jobs/'  -H 'Accept:
application/json'  -H 'Content-Type: application/json'  -H
"Authorization: Api-Key $API_KEY" | jq
{
  "count": 0,
  "next": null,
  "previous": null,
  "results": [],
  "stats": {
```

```
    "type": {
      "migrate": 0,
      "premigrate": 0,
      "recall": 0
    },
    "state": {
      "SUCCESS": 0,
      "FAILURE": 0,
      "STARTED": 0,
      "PENDING": 0,
      "ERROR": 0
    },
    "created": {},
    "site": {}
  }
}
```


Submitting Workflow

To submit a workflow, the following parameters are required:

name	description
`workflow`	The name of the workflow to submit
`paths`	A list of paths to execute the workflow on
`site`	The name of the site where the workflow should be started from. Steps within a workflow may run on different sites.

In addition, the following optional parameters may be provided:

name	description
`discovery`	Name of the file discovery technique to use. Currently the only supported discovery is `recursive`. If no discovery is specified, `recursive` will be used as the default. If explicitly set to `null`, no discovery will be performed and the provided paths will be used 'as is'.
`fields`	Additional parameter for the workflow, typically used by custom workflows.

Migrate

Using a Client Key stored in a environment variable `TOKEN`, the following is an example of migrating a file using curl.

```
curl -s -X POST 'http://example.com/api/file/workflow/'  -H 'Accept: application/json'  -H 'Content-Type: application/json'  -H "Authorization: Api-Key $TOKEN"  -d '{"paths": ["/mmfs1/data/sample_data.tgz"], "site": "dfoster1", "workflow": "migrate", "discovery": null}'
```

Note: Since we're only migrating a single file, we don't need recursive discovery, so discovery has been set to `null` to disable it.

Monitoring and Management

systemd service

Ngenea Hub is controlled via the `ngeneahub` systemd service

ngeneahubctl cli tool

The `ngeneahubctl` tool can be used to manually stop/start the services, outside of systemd, for debugging

Docker containers

Ngenea Hub uses a collection of docker containers, which can be managed by standard Docker monitoring/management tools and processes:

Container Name	Description
`ngeneahub_app_1`	Web application
`ngeneahub_jobrefresh_1`	Maintenance task controller
`ngeneahub_db_1`	Application database (Postgres)
`ngeneahub_rabbitmq_1`	Task queue broker
`ngeneahub_redis_1`	Task results backend

Health endpoints

To view the state of all sites and related nodes within known to Ngenea Hub, a GET request can be performed to `/api/health` to view all of the sites, nodes and the hub service itself. The states are currently based on how many nodes are online for each site using the following states:

State	Description
`ok`	All nodes are functional
`warning`	Some nodes are offline within a site
`critical`	One or more sites are completely offline

An example output of the health endpoint can be seen below:

```json {
```

```
"overall_health": "ok", "hub_status": {
    "health": "ok"
}, "site_status": [
    {
        "site": "site1", "health": "ok", "nodes": [
            {
                "name":  "pixstor-east-ng-test",  "health":  "ok",
                "online": true
            }
        ]
    }, {
        "site": "site2", "health": "ok", "nodes": [
            {
                "name": "pixstor-west-ng-test", "health":
                "ok", "online": true
            }
        ]
    }
]
}
```

A request can also be performed to specific sites using `/api/sites/ID/health/` to view the site specific health status:

```json {
    "site": "site1", "health": "ok", "nodes": [
        {
            "name": "pixstor-east-ng-test", "health": "ok", "online": true
        }
    ]
}
```

Custom Workflows

Defining workflows

It's possible to define custom workflows which use pre-defined rules as building blocks to create your workflow.

> **Note:** Custom workflows are not currently exposed via the UI. Use the API `/api/workflows/` endpoint to create custom workflows

A workflow definition requires the following parameters:

Name	Description
`name`	The unique name for this workflow. For easy of submission again the API, this should not contain spaces.
`label`	The human readable name for this workflow, can contain spaces.
`icon_classes`	List of icon classes to represent the workflow in the UI. Font Awesome is useful here.
`filter_rules`	A list of rules to apply to provided files that match defined states. Described in more detail below.
`fields`	A list of runtime fields. Described in more detail below.

Additionally, you can optionally provide:

Name	Description
`discovery`	Which discovery task the workflow should be used by default, this can be either recursive or snapdiff.

Filter Rules

Filter rules are defined in JSON. They are a list of individual rules in a mapping format that will be performed on each matching file result when a discovery task is complete. If called through the API with no discovery task provided, rules will be applied to any states provided in the workflow input.

Steps are defined in JSON. Steps is a list of individual steps that will be performed serially. Each rule must contain the following:

Name	Description	Required
`state`	The state of a result provided by the discovery task with any given path, an example of that could	Yes

Name	Description	Required
	be "processed" or "modified" more details about this are in the discovery section.	
`type`	The type of result the rule will apply to, the only valid types are: `file|directory|symlink|all`	Yes
`action`	A list of tasks to perform on files that match the state and type	Yes
`include`	A list of globs to apply to provided files to limit actions to just them.	No
`exclude`	A list of globs to apply to provided files. Described in more detail below.	No
`ignore_site_includes`	Whether to ignore any global includes defined on the site the workflow will run on	No
`ignore_site_excludes`	Whether to ignore any global excludes defined on the site the workflow will run on	No

These rules control which actions will be performed on certain files based on their given state that they have been given following specific discovery tasks such as `snapdiff` or provided in the initial input of a workflow. These states can allow direct control of workflows performed on files provided, allowing multiple workflow paths within the same job by utilizing multiple rules controlling specific states with additional control with include and exclude path rules.

Alongside rules bound to a state, there are two special states that rules can be used, these being `default` and `all`. Rule sets cannot have both `default` and `all` rules within them, but it is possible to have multiple of one type with different sets of exclude and include rules to allow for more granular control.

Rules defined with `default` as their `state` and `type` will perform their action on paths that have not been captured by all other rules within a given rule set. This means that if there are specific file states that need to be actioned differently, paths that do not match any other rules actioned against without ignoring those non-matching paths.

The other special rule type is rules with the `state` and `type` of `all`. This rule will perform its action on all paths regardless of their provided type and state. This is an additional operation so if another rule has an explicit rule provided it will perform multiple actions on the same path, for each matching rule in rule set. Simple

workflows are typically composed of a single rule with the state and type of `all` as this will simply process all paths provided to it.

Within each rule, there must be a list of actions to perform on the resulting file provided within the `action` key. These actions will be performed serially. Each action must be a mapping that contain the following in each entry:

Name	Description	Required
`name`	The name of the task to run, e.g. `dynamo.tasks.migrate`	Yes
`site`	The name of the site to run against, if this is not provided it will use the site provided within the workflow call.	No

If steps have optional arguments, these can be passed as additional key:value pairs in these step definition mapping to pass those optional arguments.

As an example we can define a generic rule that captures every type of file and state and sends it to a second site, this would be useful for a bulk move using the `recursive` discovery task to cover all types of files in directories provided to the task:

Example 1 - Send to london

```
{
    "state": "all",
    "type": "all",
    "action": [
        {
            "name": "dynamo.tasks.migrate"
        },
        {
            "name": "dynamo.tasks.reverse_stub",
            "site": "london"
        }
    ]
}
```

Runtime fields

A workflow needs to be able to accept parameters as it submitted. Taking example #1 above, "london" doesn't want to be hardcoded as the destination site, as that would mean a new workflow would need to be defined for each possible destination.

Instead, fields can be defined, that in turn will need to be provided at workflow submission time. Fields are defined as a mapping with the following keys:

Name	Description	Required
`name`	The name of the field.	Yes
`label`	The friendly name for this field, used for presenting in the UI	Yes
`type`	The type of the field, valid options are: • `string` - a free text field • `int` - a free text field that will be validated a integer • `bool` - a checkbox • `choices` - A dropdown box representing a list of choices, populated from choices list of objects. • `enum[enum_type]` - A dropdown box representing a choice of option, populated from enum_type. `enum_type` can be one of the following ◦ `site` - A list of all the sites Ngenea Hub has defined	Yes
`default`	The default value for runtime fields	optional

The following is an example of a custom field definition for providing a site to an action step:

Example 2 - Custom field definition

```
[
    {
        "name": "target_site",
        "label": "Site to migrate to",
        "type": "enum[site]"
    }
]

---
caption: Custom field defintion with default value
---
[
    {
        "name": "target_site",
        "label": "site to migrate to",
```

```
        "type": "enum[site]",
        "default": "london"
    }
]
```

If default value is specified in runtime fields, it will take the default value for fields while running workflow if the user input is not given otherwise it will always use the user input.

Back in the definition of an action step, any value that is prefixed with a `*` will be used as a field name and the value replaced instead of a literal string.

The following example, modifies example #1 to use the custom field as defined in example #3:

Example 3 - Updated rule now using custom fields

```
{
    "state": "all",
    "type": "all",
    "action": [
        {
            "name": "dynamo.tasks.migrate"
        },
        {
            "name": "dynamo.tasks.reverse_stub",
            "site": "*target_site"
        }
    ]
}
```

So, a complete request to create a workflow that will process all file and state types with a dynamic "site" field will look like:

Example 4 - Full workflow request

```
{
    "name": "send_file",
    "label": "Send files from one site to another",
    "icon_classes": ["fa fa-cloud fa-stack-2x text-primary", "fa fa-
refresh fa-stack-1x text-light"],
    "filter_rules": [
        {
            "state": "all",
            "type": "all",
            "action": [
                {
                    "name": "dynamo.tasks.migrate"
                },
                {
                    "name": "dynamo.tasks.reverse_stub",
                    "site": "*target_site"
                }
            ]
        }
```

```
    ],
    "fields": [
        {
            "name": "target_site",
            "label": "Site to migrate to",
            "type": "enum[site]"
        }
    ]
}
```

The following is an example of a custom field definition for providing a choices to an action step:

Example 5 - Custom field definition

```
[
    {
        "name": "sync_policy",
        "label": "sync_policy",
        "type": "choices",
        "choices": [
            {
               "label": "Newest",
               "value": "newest"
            },
            {
               "label": "Sourcesite",
               "value": "sourcesite"
            }
        ]
    }
]

choices support both string and integer type values.
```

Back in the definition of an action step, any value that is prefixed with a `*` will be used as a field name and the value replaced instead of a literal string.

The following example, uses the custom field in action:

Example 6 - Updated rule now using custom fields

```
{
    "state": "all",
    "type": "all",
    "action": [
        {
            "name": "dynamo.tasks.migrate"
        },
        {
            "name": "dynamo.tasks.reverse_stub",
            "sync_policy": "*sync_policy"
        }
    ]
}
```

So, a complete request to create a workflow that will process all file and state types with a static choices field will look like:

Example 7 - Full workflow request

```
{
    "name": "send_file",
    "label": "Send files from one site to another",
    "icon": "<span class='fa-stack'><i class='fa fa-cloud fa-
stack-2x text-primary'></i><i class='fa fa-angle-right fa-stack-2x
text-light'></i></span>"
    "filter_rules": [
        {
            "state": "all",
            "type": "all",
            "action": [
                {
                    "name": "dynamo.tasks.migrate"
                },
                {
                    "name": "dynamo.tasks.reverse_stub",
                    "sync_policy": "*sync_policy"
                }
            ]
        }
    ],
    "fields": [
        {
            "name": "sync_policy",
            "label": "sync_policy",
            "type": "choices",
            "choices": [
               {
                  "label": "Newest",
                  "value": "newest"
               },
               {
                  "label": "Sourcesite",
                  "value": "sourcesite"#
               }
            ]
        }
    ]
}
```

Running Workflows

Once a workflow has been defined, it can be performed through the file browser by selecting files and directories and clicking the actions button. It is then possible to select the workflow you wish to call, this workflow call will not use a discovery task unless a directory is selected, in that case it will make use of the `recursive` discovery step.

This can also be performed via a POST request to `/api/file/workflow`. When called through the API, you have the option to provide a discovery step, these steps can expand the initial paths provided to them to either recursively perform actions or perform something like a file difference scan.

Name	Description	Type	Required
`paths`	A list of paths to perform the workflows against, these can be just strings of file absolute file paths or can be JSON with the keys of "path" and "state", detailed example in example 7	JSON List	Yes
`site`	The site to perform the workflow against	String	Yes
`fields`	The runtime fields for a workflow	String	Yes
`discovery`	The discovery phase to use for this workflow run, this will override any defaults	String	No
`job`	The ID of a job that this workflow should be run within	Integer	No

Following the example workflow defined above, you can call the workflow to recursively send all files within any paths provided using the following POST to `/api/file/workflow`:

Example 8 - Calling example workflow

```
{
    "paths": [
        "/mmfs1/data/project_one",
        "/mmfs1/data/project_two"
    ],
    "site": "london",
    "workflow": "send_file",
    "discovery": "recursive",
    "fields": {
        "target_site": "dublin",
    }
}
```

This will now migrate all files within `/mmfs1/data/project_one` and `/mmfs1/data/project_two` and then recall them at the site defined as `dublin`.

If there is a more complex workflow that have been defined that includes rules for specific states, the input paths can include this state information. This behaviour can be only be used when no discovery state is provided, an example of a custom rule set using could be:

Example 9 - Calling workflow with state data

```
{
    "name": "migrate_state",
    "label": "Stateful file migration",
    "filter_rules": [
        {
            "type": "all",
            "state": "modified",
            "action": {
                "name": "dynamo.tasks.migrate"
            }
        {
            "type": "all",
            "state": "moved",
            "action": {
                "name": "dynamo.tasks.delete_paths_from_gpfs"
            }
        }
    ],
    "discovery": null,
    "fields": []
}
```

Here is a simple rule set that will migrate all paths provided with the state `modified` and will delete all paths provided with the state `moved`. With this example workflow provided you can perform a POST to `/api/file/workflow` with the following JSON:

Example 10 - Calling workflow with state data

```
{
    "paths": [
        {
            "path": "/mmfs1/data/project_one",
            "state": "modified"
        {
            "path": "/mmfs1/data/project_two",
            "state": "moved"
        }
    ],
    "site": "london",
    "workflow": "migrate_state",
    "discovery": null,
    "fields": {}
}
```

Using multiple state based rules with different include and exclude path filters, you could achieve more complex behaviour in workflow calls for more finite control.

Discovery Steps

Discovery steps can make complex large bulk operations much more manageable to call, allowing you to provide a single path that expands to cover all the contents of a path, or to see time based differences for a given path.

> **Note:** If a workflow is submitted without a discovery task explicitly provided, it will default to using the discovery task defined as the default during the workflow's creation, visible via the workflow's "discovery" attribute. To avoid this, it is possible to explicIty pass `null` as the discovery task via the API to skip any discovery phase and additional processing on the paths provided and instead process the actions specified using the rules, without any additional checks.

Name	Description	Supported states
`recursive`	Performs a recursive expansion of the initial provided paths. This allows paths to be expanded to cover all sub file and directories, it will then perform the defined action for all the generic rules in a workflow against all resulting files.	`all`
`snapdiff`	Performs a time based file scan on an independant fileset between the last time a scan was performed. It will retrieve all file differences between those moments in time and the state of that file.	`created|updated|moved|deleted|all`

For more complex discovery steps such as `snapdiff`, there are defined states that files, directories and links can be in once it has completed its scan. This allows more explicit control of file and state control within a single call to a workflow. If for example you want all results with the type of `file` that have the state `created` to be sent to another site without any temporary files, a rule to cover that could be:

Example 11 - Custom rule for filtering snapdiff discovery results

```
{
    "state": "created",
    "type": "file",
    "exclude": ["*.temp"],
    "action": [
```

```
        {
            "name": "dynamo.tasks.migrate"
        },
        {
            "name": "dynamo.tasks.reverse_stub",
            "site": "*target_site"
        }
    ]
}
```

Includes and Excludes

Includes and excludes can be used to select paths that individual filter rules should apply to, or generally limit which paths should be handled during a workflow run.

Include/exclude patterns behave like unix shell pattern matching ('globbing'). The 'wildcard' asterisk character `*` will match any characters within a string. Patterns must match whole paths; partial matches are not supported, except through the use of wildcards.

Includes and excludes are combined as "a path matching any includes and not any excludes". For example `{"include": ["/mmfs1/data/*"], "exclude": ["*.tmp"]}` would match only files in `/mmfs1/data`, but not files in that directory with the `.tmp` extension. If no includes are defined, then all files are considered included (unless explicitly excluded).

There are three places where path include and exclude patterns can be defined:

- on a site
- within a filter rule
- at runtime, when a workflow is submitted

Site patterns can be used to apply includes and excludes globally, to all workflows and workflow steps. If defined, these will be appended to any patterns defined within a filter rule or at runtime.

For example, if a rule defines `{"exclude": ["*.tmp"]}` and the site defines `{"exclude": ["*.cache"]}`, then the combined excludes for that rule would be `{"exclude": ["*.tmp", "*.cache"]}`

If not desired, this behaviour can be overridden by specifying `ignore_site_includes` / `ignore_site_excludes` either on a per-rule basis, or for all rules by passing those parameters when submitting a workflow run.

For workflows involving multiple sites, such as send and sync, only the primary (source) site patterns will be considered.

If includes and excludes are passed when submitting a workflow, they will be applied to all filter rules within the workflow, replacing any patterns already defined within the workflow rules. Any site patterns will be appended to the runtime patterns, unless 'ignore' is specified.

Site Sync

Ngenea Hub provides the facility for syncing data from one site to another.

Site Sync applies changes in one direction. See also Bidirectional Site Sync for applying changes in both directions.

Synchronisation is achieved by utilising a scheduled workflow which periodically discovers file and directory changes within an Independent Fileset on a source site and applying those changes to a target site. Changes are applied by sending newly created or recently modified files and directories, including deleting or moving files or directories in place on the target site as necessary to match the source site.

The following walkthrough details how to set up a sync between two sites ('Site Sync') using the Ngenea Hub UI.

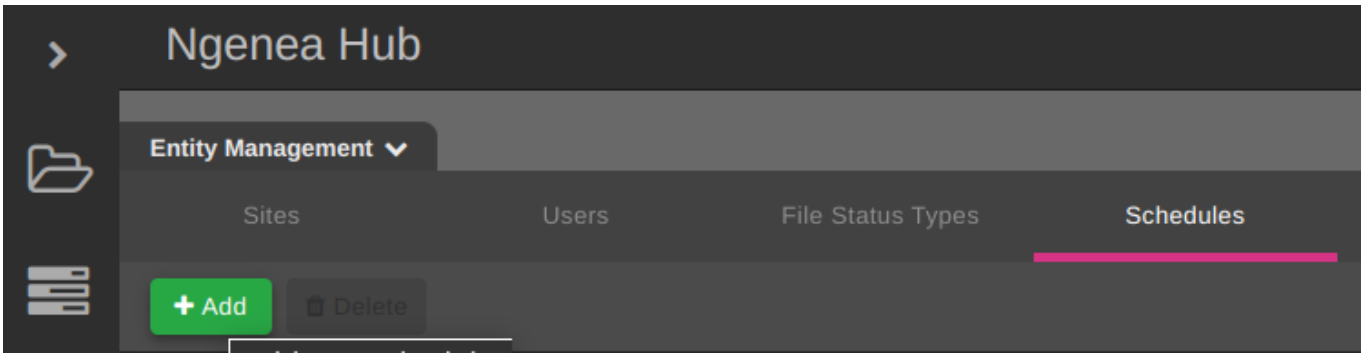
The REST API can also be utilised to achieve the same setup.

Schedule

A schedule defines when and how often a sync occurs.

In the Ngenea Hub UI, navigate to the `Administration` page, and select the `Schedules` tab.

To create a new schedule, click the `Add` button.



Configure the schedule by entering appropriate details into the `Add new schedule` dialog:

1. Enter a descriptive `Name` for the schedule, E.G. "sync-to-site2"
2. Select the `source` site from the dropdown menu
3. Set the `Discovery` to `snapdiff`. Snapdiff discovery uses snapshots to track changes over time within an Independent Fileset.
4. Set the `Path` to the top level directory (path) of the Independent Fileset to be synchronised
5. Set the time criteria
6. Click the `Add` button to add the new schedule

Time criteria

A schedule utilises cron syntax. By default, all the fields are populated with `*` - this means the schedule will run every minute. This is likely too frequent as the calculation of file changes could take longer than the time between the schedule running every minute. Sizing for an ideal setup will balance the need to sync files quickly versus how long the sync snapdiff stage takes to run. This can be most effectively achieved through observation.

Examples:

- To enact the schedule every 5 minutes, specify the `Minute` field as follows: `*/5`.
- To enact the schedule once an hour, set the `Minute` field to a number of minutes past the hour. Setting `Minute` to `5` will run the sync once an hour at 5 minutes past the hour.
- To enact the schedule at set times throughout the hour, change the `Minute` field separating entries by commas (E.G.) `7,23,46`

Workflow

Schedules enact a workflow. A workflow performs one or more series of tasks to files and/or directories.

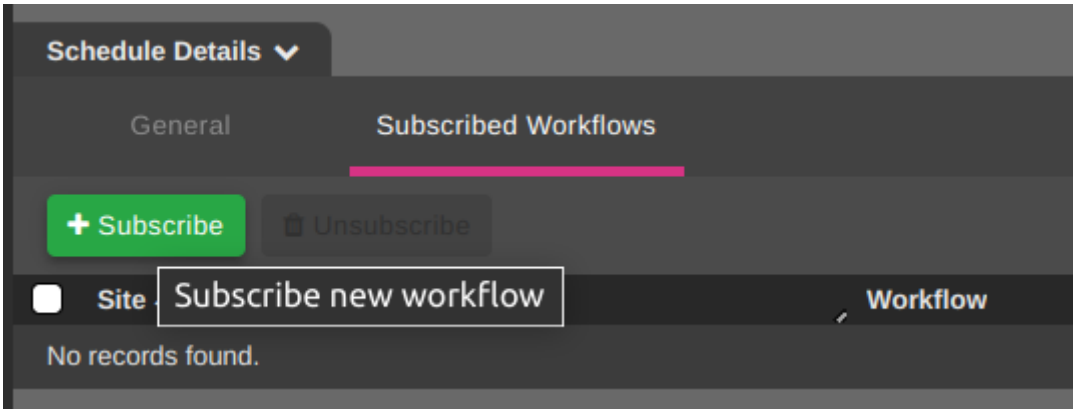
In the Ngenea Hub Hub UI, navigate to the `Administration` page, and select the `Schedules` tab.

Select the prior created schedule in the `Schedules` tab, which then displays the `Schedule Details` page.

Subscribing a workflow

To associate a workflow with a schedule undertake the following actions:

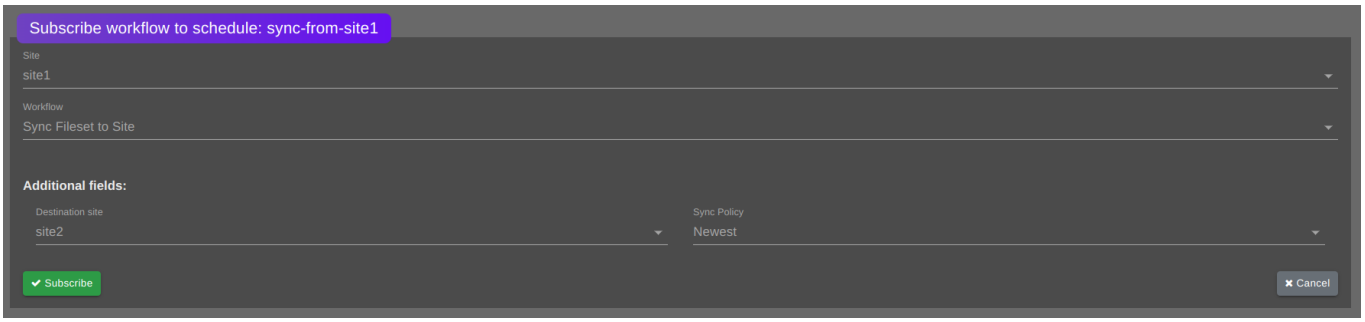
1. Click `Subscribe` and in the create dialogue set `Site` to the source site from which to synchronise.



Ngenea Hub provides a built-in workflow for synchronising changes from an Independent Fileset on a source site to a target site.

1. Select `Sync Fileset to Site` from the `Workflow` dropdown menu.

Configure the workflow subscription settings by entering appropriate details into the `Subscribe workflow to schedule` dialog:



1. Select the target site in the `Destination site`
2. Select an appropriate `Sync Policy`
3. Click `Subscribe` to associate the workflow to the schedule

Sync Policy

The `Sync Policy` setting controls how conflicts are resolved. A conflict may occur if (E.G.) different versions of the same file exist on both source and target sites. By default, the newest version of the file will be retained.

Outcome

Now associated (subscribed), files and/or directories will synchronise at the next scheduled execution time. When executing the scheduled workflow, two jobs appear within the schedule details page. One for the snapdiff discovery, and one for the subscribed sync workflow.

The first enactment of the sync will synchronise all files and directories in the Independent Fileset from the source site. Further enactments will synchronise files and/or directories created, modified, deleted or moved since the last enactment and the time of the most recent enactment.

Additional Options

Associating more than one workflow to the same schedule can be utilised to synchronise from the same source site to another target site. Additional workflows cause creation of an additional job, and run simultaneously with other syncs enacted from the same schedule.

Troubleshooting

If you encounter issues with sync, refer to the sync troubleshooting guide page for guidance.

Bidirectional Site Sync

Ngenea Hub provides the facility for syncing data between sites.

Bidirectional Site Sync is similar to Site Sync, but applies changes in both directions. If a Bidirectional Site Sync runs between sites A and B, changes on site A will be applied on site B, followed by changes on site B being applied on site A.

Synchronisation is achieved by utilising a scheduled workflow which periodically discovers file and directory changes within an Independent Fileset on each of the sites and applies those changes to the other site. Changes are applied by sending newly created or recently modified files and directories, including deleting or moving files or directories in place on the target site as necessary to match the other site.

A prerequisite for a working Bidirectional Site Sync is that the Independent Fileset has been created on both sites.

The principles of setting up Bidirectional Site Sync are similar (but with some differences explained below) to setting up a Site Sync, so please refer to the walkthrough on the Site Sync page for detailed instructions. It can be done using the Ngenea Hub UI or the REST API.

Schedule

Discovery

Create a new schedule. Unlike with Site Sync, for Bidirectional Site Sync set the schedule's `Discovery` to `bidirectional_snapdiff`.

This discovery type operates identically to the `snapdiff` discovery type, with snapshots used to track changes over time within an Independent Fileset.

A separation is enforced between `bidirectional_snapdiff` schedules and schedules with other discovery types. Whereas the `Discovery` of other schedules may be changed, a change from or to the `bidirectional_snapdiff` discovery type is not allowed. This is to ensure consistency of the specific workflow rules that apply to Bidirectional Site Sync.

Time criteria

Unlike with Site Sync, a Bidirectional Site Sync will not be triggered if the previous sync is still running. This means that some syncs will be skipped if scheduled closer together than the time it takes to complete a sync.

Workflow

Subscribing a workflow

Select the `Bi-directional Site Sync` workflow.

This is the only workflow that may be subscribed to a `bidirectional_snapdiff` schedule, and it cannot be subscribed to other types of schedule.

A `bidirectional_snapdiff` schedule may have at most one subscribed workflow.

Outcome

When executing the scheduled workflow, only one job is created and appears within the schedule details page. This job includes the tasks for both the snapdiff discoveries (one on each site), as well as the tasks that enact the sync in each direction.

Troubleshooting

If you encounter issues with sync, refer to the sync troubleshooting guide page for guidance.

Search

> **Note:** Before you can use the search feature, additional set-up is necessary, as described in the search feature page.

The search endpoint provides the ability to search for files across multiple sites, and aggregate the results.

Search is performed in two steps - submitting a query, and retrieving the results.

Submitting a query

Search performs a query by submitting asynchronous tasks to each requested site. The sites then perform the actual search and return results as available.

A search is initiated by `POST`ing a query to the search endpoint

```
curl -s -X POST 'http://example.com/api/search/'  -H 'Accept:
application/json'  -H 'Content-Type: application/json'  -H
"Authorization: Api-Key $TOKEN"  -d '{"path": "/mmfs1/data",
"sites": ["site1"], "recursive": true, "filters": {"hsm.status":
"migrated"}}'
```

The search request payload is made of

name	description
`path`	Directory to query
`sites`	List of one or more sites to search. Default: all sites
`recursive`	Whether to search the `path` recursively. Default: false, only immediate children of `path` will be returned.
`filters`	A collection of filters against arbitrary metadata, see below. Default: None
`metadata_fields`	A list of metadata fields to include in search results. This can include specific field names (e.g. `hsm.status`), or namespace wildcards (e.g. `core.*`) to select all fields in a given namespace. Default: all available fields.
`merge`	If the same file exists on multiple site, this will cause them to be merged in the results (see below). Default: false

Upon successful submission, the request will return status `201 (Created)`, and a response body which includes the url for retrieving search results (see below)

```
{"id":1,"url":"http://example.com/api/search/1/"}
```

Filters

Filters are a collection of filters to apply to arbitrary file metadata.

The specific metadata available to be filtered on depends on the search backend being used. The fields in the following examples may not be available for all backends.

At a minimum, one can expect to be able to filter on `core.filename`, the file basename. For example to filter only jpeg files, `{"core.filename": "*.jpg"}`

Possible filter types are

type	description	example
exact match	match a value exactly	`{"core.filename": "cats-01.jpg"}`, `{"core.size": 0}`

type	description	example
match list	match any of the values in the list (value1 OR value2 OR ...)	`{"core.group.name": ["editor", "admin"]}`
wildcard	any string value containing an asterisk (`*`) is treated as a wildcard	`{"core.filename", "*.jpg"}`
range	numerical or date range, using any combination of less-than (`lt`), less-than-equal (`lte`), greater-than (`gt`), greater-than-equal (`gte`)	`{"core.modificationtime": {"gte": "2021-01-01", "lt": "2021-02-01"}}`, `{"core.size": {"gt": 1000000000}}`
negation	exclude anything matching a given filter	`{"not": {"core.filename": ".DS_Store"}}`

Filters are combined as AND, e.g. `{"core.extension": ".jpg", "hsm.status": "migrated"}` matches `.jpg` files which are HSM migrated.

Retrieving results

When search results are read, they can be retrieved using the url returned when the query was submitted.

```
$ curl 'http://example.com/api/search/1/' -H "Authorization: Api-Key
$TOKEN"
{
  "count": 1,
  "next": null,
  "previous": null,
  "items": [
    {
      "href": "http://example.com/api/file/?
path=%2Fmmfs1%2Fdata%2Fhello.txt&site=site1",
      "site": "site1",
      "path": "/mmfs1/data",
      "name": "hello.txt",
      "metadata": {
          "core.accesstime": "2021-10-12T16:27:28",
          "core.changetime" : "2021-10-12T16:28:45",
          "core.directory" : "/mmfs1/data",
          "core.extension" : ".txt",
          "core.filename" : "hello.txt",
          "core.group.id" : 0,
          "core.group.name" : "root",
          "core.hash.sha512": "db3974a97...94d2434a593",
          "core.modificationtime" : "2021-10-12T16:28:45",
          "core.pathname": "/mmfs1/data/hello.txt",
          "core.size" : 12,
```

```
          "core.user.id" : 0,
          "core.user.name" : "root",
          "gpfs.filesetname" : "root",
          "gpfs.filesystem" : "mmfs1",
          "gpfs.kballocated" : 0,
          "gpfs.poolname" : "sas1",
          "hsm.status" : "migrated"
          "ngenea.pathname" : "data/hello.txt",
          "ngenea.size" : 12,
          "ngenea.target" : "awss3",
          "ngenea.uuid": "acf1a307-5b6a-43b0-8fb2-d2b366e88008",
      }
    }
  ],
  "metadata_fields": ["core.accesstime", ...],
  "complete": true,
  "errors": {"site2": "Search backend is offline"}
}
```

Results from different sites may not arrive at the same time. The `complete` field indicates whether all sites what returned their results. This includes when a site returns with an error.

Results from different sites are 'concatenated', meaning if the same file exists on multiple sites, there will be separate result items for the file for each site.

The `metadata` field on each item contains arbitrary file metadata. The specific metadata will vary depending on the search backend being used. In the case of the PixStor Search backend, the available fields will vary depending on file type, and which plugins were used when the files were ingested.

If `metadata_fields` was specified when the query was submitted, the `metadata_fields` entry in the response will match, with any wildcards expanded to list the avilable fields which match those wildcards. Otherwise, the `metadata_fields` entry will list all the available metadata fields which could be returned from the search backend. Individual files may not have all the listed fields.

All search backends format results to be namespaced, similar to PixStor Search, for consistency.

If an error occurs while performing the search on any of the sites, the `errors` entries will provide a mapping of site names and error messages.

Parameters

Search results are paginated. The following parameters can be used to control what results are returned

name	description
`page`	Numbered page of results to fetch. Default: 1
`page_size`	

name	description
	Maximum number of results to return per page. Default: 20
`sort`	One or more fields to sort results on, separated by commas, e.g. `?sort=name,site`. Field names can be prefixed with `-` to reverse order. For fields in `metadata`, the field name is specified as is, e.g. `?sort=-core.accesstime`. Default: arbitrary order.

Merged results

When a search is submitted with `"merge": true`, the search results will be 'merged'.

This means that entries for matching files from different sites will be combine. An entry is considered to be matching if it has the same full path.

```
$ curl 'http://example.com/api/search/2/' -H "Authorization: Api-Key
$TOKEN"
{
  "count": 1,
  "next": null,
  "previous": null,
  "items": [
    {
      "path": "/mmfs1/data",
      "name": "hello.txt",
      "metadata": {
          "core.accesstime": "2021-10-12T16:27:28",
          "core.changetime" : "2021-10-12T16:28:45",
          "core.directory" : "/mmfs1/data",
          "core.extension" : ".txt",
          "core.filename" : "hello.txt",
          "core.group.id" : 0,
          "core.group.name" : "root",
          "core.hash.sha512": "db3974a97...94d2434a593",
          "core.modificationtime" : "2021-10-12T16:28:45",
          "core.pathname": "/mmfs1/data/hello.txt",
          "core.size" : 12,
          "core.user.id" : 0,
          "core.user.name" : "root",
          "gpfs.filesetname" : "root",
          "gpfs.filesystem" : "mmfs1",
          "gpfs.kballocated" : 0,
          "gpfs.poolname" : "sas1",
          "hsm.status" : "migrated"
          "ngenea.pathname" : "data/hello.txt",
          "ngenea.size" : 12,
          "ngenea.target" : "awss3",
```

```
            "ngenea.uuid": "acf1a307-5b6a-43b0-8fb2-d2b366e88008",
        },
        "status": {
            "site1": true,
            "site2": false
        }
      }
  ],
  "metadata_fields": ["core.accesstime", ...],
  "complete": true
}
```

Merged results no longer have the `site` and `href` fields. In their place is a `status` field, which maps sites to whether the file is 'resident' on that site.

A file is considered resident if the file is not migrated, or is premigrated ('hydrated'). A file is considered not resident if the file is migrated (stubbed), or not present at all.

Max Results

There is a hard limit on the number of results returned, per site. By default, each site will return, at most, 200 results.

Fetching a lot of results makes queries slower and, since results are stored in the DB, storing more results uses more space. One the other hand, the limiting may lead to some matches not being returned.

The maximum number of results per site is controlled by the `search_max_results` configuration - see Configuration for more info.

Result limiting is applied when the search query is submitted, not when results are retrieved. If you change `search_max_results`, you will need to resubmit your query to fetch any additional matches.

Note, some backends have a hard limit of 10,000 results.

Housekeeping

The results from a query are stored, so they can be retrieved multiple times without performing a new query.

However, over time, the files on each site will change, and the stored results may no longer accurately reflect the active file system.

Therefore, old results are periodically culled. The housekeeping process runs once a day, and removes results for any search which was submitted more than a week ago (by default). A different 'time-to-live' (TTL) can be set using the `search_result_ttl` configuration - see Configuration for more information.

Results can also be manually removed by performing a `DELETE` request against the given search result endpoint

```
curl -X DELETE 'http://example.com/api/search/1/' -H "Authorization:
Api-Key $TOKEN"
```

Controlling Bandwidth Usage

Ngenea Hub can control the amount of traffic speed for each node within each site that processes files through Ngenea. This will limit both outgoing and incoming traffic with defined cloud services.

Enabling the bandwidth feature in the UI

Regardless of enabling this feature, it is possible to change the bandwidth via REST and will enable changing of bandwidth when the `Site` instance has its `bandwidth` attribute changed.

This can be enabled by enabling the feature flag, details on how to do this can be found on the Feature Flags page.

This will enable the UI within the Site details page and represents the bandwidth limit in Mb/s.

Checking Node status

Each site within Ngenea Hub is the collection of all nodes running an instance of Ngenea Worker using the name of a given site. These are nodes within a cluster that are collectively listening to the same queue for a given site. Each time any worker comes online on a new node or a known node, this is tracked within Ngenea Hub using `Node` objects. These are automatically created when first starting up a worker, after creation their online status can be monitored.

You can view the nodes for each site within `/api/nodes/` this will be a complete list of all nodes known to Ngenea Hub.

For bandwidth control, there will need to be existing nodes known to Ngenea Hub, otherwise the bandwidth rules cannot be applied. If your worker coming online was not tracked due to timing issues, you can manually scan for existing nodes using one of the Actions.

Registering Datastores

In order to control the bandwidth for all nodes under a site, the site will need to have the Ngenea policy targets defined as `Datastore` instances within Ngenea Hub.

The following example is for defining an AWS S3 target for Ngenea within Ngenea Hub:

```
{
    "name": "site1_amazon",
    "type": "S3",
```

```
    "bucket": "bucket01",
    "secretaccesskey": "secret-key",
    "accesskey": "access-key"
}
```

With this established datastore, all that is left to do is link the created datastore to a site so that when a change in bandwidth is applied, the site knows what service to limit the traffic to.

Cloud IPAddresses

Each datastore that points to a cloud target will have access to a list of all known IP addresses that are associated with that specific service. This will be used to limit all traffic between those IP ranges and the nodes currently running a worker instance when a bandwidth limit is applied.

These IP ranges are updated internally once a day in a scheduled task.

Using manual IP addresses

Manual IP addresses can be used to override the list of cloud related IPs, this can be useful to control bandwidth to custom endpoint targets such as services like minio making use of POST `api/ipaddresses/`. Using the following example to add an address to the list of IPAddresses:

> **Note:** This will disable the use of all cloud addresses for a datastore and will instead only use the custom IP addresses.

```
{
    "ipaddr": "208.65.153.237",
    "datastore": 1
}
```

This will ensure that all traffic between those nodes and the defined IP addresses will be limited to the max bandwidth limit

Linking a Datastore to a Site

With our example site `site1` creating a `SiteLink` between `site1` and the datastore `site1_amazon` allows the bandwidth to be applied to all S3 targets on all nodes that are running the Ngenea Worker service.

For this example all local data for this Ngenea is located in `/mmfs1/data/aws_data` with data being placing data in the bucket `bucket01` under `aws_data/`.

The following `SiteLink` can be used to represent this:

```
{
    "site": "site1",
    "datastore": "site1_amazon",
```

```
    "site_path": "/mmfs1/data/aws_data/",
    "datastore_path": "aws_data/"
}
```

This will ensure that each `Node` under `site1` will limit its traffic to all related addresses.

Applying the bandwidth

> **Note:** This will effect all traffic on each node running a worker to the cloud services defined with the sites linked datastores.

With the SiteLink in place, changing the bandwidth attribute to the Mb/s desire on the site instance via the API route PATCH `/api/site/{id}`:

```
{
  "bandwidth": 1000
}
```

Using this or the UI, this will cause the hub to signal the worker to limit traffic using the IP ranges defined for the datastores.

This total bandwidth will be divided between all nodes for any given site, so if a site has a bandwidth of 1Gb/s then both nodes will be limited to 500mb/s.

Configuration

Global Configurations

Some configurations are stored in the Ngenea Hub configuration file, as described in Hub Configuration. These are generally static, or sensitive settings. Changes to these settings require a service restart.

In addition, there are configurations which can be changed on-the-fly, typically to change Ngenea Hub behaviour. These settings can be viewed and changed via the REST API, as described below.

Available Settings

Name	Description	Default
search_backend	Backend to use when performing searches. Currently supported backends: `analytics`, `pixstor_search`.	`analytics`
search_result_ttl	How long search results should be stored, in days.	7
search_max_results	Maximum number of search results to fetch from the search backend, per site. Fetching more results will make queries slower and will require	200

Name	Description	Default
	more storage space. Fetching fewer results may lead to some files being missing. Note, some backends have a hard limit of 10,000 results.	
stat_timeout	How long to wait for results from the `/api/file/` endpoint, in seconds.	10

REST API

Configurations can be listed and set via the Ngenea Hub REST API.

> **Note:** The configurations endpoint does not support client key authentication. You must use JWT Authentication.

To list the current configuration settings,

```
$ curl -s 'http://example.com/api/configurations/' -H 'Accept:
application/json' -H 'Authorization: Bearer $JWT_ACCESS_TOKEN'
{
    "search_backend": "analytics",
    "search_max_results": 200,
    "search_result_ttl": 7,
    ...
}
```

To change one or more configuration settings, make a `PATCH` request the same endpoint

```
$ curl -s -X PATCH 'http://example.com/api/configurations/' -H
'Accept: application/json' -H "Authorization: Bearer
$JWT_ACCESS_TOKEN" -H 'Content-Type: application/json' -d
'{"search_max_results": 500}'
{
    "search_max_results": 500,
    ...
}
```

Settings Migration

In versions 1.9.0 and earlier, some of the above settings were configured via the Ngenea Hub config file (Hub Configuration).

Upon updating to version 1.10.0 or above, any values currently set in that config file will be captured. Thereafter, any changes to those settings within the config file will be ignored.

Site-specific Configurations

Some configuration options can be set on a per-site level, and may differ between sites.

These can be viewed and changed via the REST API, as described below. They can also be viewed and changed in the Ngenea Hub UI, from the 'Sites' tab on the Administration page.

Available Settings

Name	Description	Default
bandwidth	Limit the bandwidth for the site (In MB/s). In the UI, it is hidden behind the `bandwidth_controls` feature flag (see Feature Flags)	not set (unlimited)
elasticsearch_url	URL used to interact with Elasticsearch when `search_backend` is set to `analytics` (see **Global Configurations** above). The URL is evalutated on the node(s) on which the site worker is running.	`localhost:9200`
file_batch_gb	Limit the total size of file data in a batch, in gigabytes. See **File Batching** below	1
file_batch_size	Limit the total number of files in a batch. See **File Batching** below	40
lock_threshold	The `snapdiff` discovery uses locking to prevent multiple snapdiff running against the same fileset at once. To prevent stale locks, lock are considered 'expired' after the `lock_threshold`, given in seconds.	86400 (one day)
include	A list of include glob patterns which will apply to all workflows run against this site	not set
exclude	A list of exclude glob patterns which will apply to all workflows run against this site	not set

File Batching

The file list generated by discovery tasks may be broken into smaller batches before passing them to workflow steps.

This makes the overall job execution more granular. Individual tasks will be smaller and faster. This also makes it easier to cancel a job, given that only PENDING tasks can be cancelled.

On the other hand, if the batching is too small, the large number of tasks generated may saturate the job queue, blocking out tasks from other jobs.

File batching is based on both `file_batch_gb` and `file_batch_size`. Whichever limit results in a smaller batch is the one which is used. For example, given 100 files of 500MB each, a `file_batch_gb` of 1 and `file_batch_size` of 10 will result in 50 batches of 2 files each (1GB total per batch), because 1GB (2 files) is smaller than 10 files (5GB).

REST API

Configurations can be listed and set via the Ngenea Hub REST API.

The sites endpoint supports client key authentication

To list the current site configuration settings,

```
$ curl -s 'http://example.com/api/sites/1/' -H 'Accept: application/
json' -H 'Authorization: Api-Key $APIKEY'
{
    "name": "site1",
    "elasticsearch_url": "localhost:19200",
    "file_batch_size": 100,
    ...
}
```

Note - configurations are only included when fetching a specific site, not when listing all sites.

To change one or more configuration settings, make a `PATCH` request the same endpoint

```
$ curl -s -X PATCH 'http://example.com/api/sites/1/' -H 'Accept:
application/json' -H "Authorization: Api-Key $APIKEY" -H 'Content-
Type: application/json' -d '{"file_batch_gb": 5}'
{
    "file_batch_gb": 5,
    ...
}
```

Feature Flags

Feature flags control whether selected pre-release features are enabled.

Certain features may be included in a release which aren't yet fully implemented, or fully tested. By default, these features are disabled and 'hidden', so should not affect normal functionality.

However, these features may be enabled on a 'preview' basis, on the understanding that they may be incomplete or unstable.

Warning: **Do not** enable preview features unless you are willing to accept the potential risks.

Once a feature is finalised and stable, it will be released officially, and the corresponding feature flag will be removed.

Available Features

The following features are currently available.

name	description	stability	default
`searchui`	Enable search features in the Ngenea Hub UI	inprogress	False
`bandwidth_controls`	Enable bandwidth controls in the Ngenea Hub UI	inprogress	False

REST API

Features can be listed, enabled, or disabled via the Ngenea Hub REST API.

To list the available features, and whether they're currently enabled

```
$ curl -s 'http://example.com/api/features/'  -H 'Accept:
application/json'  -H "Authorization: Api-Key $TOKEN"
{
    "count": 1,
    "next": null,
    "results": [
        {
            "name": "searchui",
            "description": "Enable search features in the Ngenea Hub
UI",
            "enabled": false
        },
        {
            "name": "bandwidth_controls",
            "description": "Enable bandwidth controls in the Ngenea
Hub UI",
            "enabled": false,
        }
    ]
}
```

Individual features are keyed by their name, e.g. `http://example.com/api/features/searchui/`

To enable a feature, make a `PATCH` request against the desired feature

```
$ curl -s -X PATCH 'http://example.com/api/features/searchui/'  -H
'Accept: application/json'  -H "Authorization: Api-Key $TOKEN" -H
'Content-Type: application/json' -d '{"enabled": true}'
[
    {
        "name": "searchui",
        "description": "Enable search features in the Ngenea Hub UI",
        "enabled": true
```

```
    },
    ...
]
```

And similarly, to disable a feature

```
curl -s -X PATCH 'http://example.com/api/features/searchui/'  -H
'Accept: application/json'  -H "Authorization: Api-Key $TOKEN" -H
'Content-Type: application/json' -d '{"enabled": false}'
```

Note: It may be necessary to restart the Ngenea Hub service for a feature change to take effect.

ngclient

Alternatively, feature flags can be interacted with using ngclient.

To list available features and whether they're currently enabled

```
$ ngclient features list
 [X]  searchui              Enable search features in the UI
 [ ]  bandwidth_controls    Enable bandwidth controls in the UI
```

To enable a feature

```
ngclient features enable bandwidth_controls
```

And to disable a feature

```
ngclient features disable bandwidth_controls
```

See ngclient features for more information.

Actions

Actions allow administrative users to perform certain operations, as described below.

Available Actions

`discover_nodes`

Ngenea Hub monitors for when new nodes come online.

The `discover_nodes` action can be used to manually scan for nodes. This may be necessary if a node was previously manually removed from Ngenea Hub

This action takes no arguments.

REST API

Actions are submitted via the `/api/actions/` endpoint, and typically execute asynchronously. The state of the action can be viewed via the same endpoint.

> **Note:** The actions endpoint does not support client key authentication. You must use JWT Authentication.

To submit and action, make a `POST` request against the `/api/actions` endpoint

```
$ curl -s -X POST 'http://example.com/api/actions/' -H 'Accept:
application/json' -H "Authorization: Bearer $JWT_ACCESS_TOKEN" -H
'Content-Type: application/json' -d '{"action": "discover_nodes"}'
{
    "id": 12345,
    ...
}
```

This returns a unique id, which can be used to check the status and results of the action

```
$ curl -s 'http://example.com/api/actions/12345/' -H 'Accept:
application/json' -H "Authorization: Bearer $JWT_ACCESS_TOKEN"
{
    "action": "discover_nodes",
    "user": "myuser",
    "state": "SUCCESS"
    "results": {...}
}
```

Limitations

Site Sync

Site Sync must only be used in one direction

The intent of `site_sync` is for one source site to synchronise all required changes to any amount of destination sites and not consider the state of the destination site(s). Site Sync must only be utilised when data is required to be synchronised without concern for any active changes on destination site(s).

Site Sync only considers changes within an applicable time window for synchronisation

Synchronisation is not ‘event driven’. Changes are collated within a window of time (defined by the associated schedule), and sent as a group.

The order in which certain events occurred cannot be determined. For example; delete determination; where no data point exists to determine the ‘change time’ [ctime] of a file or directory due to deletion prior to the sync time window.

During bidirectional synchronisation, when conflicting create/modify and delete events occur for files, the create/modify event takes precedence over the delete event to prevent data loss. In such scenarios, a newer version of the file which was prior deleted will be present after synchronisation. Directory behaviour is not affected.

Site Sync supports Independent Filesets

Site Sync methodology is incompatible with any requirement to synchronise an entire file system, nor is use of Dependent Filesets, or arbitrary directory trees supported.

Destination site Independent Filesets must exist prior to synchronisation

Site Sync does not create Independent Filesets on destination site(s) prior to synchronisation. Destination Independent Fileset creation is an administrative function and must be undertaken prior to configuration and operation of a Site Sync methodology to the destination site.

Directory deletion is not supported

Site sync does not support directory deletion. Deletion of a large file tree structure on a source site will delete files within the directory structure, resulting in an empty directory tree on the destination site.

Bidirectional Site Sync

Bidirectional Site Sync implements eventual consistency

Site Sync adheres to the principle of eventual consistency whereby one or more subsequent Site Sync jobs or tasks are required to be enacted for source and destination sites to be in sync . Prior to all required Site Sync jobs enacting the synchronisation status is viewed as partially synchronised. Each subsequent job increases the totality of synchronisation.

Data which has failed to be synchronised in prior synchronisations is placed into subsequent synchronisation runs, leading to eventual consistency.

Bidirectional Site Sync is only supported via schedules

Ref: Schedules

A bidirectional Site Sync is created via defining a schedule using the `bidirectional_snapdiff` discovery and subscribing the `bidirectional_sync` workflow to the schedule.

A path may only be managed by one schedule per site, and at most one workflow may be subscribed to a `bidirectional_snapdiff` schedule.

Hub does not support multiple bidirectional synchronisation with the same source site for the same Independent Fileset (E.G. between site1 and site2, and between site1 and site3).

The `bidirectional_snapdiff` discovery causes all iterative Independent Fileset changes to be tracked. Identified changes are compared to across iterative runs. When both sets of changes have been evaluated, only appropriately valid changes are synchronised to the destination site.

Bidirectional synchronisation is sequential

When setting up a bidirectional site sync ‘site1’ is the site which is configured when creating the schedule, and ‘site2’ is the site configured as the `destinationsite` when subscribing the workflow to the schedule.

A bidirectional site sync will first synchronise changes from site1 to site2, and then from site2 to site1.

A failure while synchronising site1 to site2 will not block the reverse direction sync from enacting.

The sequential nature of synchronisation ensures conflict situations where the synchronisation from site2 to site1 wins by virtue of the last write [most recent write at any site] of a file taking precedence.

Swapping files / Last write wins

Synchronisation of a set of file moves whereby the actions include synchronisation determination of the paths of two files being swapped is complex and can result in conflicts. Where identical file paths exist at the destination site, file moves will fail rather than overwriting the existing files.

Manual intervention is required before a synchronisation will again succeed. N.B.: the replaying of the swap of the files on the destination site is not sufficient to resolve the conflict.

Renaming or deleting files and directories causes re-sending of data

Where a file is moved on site1 and the same file is deleted on site2 during an active synchronisation, the moved file from site1 will be resent to site2. This behaviour will be observed even if the delete event occurred later chronologically.

Renaming or files and directories on both bidirectional Site Sync sites causes duplication of data

This behaviour is observed when a file or directory is moved on both sites to different locations during an active synchronisation. E.G.:

- `/mmfs1/data/path1` is moved to `/mmfs1/data/path2` on `site1`
- `/mmfs1/data/path1` is moved to `/mmfs1/data/path3` on `site2`

This scenario results in duplicate data at both sites in both `/mmfs1/data/path2` and `/mmfs1/data/path3`.

Creation or deletion of empty directories on site 1 does not synchronise to site 2

Site Sync does not perform deletions of empty directories on a destination site and does not create empty directories on a destination site.

Troubleshooting

This section outlines steps for troubleshooting issues with Ngenea Hub

Service Status

To check the status of Ngenea Hub and its individual services

```
ngeneahubctl status
```

To check the status of Ngenea Worker

```
systemctl status ngenea-worker
```

Service Logs

The full logs for Ngenea Hub can be viewed with

```
journalctl -u ngeneahub
```

To view Ngenea Worker logs

```
journalctl -u ngenea-worker
```

Specific Features

Site Sync

Site sync - both one-way and bidirectional - may fail due to conflicts which cannot be automatically resolved. This page outlines options for intervening to resolve such conflicts.

Manual Resolution

In some cases it is possible to resolve conflicts by manually applying changes.

For example, if a file is moved on site A and deleted on site B, sync will fail because there is no file to move (or delete, depending on the sync direction) on the target site. In this case, maunally deleting the file on site A, or re-sending the file (in its new location) onto site B will resolve the conflict. Thereafter, sync will be able to run without issue.

Re-sync all

Another option is to re-sync everything from scratch. This is the safest option, as it ensures that no file changes are lost.

Note that sync will skip any files which are already in the correct state, so a re-sync won't take as long as syncing to a brand new target.

Snapdiff-based sync uses filesystem snapshots to track file changes over time. The snapdiff discovery uses a 'last snap' file to record the last snapshot which was successfully synced.

This last snap file is located at `/mmfs1/.rotate/ngenea-worker.lastsnap.<fileset_name>`

By removing the last snap file, the next sync run will behave as if it has not been run before, and so will sync everything. Once sync has run successfully, you can safely delete the snapshot which was previously recorded in the last snap file (before that file was deleted).

Force rotate

The riskiest option is to force a snapshot rotation. This effectively says that you don't care about the current failure and just want the sync to move on. Note that this method may result in some files changes not being synced. For a safer option, see **Re-sync all** above.

To force a rotate, you should first temporarily disable sync. This can be done by setting the sync schedule to disabled in the Ngenea Hub UI.

Next, create a new snapshot of the fileset. The name is expected to be of the form `ngenea-worker.snapdiff.<timestamp>`.

Update the 'last snap' file (described above), replacing the currently recorded snapshot name with the name of the new snapshot you just created.

This last snap file is located at `/mmfs1/.rotate/ngenea-worker.lastsnap.<fileset_name>`

Finally, re-enable sync. Sync will now pick up new file changes starting from the point at which the new snapshot was created.

Once sync has run successfully, you can safely delete the snapshot which was previously recorded in the last snap file (before that file was updated).

Locking Errors

Lock files are used to ensure that a sync for a given fileset will not run if one is already running.

In this case, any new sync job will fail, and under the snapdiff task details, you will see an error like

```
SnapdiffLockError: Could not perform snapdiff as one is currently
running for provided fileset
```

Under rare circumstances, a lock may not be correctly cleaned up, preventing syncs from running, even though there are none currently active.

In that case, the lock file can be removed manually. First, ensure there there aren't any syncs running. For extra safety, temporarily disable any scheduled syncs.

The lock file is located at `/mmfs1/.rotate/snapdiff-<fileset_name>.lock`

Any snapdiff lock file will automatically expire after 24 hours by default. The lifetime can be changed using the `lock_threshold` site setting

Reference

Default Workflows

This section documents all the workflows which come installed in Ngenea Hub by default. See Custom Workflows for guidance on how to create your own workflows.

`migrate`

Migrate one or more files from a site.

discovery: `recursive`

steps:

- dynamo.tasks.migrate

fields:

- `lock_level` (choice): Filesystem locking level for ngenea operations. One of: `partial`, `implicit`. Default=`partial`

premigrate

Premigrate one or more files from a site.

discovery: `recursive`

steps:

- dynamo.tasks.migrate
  - `premigrate`: True

fields:

- `lock_level` (choice): Filesystem locking level for ngenea operations. One of: `partial`, `implicit`. Default=`inplicit`

recall

Recall one or more files on to a site.

discovery: `recursive`

steps:

- dynamo.tasks.recall

fields:

- `lock_level` (choice): Filesystem locking level for ngenea operations. One of: `partial`, `implicit`. Default=`inplicit`

send

Send files from one site to another via cloud storage.

discovery: `recursive`

steps:

- dynamo.tasks.remove_location_xattrs_for_moved
- dynamo.tasks.migrate
  - `premigrate`: True
- dynamo.tasks.reverse_stub

fields:

- `destinationsite` (string): site to send files to
- `hydrate` (bool): hydrate files on the destination site

site_sync

Sync a fileset from one site to another via cloud storage.

The `snapdiff` discovery looks for changes within the fileset on the source site since the last time the workflow was invoked. These changes are then synced to the destination site.

The workflow should be invoked with a single path which is the link point of the independent fileset to be synced.

discovery: `snapdiff`

fields:

- `destinationsite` (string): site to sync changes to
- `sync_preference` (string): determines how conflicts should be resolved on the remote site. One of: newest, local, ignore

created

Files with state `created` are sent to the destination site, subject to `sync_preference`

steps:

- dynamo.tasks.remove_location_xattrs_for_moved
- dynamo.tasks.migrate
  - `premigrate`: True
  - `overwrite`: True
- dynamo.tasks.reverse_stub
  - `overwrite`: True

updated

Files with state `updated` are sent to the destination site, subject to `sync_preference`

steps:

- dynamo.tasks.check_sync_state
- dynamo.tasks.remove_location_xattrs_for_moved
- dynamo.tasks.migrate
  - `premigrate`: True
  - `overwrite`: True
- dynamo.tasks.reverse_stub
  - `overwrite`: True

moved

Files with state `moved` are moved 'in-place' on the destination site

steps:

- dynamo.tasks.move_paths_on_gpfs

deleted

Files with state `deleted` are removed on the destination site

steps:

- dynamo.tasks.delete_paths_from_gpfs

Workflow Steps

This section documents all the currently supported steps in Ngenea Hub. See Custom Workflows for guidance on how to use these steps in your own workflows.

dynamo.tasks.migrate

Migrates a list of files to a pre-defined remote target using Ngenea.

Argument	Type	Default	Description
`premigrate`	`bool`	`False`	retain the content of every migrated file and do not set the OFFLINE flag for the file.migrating.
`stub_size`	`int`	`0`	retain a segment of every migrated file starting from its beginning and having a specified approximate length in bytes.
`overwrite`	`bool`	`False`	overwrite remote objects if they already exist--do not create remote object instances with various UUID suffixes
`lock_level`	`string`	`implicit`	Defined the locking mode that ngenea will use when performing the migrate
`endpoint`	`string`		specify the endpoint to migrate

dynamo.tasks.recall

Recalls a list of files to a pre-defined remote target using Ngenea.

Argument	Type	Default	Description
`skip_hash`	`bool`	`False`	If the recall should skip checking the hash of the file
`endpoint`	`string`		specify which endpoint(site) to recall from
`lock_level`	`string`	`partial`	Defines the locking level ngenea will use during the recall
`default_uid`	`string`		When a file is recalled, it uses this UID if one is not set on the remote object
`default_gid`	`string`		

Argument	Type	Default	Description
			When a file is recalled, it uses this GID if one is not set on the remote object

dynamo.tasks.reverse_stub

Recalls a list of files to a pre-defined remote target using Ngenea.

Argument	Type	Default	Description
`hydrate`	`bool`	`False`	If the file should be premigrated instead of a regular stub
`stub_size`	`int`	`0`	The max file size before files will be stubbed for this task
`skip_hash`	`bool`	`False`	If the recall should skip checking the hash of the file
`overwrite`	`bool`	`False`	overwrite local files if they already exist.
`endpoint`	`string`		specify which endpoint(site) to recall from.
`retry_stale`	`string`	None	Controls if the worker should attempt to retry file failures due to stale file handles. This string can be either stub for only removing reverse stubbed files or all.
`lock_level`	`string`	`implicit`	Defines the locking level ngenea will use during the recall
`default_uid`	`string`		When a file is recalled, it uses this UID if one is not set on the remote object
`default_gid`	`string`		When a file is recalled, it uses this GID if one is not set on the remote object

dynamo.tasks.delete_paths_from_gpfs

Removes a list of files from a GPFS filesystem.

Argument	Type	Default	Description
`recursive`	`bool`	`False`	If any directory path is provided and this is set, it will remove the entire file tree, otherwise it will only remove empty directories

dynamo.tasks.check_sync_state

Checks a provided site against the calling sites to ensure that the local file is in a specified state compared to another site. Using this task will also perform `dynamo.tasks.stat_paths` on the provided site before execution.

Argument	Type	Default	Description
`sync_preference`	`string`	`None`	Dictates what state the local file should be to pass the check. Options are "newest"

Argument	Type	Default	Description
			which passes if the local file is the latest version of the file on either site, "local" which accepts the local file version regardless of the check and "ignore" which always uses the other sites file version.

dynamo.tasks.move_paths_on_gpfs

Moves a file on the filesystem using provided paths with a `source` key.

Argument	Type	Default	Description
`delete_remote_xattrs`	`bool`	`False`	If set, after a file has been moved all remote location xattrs will be removed

dynamo.tasks.remove_location_xattrs_for_moved

This task removes all remote location xattrs on all provided paths.

This step takes no additional arguments.

dynamo.tasks.move_in_cloud

Moves a file on the filesystem's related cloud storage platform using provided paths with a `source` key.

This step takes no additional arguments.

dynamo.tasks.remove_from_cloud

Deletes a file on the filesystem's related cloud storage platform using provided paths.

This step takes no additional arguments.

dynamo.tasks.ensure_cloud_file_exists

Ensures all files provided to the task exist on the filesystem's related cloud storage platform. If some do not, it will attempt to retry this check an additional two more times before failing.

This step takes no additional arguments.

Job States

When jobs are created on the call of a workflow, they can end up in specific state that

State	Description
`Pending`	The job is being populated with tasks through its discovery task and will begin when paths have been collected
`Started`	Some tasks within the job have started to be processed
`Success`	All tasks in a job have completed successfully
`Failure`	There was an error or failure when attempting a task within a job, meaning the job could not complete
`Skipped`	Based on the output of the provided discovery task, no tasks needed to be created so there is no work to
`Cancelled`	The job has been manually closed via request and all remaining task have been cancelled

Task States

When jobs create tasks, after performing their action on the provided files they can end up in specific state as seen below

State	Description
`Pending`	This task is has been created but has not yet been picked up by a site
`Started`	This task is now running on site
`Success`	This task has completed successfully
`Failure`	There was a unexpected error when attempting a task within a job, meaning the job could not complete and could not provide structured output
`Error`	There was a captured error when attempting a task within a job, meaning the job will not have processed all paths but has structured output of what has been completed. Any paths which were successfully processed will be handled by subsequent tasks in a task chain.
`Skipped`	This task has will have no work to perform so it has been automatically skipped by another task
`Cancelled`	Either a previous task has failed or the job has been manually cancelled, causing this task to no longer run

Note: Job and task states are updated asynchronously. There may be, for example, a short delay between a job/task completing and its state being reported as such.

API

`GET /actions/`

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination

and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].action** (string) -- (required)
- **results[].id** (integer) -- (read only)
- **results[].state** (string) --

`POST /actions/`

Request JSON Object:

- **action** (string) -- (required)

Status Codes:

- 201 Created --

Response JSON Object:

- **action** (string) -- (required)

`GET /actions/{id}/`

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **action** (string) -- (required)
- **id** (integer) -- (read only)
- **results** (string) -- (read only)
- **state** (string) --
- **user** (integer) --

`GET /auth/clientkeys/`

API endpoint for managing client keys

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --

- **results[].id** (integer) -- (read only)
- **results[].name** (string) -- Name of the client key (required)
- **results[].url** (string) -- (read only)

`POST /auth/clientkeys/`

API endpoint for managing client keys

Request JSON Object:

- **api_key** (string) -- (read only)
- **id** (integer) -- (read only)
- **name** (string) -- Name of the client key (required)
- **url** (string) -- (read only)

Status Codes:

- 201 Created --

Response JSON Object:

- **api_key** (string) -- (read only)
- **id** (integer) -- (read only)
- **name** (string) -- Name of the client key (required)
- **url** (string) -- (read only)

`GET /auth/clientkeys/{id}/`

API endpoint for managing client keys

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **id** (integer) -- (read only)
- **name** (string) -- Name of the client key (required)
- **url** (string) -- (read only)

`PATCH /auth/clientkeys/{id}/`

API endpoint for managing client keys

Parameters:

- **id** (string) --

Request JSON Object:

- **id** (integer) -- (read only)
- **name** (string) -- Name of the client key (required)
- **url** (string) -- (read only)

Status Codes:

- 200 OK --

Response JSON Object:

- **id** (integer) -- (read only)
- **name** (string) -- Name of the client key (required)
- **url** (string) -- (read only)

`DELETE /auth/clientkeys/{id}/`

API endpoint for managing client keys

Parameters:
- **id** (string) --

Status Codes:
- 204 No Content --

POST /auth/token/

Request JSON Object:
- **password** (string) -- (required)
- **username** (string) -- (required)

Status Codes:
- 201 Created --

Response JSON Object:
- **password** (string) -- (required)
- **username** (string) -- (required)

GET /auth/token/publickey/

API endpoint for retrieving public key that is used for token verification.

Status Codes:
- 200 OK --

POST /auth/token/refresh/

Takes a refresh type JSON web token and returns an access type JSON web token if the refresh token is valid.

Request JSON Object:
- **access** (string) -- (read only)
- **refresh** (string) -- (required)

Status Codes:
- 201 Created --

Response JSON Object:
- **access** (string) -- (read only)
- **refresh** (string) -- (required)

POST /auth/token/verify/

Verifies that the token is not expired AND the token owner exists in the database AND the token owner is an active user.

Request JSON Object:
- **token** (string) -- (required)
- **type** (string) -- Token type e.g: access or refresh (required)

Status Codes:
- 201 Created --

Response JSON Object:
- **token** (string) -- (required)
- **type** (string) -- Token type e.g: access or refresh (required)

GET /configurations/

API endpoint for viewing and setting configurations.

Status Codes:

- 200 OK --

Response JSON Object:

- **search_backend** (string) -- Search backend
- **search_max_results** (integer) -- Maximum search results
- **search_result_ttl** (integer) -- Maximum time to store search results in days
- **stat_timeout** (integer) -- Maximum time to wait for results from stat in seconds

`PATCH /configurations/`

API endpoint for viewing and setting configurations.

Request JSON Object:

- **search_backend** (string) -- Search backend
- **search_max_results** (integer) -- Maximum search results
- **search_result_ttl** (integer) -- Maximum time to store search results in days
- **stat_timeout** (integer) -- Maximum time to wait for results from stat in seconds

Status Codes:

- 200 OK --

Response JSON Object:

- **search_backend** (string) -- Search backend
- **search_max_results** (integer) -- Maximum search results
- **search_result_ttl** (integer) -- Maximum time to store search results in days
- **stat_timeout** (integer) -- Maximum time to wait for results from stat in seconds

`GET /datastores/`

API endpoint for managing DataStores.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].accesskey** (string) --
- **results[].accesskeyid** (string) --
- **results[].bucket** (string) --
- **results[].container** (string) --
- **results[].credentialsfile** (string) --
- **results[].endpoint** (string) --

- **results[].id** (integer) -- (read only)
- **results[].name** (string) -- DataStore Name (required)
- **results[].region** (string) --
- **results[].secretaccesskey** (string) --
- **results[].storageaccount** (string) --
- **results[].type** (string) -- Site Type (required)
- **results[].url** (string) -- (read only)

`POST /datastores/`

API endpoint for managing DataStores.

Request JSON Object:

- **accesskey** (string) --
- **accesskeyid** (string) --
- **bucket** (string) --
- **container** (string) --
- **credentialsfile** (string) --
- **endpoint** (string) --
- **id** (integer) -- (read only)
- **name** (string) -- DataStore Name (required)
- **region** (string) --
- **secretaccesskey** (string) --
- **storageaccount** (string) --
- **type** (string) -- Site Type (required)
- **url** (string) -- (read only)

Status Codes:

- 201 Created --

Response JSON Object:

- **accesskey** (string) --
- **accesskeyid** (string) --
- **bucket** (string) --
- **container** (string) --
- **credentialsfile** (string) --
- **endpoint** (string) --
- **id** (integer) -- (read only)
- **name** (string) -- DataStore Name (required)
- **region** (string) --
- **secretaccesskey** (string) --
- **storageaccount** (string) --
- **type** (string) -- Site Type (required)
- **url** (string) -- (read only)

`GET /datastores/{id}/`

API endpoint for managing DataStores.

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **accesskey** (string) --
- **accesskeyid** (string) --
- **bucket** (string) --
- **container** (string) --
- **credentialsfile** (string) --
- **endpoint** (string) --
- **id** (integer) -- (read only)
- **name** (string) -- DataStore Name (required)
- **region** (string) --
- **secretaccesskey** (string) --
- **storageaccount** (string) --
- **type** (string) -- Site Type (required)
- **url** (string) -- (read only)

`PATCH /datastores/{id}/`

API endpoint for managing DataStores.

Parameters:

- **id** (string) --

Request JSON Object:

- **accesskey** (string) --
- **accesskeyid** (string) --
- **bucket** (string) --
- **container** (string) --
- **credentialsfile** (string) --
- **endpoint** (string) --
- **id** (integer) -- (read only)
- **name** (string) -- DataStore Name (required)
- **region** (string) --
- **secretaccesskey** (string) --
- **storageaccount** (string) --
- **type** (string) -- Site Type (required)
- **url** (string) -- (read only)

Status Codes:

- 200 OK --

Response JSON Object:

- **accesskey** (string) --
- **accesskeyid** (string) --
- **bucket** (string) --
- **container** (string) --
- **credentialsfile** (string) --
- **endpoint** (string) --
- **id** (integer) -- (read only)
- **name** (string) -- DataStore Name (required)
- **region** (string) --
- **secretaccesskey** (string) --
- **storageaccount** (string) --
- **type** (string) -- Site Type (required)
- **url** (string) -- (read only)

DELETE /datastores/{id}/

API endpoint for managing DataStores.

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

GET /features/

API endpoint for managing feature flags.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].description** (string) -- Description of what the feature does (read only)
- **results[].enabled** (boolean) -- Whether the feature has been enabled
- **results[].name** (string) -- Name of the feature (read only)

GET /features/{name}/

API endpoint for managing feature flags.

Parameters:

- **name** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **description** (string) -- Description of what the feature does (read only)
- **enabled** (boolean) -- Whether the feature has been enabled
- **name** (string) -- Name of the feature (read only)

PATCH /features/{name}/

API endpoint for managing feature flags.

Parameters:

- **name** (string) --

Request JSON Object:

- **description** (string) -- Description of what the feature does (read only)
- **enabled** (boolean) -- Whether the feature has been enabled
- **name** (string) -- Name of the feature (read only)

Status Codes:

- 200 OK --

Response JSON Object:

- **description** (string) -- Description of what the feature does (read only)
- **enabled** (boolean) -- Whether the feature has been enabled
- **name** (string) -- Name of the feature (read only)

`GET /file/`

Retrieves list of files under given path for given site.

Query Parameters:

- **path** (string) -- Target directory path
- **site** (string) -- Site name
- **details** (boolean) -- Show details of children objects

Status Codes:

- 200 OK --

`GET /file/test/`

API endpoint for managing files.

Status Codes:

- 200 OK --

`POST /file/workflow/`

Performs a workflow on a list of files

Request JSON Object:

- **discovery** (string) -- Discovery name
- **exclude[]** (string) --
- **fields** (object) --
- **ignore_site_excludes** (boolean) --
- **ignore_site_includes** (boolean) --
- **include[]** (string) --
- **job** (integer) -- Job ID
- **paths[]** (object) --
- **site** (string) -- Site name (required)
- **workflow** (string) -- Workflow name (required)

Status Codes:

- 201 Created --

Response JSON Object:

- **discovery** (string) -- Discovery name
- **exclude[]** (string) --
- **fields** (object) --
- **ignore_site_excludes** (boolean) --
- **ignore_site_includes** (boolean) --
- **include[]** (string) --
- **job** (integer) -- Job ID
- **paths[]** (object) --
- **site** (string) -- Site name (required)
- **workflow** (string) -- Workflow name (required)

GET /filesets/

Retrieve list of filesets on a given site.

Query Parameters:

- **site** (string) -- Site name

Status Codes:

- 200 OK --

GET /filestatustypes/

API endpoint for managing file status types.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].background_color** (string) -- (required)
- **results[].key** (string) -- (required)
- **results[].label** (string) -- (required)
- **results[].text_color** (string) -- (required)
- **results[].url** (string) -- (read only)

GET /filestatustypes/{key}/

API endpoint for managing file status types.

Parameters:

- **key** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **background_color** (string) -- (required)
- **key** (string) -- (required)
- **label** (string) -- (required)
- **text_color** (string) -- (required)
- **url** (string) -- (read only)

PATCH /filestatustypes/{key}/

API endpoint for managing file status types.

Parameters:

- **key** (string) --

Request JSON Object:

- **background_color** (string) -- (required)
- **label** (string) -- (required)
- **text_color** (string) -- (required)

Status Codes:

- 200 OK --

Response JSON Object:

- **background_color** (string) -- (required)
- **label** (string) -- (required)
- **text_color** (string) -- (required)

`GET /groups/`

API endpoint for managing groups.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].name** (string) -- (required)
- **results[].permissions[]** (string) --
- **results[].users** (string) -- (required)

`POST /groups/`

API endpoint for managing groups.

Request JSON Object:

- **name** (string) -- (required)
- **permissions[]** (string) --
- **users** (string) -- (required)

Status Codes:

- 201 Created --

Response JSON Object:

- **name** (string) -- (required)
- **permissions[]** (string) --
- **users** (string) -- (required)

`GET /groups/{id}/`

API endpoint for managing groups.

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **name** (string) -- (required)
- **permissions[]** (string) --
- **users** (string) -- (required)

`PATCH /groups/{id}/`

API endpoint for managing groups.

Parameters:

- **id** (string) --

Request JSON Object:

- **name** (string) -- (required)
- **permissions[]** (string) --
- **users** (string) -- (required)

Status Codes:

- 200 OK --

Response JSON Object:

- **name** (string) -- (required)
- **permissions[]** (string) --
- **users** (string) -- (required)

`DELETE /groups/{id}/`

API endpoint for managing groups.

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

`GET /health/`

Status Codes:

- 200 OK --

`GET /ipaddresses/`

API endpoint for managing IP addresses.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.
- **datastoreId** (integer) -- Datastore id that the IP is assigned to

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)

- **next** (string) --
- **previous** (string) --
- **results[].datastore.id** (integer) -- (read only)
- **results[].datastore.name** (string) -- DataStore Name (required)
- **results[].datastore.url** (string) -- (read only)
- **results[].id** (integer) -- (read only)
- **results[].ipaddr** (string) -- IP Address (IPv4) (required)
- **results[].url** (string) -- (read only)

`POST /ipaddresses/`

API endpoint for managing IP addresses.

Request JSON Object:

- **datastore** (integer) -- (required)
- **id** (integer) -- (read only)
- **ipaddr** (string) -- (required)
- **url** (string) -- (read only)

Status Codes:

- 201 Created --

Response JSON Object:

- **datastore** (integer) -- (required)
- **id** (integer) -- (read only)
- **ipaddr** (string) -- (required)
- **url** (string) -- (read only)

`GET /ipaddresses/{id}/`

API endpoint for managing IP addresses.

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **datastore.id** (integer) -- (read only)
- **datastore.name** (string) -- DataStore Name (required)
- **datastore.url** (string) -- (read only)
- **id** (integer) -- (read only)
- **ipaddr** (string) -- IP Address (IPv4) (required)
- **url** (string) -- (read only)

`PATCH /ipaddresses/{id}/`

API endpoint for managing IP addresses.

Parameters:

- **id** (string) --

Request JSON Object:

- **datastore** (integer) -- (required)
- **id** (integer) -- (read only)
- **ipaddr** (string) -- (required)
- **url** (string) -- (read only)

Status Codes:

- 200 OK --

Response JSON Object:

- **datastore** (integer) -- (required)
- **id** (integer) -- (read only)
- **ipaddr** (string) -- (required)
- **url** (string) -- (read only)

`DELETE /ipaddresses/{id}/`

API endpoint for managing IP addresses.

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

`GET /jobs/`

API endpoint for managing jobs.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.
- **created** (string) -- Time period string for filtering jobs by time. Leave `null` for displaying jobs in all times.
- **created_time_from** (string) -- Start time for filtering jobs by creation time in UTC. Discarded when `created` parameter is given.
- **created_time_to** (string) -- End time for filtering jobs by creation time in UTC. Discarded when `created` parameter is given.
- **completed_time_from** (string) -- Start time for filtering jobs by completion time in UTC.
- **completed_time_to** (string) -- End time for filtering jobs by completion time in UTC.
- **jobtype** (string) -- Job type
- **state** (array) --
- **owner** (string) -- Job owner's username
- **clientkey** (string) -- Job clientkey information if the owner is an automated process
- **schedule** (string) -- Job schedule information if the job is scheduled
- **clientkeyId** (integer) -- Job clientkey ID
- **scheduleId** (integer) -- Job schedule ID
- **siteId** (integer) -- Job site ID
- **pathPrefix** (string) -- Path prefix for the job paths
- **show_noop** (boolean) -- Show jobs with no processed files

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].clientkey** (string) -- (read only)
- **results[].completed** (string) -- Time of the job completion
- **results[].created** (string) -- Time of the job creation
- **results[].dir_walk_complete** (string) -- (read only)
- **results[].fields** (object) --
- **results[].id** (integer) -- (read only)
- **results[].jobtype** (string) -- (required)
- **results[].num_dir_tasks** (string) -- (read only)
- **results[].numfailedfiles** (integer) --
- **results[].numfiles** (integer) --
- **results[].numprocessedfiles** (integer) --
- **results[].numskippedfiles** (integer) --
- **results[].owner** (string) -- (read only)
- **results[].runtime** (number) --
- **results[].schedule** (string) -- (read only)
- **results[].site** (string) -- Site Name (required)
- **results[].started** (string) -- Time the job started executing
- **results[].state** (string) --
- **results[].url** (string) -- (read only)

`POST /jobs/`

API endpoint for managing jobs.

Request JSON Object:

- **discovery** (string) -- Path discovery method
- **jobtype** (string) -- (required)
- **paths[]** (string) --
- **site** (string) -- Site Name (required)
- **state** (string) --

Status Codes:

- 201 Created --

Response JSON Object:

- **discovery** (string) -- Path discovery method
- **jobtype** (string) -- (required)
- **paths[]** (string) --
- **site** (string) -- Site Name (required)
- **state** (string) --

`GET /jobs/recent/`

Retrieves last N jobs as recent jobs. N = 5 by default (defined in dynamohub/settings/base.py).

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.

- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].clientkey** (string) -- (read only)
- **results[].completed** (string) -- Time of the job completion
- **results[].created** (string) -- Time of the job creation
- **results[].dir_walk_complete** (string) -- (read only)
- **results[].discovery** (string) -- Path discovery method
- **results[].fields** (object) --
- **results[].id** (integer) -- (read only)
- **results[].jobtype** (string) -- (required)
- **results[].num_dir_tasks** (string) -- (read only)
- **results[].numfailedfiles** (integer) --
- **results[].numfiles** (integer) --
- **results[].numprocessedfiles** (integer) --
- **results[].numskippedfiles** (integer) --
- **results[].owner** (string) -- (read only)
- **results[].paths** (string) -- (read only)
- **results[].runtime** (number) --
- **results[].schedule** (string) -- (read only)
- **results[].site** (string) -- Site Name (required)
- **results[].started** (string) -- Time the job started executing
- **results[].state** (string) --
- **results[].url** (string) -- (read only)

`GET /jobs/stats/`

API endpoint for managing jobs.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.
- **created** (string) -- Time period string for filtering jobs by time. Leave `null` for displaying jobs in all times.
- **created_time_from** (string) -- Start time for filtering jobs by creation time in UTC. Discarded when `created` parameter is given.
- **created_time_to** (string) -- End time for filtering jobs by creation time in UTC. Discarded when `created` parameter is given.

- **completed_time_from** (string) -- Start time for filtering jobs by completion time in UTC.
- **completed_time_to** (string) -- End time for filtering jobs by completion time in UTC.
- **jobtype** (string) -- Job type
- **state** (array) --
- **owner** (string) -- Job owner's username
- **clientkey** (string) -- Job clientkey information if the owner is an automated process
- **schedule** (string) -- Job schedule information if the job is scheduled
- **clientkeyId** (integer) -- Job clientkey ID
- **scheduleId** (integer) -- Job schedule ID
- **siteId** (integer) -- Job site ID
- **pathPrefix** (string) -- Path prefix for the job paths
- **show_noop** (boolean) -- Show jobs with no processed files

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].clientkey** (string) -- (read only)
- **results[].completed** (string) -- Time of the job completion
- **results[].created** (string) -- Time of the job creation
- **results[].dir_walk_complete** (string) -- (read only)
- **results[].discovery** (string) -- Path discovery method
- **results[].fields** (object) --
- **results[].id** (integer) -- (read only)
- **results[].jobtype** (string) -- (required)
- **results[].num_dir_tasks** (string) -- (read only)
- **results[].numfailedfiles** (integer) --
- **results[].numfiles** (integer) --
- **results[].numprocessedfiles** (integer) --
- **results[].numskippedfiles** (integer) --
- **results[].owner** (string) -- (read only)
- **results[].paths** (string) -- (read only)
- **results[].runtime** (number) --
- **results[].schedule** (string) -- (read only)
- **results[].site** (string) -- Site Name (required)
- **results[].started** (string) -- Time the job started executing
- **results[].state** (string) --
- **results[].url** (string) -- (read only)

`GET /jobs/{id}/`

API endpoint for managing jobs.

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **clientkey** (string) -- (read only)
- **completed** (string) -- Time of the job completion
- **created** (string) -- Time of the job creation
- **dir_walk_complete** (string) -- (read only)
- **discovery** (string) -- Path discovery method
- **fields** (object) --
- **id** (integer) -- (read only)
- **jobtype** (string) -- (required)
- **num_dir_tasks** (string) -- (read only)
- **numfailedfiles** (integer) --
- **numfiles** (integer) --
- **numprocessedfiles** (integer) --
- **numskippedfiles** (integer) --
- **owner** (string) -- (read only)
- **paths** (string) -- (read only)
- **runtime** (number) --
- **schedule** (string) -- (read only)
- **site** (string) -- Site Name (required)
- **started** (string) -- Time the job started executing
- **state** (string) --
- **url** (string) -- (read only)

`PATCH /jobs/{id}/`

API endpoint for managing jobs.

Parameters:

- **id** (string) --

Request JSON Object:

- **discovery** (string) -- Path discovery method
- **jobtype** (string) -- (required)
- **paths[]** (string) --
- **site** (string) -- Site Name (required)
- **state** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **discovery** (string) -- Path discovery method
- **jobtype** (string) -- (required)
- **paths[]** (string) --
- **site** (string) -- Site Name (required)
- **state** (string) --

`DELETE /jobs/{id}/`

API endpoint for managing jobs.

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

POST /jobs/{id}/cancel/

Cancels the pending and started tasks currently on the MQ for the given ID's job.

Parameters:

- **id** (string) --

Status Codes:

- 201 Created --

Response JSON Object:

- **clientkey** (string) -- (read only)
- **completed** (string) -- Time of the job completion
- **created** (string) -- Time of the job creation
- **dir_walk_complete** (string) -- (read only)
- **discovery** (string) -- Path discovery method
- **fields** (object) --
- **id** (integer) -- (read only)
- **jobtype** (string) -- (required)
- **num_dir_tasks** (string) -- (read only)
- **numfailedfiles** (integer) --
- **numfiles** (integer) --
- **numprocessedfiles** (integer) --
- **numskippedfiles** (integer) --
- **owner** (string) -- (read only)
- **paths** (string) -- (read only)
- **runtime** (number) --
- **schedule** (string) -- (read only)
- **site** (string) -- Site Name (required)
- **started** (string) -- Time the job started executing
- **state** (string) --
- **url** (string) -- (read only)

GET /jobs/{id}/files/

Retrieves the files related with a job, with their execution status.

Parameters:

- **id** (string) --

Query Parameters:

- **category** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **clientkey** (string) -- (read only)
- **completed** (string) -- Time of the job completion
- **created** (string) -- Time of the job creation
- **dir_walk_complete** (string) -- (read only)
- **discovery** (string) -- Path discovery method
- **fields** (object) --
- **id** (integer) -- (read only)
- **jobtype** (string) -- (required)
- **num_dir_tasks** (string) -- (read only)

- **numfailedfiles** (integer) --
- **numfiles** (integer) --
- **numprocessedfiles** (integer) --
- **numskippedfiles** (integer) --
- **owner** (string) -- (read only)
- **paths** (string) -- (read only)
- **runtime** (number) --
- **schedule** (string) -- (read only)
- **site** (string) -- Site Name (required)
- **started** (string) -- Time the job started executing
- **state** (string) --
- **url** (string) -- (read only)

`POST /jobs/{id}/resubmit/`

Resubmits the job with given id. If the job is not finished yet, this action will not have an effect.

Parameters:

- **id** (string) --

Status Codes:

- 201 Created --

`GET /nodes/`

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].id** (integer) -- (read only)
- **results[].name** (string) -- Hostname for a given worker node (required)
- **results[].online** (boolean) -- Whether the node is online
- **results[].site** (string) -- (required)
- **results[].url** (string) -- (read only)

`GET /nodes/{id}/`

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **id** (integer) -- (read only)

- **name** (string) -- Hostname for a given worker node (required)
- **online** (boolean) -- Whether the node is online
- **site** (string) -- (required)
- **url** (string) -- (read only)

PATCH /nodes/{id}/

Parameters:

- **id** (string) --

Request JSON Object:

- **id** (integer) -- (read only)
- **name** (string) -- Hostname for a given worker node (required)
- **online** (boolean) -- Whether the node is online
- **site** (string) -- (required)
- **url** (string) -- (read only)

Status Codes:

- 200 OK --

Response JSON Object:

- **id** (integer) -- (read only)
- **name** (string) -- Hostname for a given worker node (required)
- **online** (boolean) -- Whether the node is online
- **site** (string) -- (required)
- **url** (string) -- (read only)

DELETE /nodes/{id}/

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

GET /schedules/

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].day_of_month** (string) -- The day setting for the cron schedule
- **results[].day_of_week** (string) -- The week setting for the cron schedule
- **results[].discovery** (string) --
- **results[].enabled** (boolean) -- If the schedule should be enabled
- **results[].hour** (string) -- The hour setting for the cron schedule
- **results[].id** (integer) -- (read only)

- **results[].managed_paths** (object) -- Path of managed filesystem elements
- **results[].minute** (string) -- The minute setting for the cron schedule
- **results[].month_of_year** (string) -- The month setting for the cron schedule
- **results[].name** (string) -- Schedule Name (required)
- **results[].site** (string) -- Related site to the schedule (required)
- **results[].url** (string) -- (read only)

POST /schedules/

Request JSON Object:

- **day_of_month** (string) -- The day setting for the cron schedule
- **day_of_week** (string) -- The week setting for the cron schedule
- **discovery** (string) --
- **enabled** (boolean) -- If the schedule should be enabled
- **hour** (string) -- The hour setting for the cron schedule
- **id** (integer) -- (read only)
- **managed_paths** (object) -- Path of managed filesystem elements
- **minute** (string) -- The minute setting for the cron schedule
- **month_of_year** (string) -- The month setting for the cron schedule
- **name** (string) -- Schedule Name (required)
- **site** (string) -- Related site to the schedule (required)
- **url** (string) -- (read only)

Status Codes:

- 201 Created --

Response JSON Object:

- **day_of_month** (string) -- The day setting for the cron schedule
- **day_of_week** (string) -- The week setting for the cron schedule
- **discovery** (string) --
- **enabled** (boolean) -- If the schedule should be enabled
- **hour** (string) -- The hour setting for the cron schedule
- **id** (integer) -- (read only)
- **managed_paths** (object) -- Path of managed filesystem elements
- **minute** (string) -- The minute setting for the cron schedule
- **month_of_year** (string) -- The month setting for the cron schedule
- **name** (string) -- Schedule Name (required)
- **site** (string) -- Related site to the schedule (required)
- **url** (string) -- (read only)

GET /schedules/{id}/

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **day_of_month** (string) -- The day setting for the cron schedule
- **day_of_week** (string) -- The week setting for the cron schedule
- **discovery** (string) --
- **enabled** (boolean) -- If the schedule should be enabled
- **hour** (string) -- The hour setting for the cron schedule
- **id** (integer) -- (read only)
- **managed_paths** (object) -- Path of managed filesystem elements

- **minute** (string) -- The minute setting for the cron schedule
- **month_of_year** (string) -- The month setting for the cron schedule
- **name** (string) -- Schedule Name (required)
- **site** (string) -- Related site to the schedule (required)
- **url** (string) -- (read only)

PATCH /schedules/{id}/

Parameters:
- **id** (string) --

Request JSON Object:
- **day_of_month** (string) -- The day setting for the cron schedule
- **day_of_week** (string) -- The week setting for the cron schedule
- **discovery** (string) --
- **enabled** (boolean) -- If the schedule should be enabled
- **hour** (string) -- The hour setting for the cron schedule
- **id** (integer) -- (read only)
- **managed_paths** (object) -- Path of managed filesystem elements
- **minute** (string) -- The minute setting for the cron schedule
- **month_of_year** (string) -- The month setting for the cron schedule
- **name** (string) -- Schedule Name (required)
- **site** (string) -- Related site to the schedule (required)

Status Codes:
- 200 OK --

Response JSON Object:
- **day_of_month** (string) -- The day setting for the cron schedule
- **day_of_week** (string) -- The week setting for the cron schedule
- **discovery** (string) --
- **enabled** (boolean) -- If the schedule should be enabled
- **hour** (string) -- The hour setting for the cron schedule
- **id** (integer) -- (read only)
- **managed_paths** (object) -- Path of managed filesystem elements
- **minute** (string) -- The minute setting for the cron schedule
- **month_of_year** (string) -- The month setting for the cron schedule
- **name** (string) -- Schedule Name (required)
- **site** (string) -- Related site to the schedule (required)

DELETE /schedules/{id}/

Parameters:
- **id** (string) --

Status Codes:
- 204 No Content --

GET /schedules/{parent_lookup_schedule}/workflows/

Parameters:
- **parent_lookup_schedule** (string) --

Query Parameters:
- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination

and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].fields** (object) -- Mapping of path to operation for task usage
- **results[].id** (integer) -- (read only)
- **results[].site** (string) -- (required)
- **results[].url** (string) -- (read only)
- **results[].workflow** (string) -- The target workflow for the schedule (required)

`POST /schedules/{parent_lookup_schedule}/workflows/`

Parameters:

- **parent_lookup_schedule** (string) --

Request JSON Object:

- **fields** (object) -- Mapping of path to operation for task usage
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **workflow** (string) -- The target workflow for the schedule (required)

Status Codes:

- 201 Created --

Response JSON Object:

- **fields** (object) -- Mapping of path to operation for task usage
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **workflow** (string) -- The target workflow for the schedule (required)

`GET /schedules/{parent_lookup_schedule}/workflows/{id}/`

Parameters:

- **id** (string) --
- **parent_lookup_schedule** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **fields** (object) -- Mapping of path to operation for task usage
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **url** (string) -- (read only)
- **workflow** (string) -- The target workflow for the schedule (required)

`PATCH /schedules/{parent_lookup_schedule}/workflows/{id}/`

Parameters:

- **id** (string) --
- **parent_lookup_schedule** (string) --

Request JSON Object:

- **fields** (object) -- Mapping of path to operation for task usage
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **url** (string) -- (read only)
- **workflow** (string) -- The target workflow for the schedule (required)

Status Codes:

- 200 OK --

Response JSON Object:

- **fields** (object) -- Mapping of path to operation for task usage
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **url** (string) -- (read only)
- **workflow** (string) -- The target workflow for the schedule (required)

`DELETE /schedules/{parent_lookup_schedule}/workflows/{id}/`

Parameters:

- **id** (string) --
- **parent_lookup_schedule** (string) --

Status Codes:

- 204 No Content --

`GET /search/`

API endpoint for file search

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].id** (integer) -- (read only)
- **results[].url** (string) -- (read only)

`POST /search/`

API endpoint for file search

Request JSON Object:

- **filters** (object) -- Metadata filters to apply to search
- **merge** (boolean) -- Whether matching files should be merged
- **metadata_fields** (object) -- Available metadata fields from this search
- **path** (string) -- Directory to search (required)
- **recursive** (boolean) -- Search the target path recursively

- **sites[]** (string) --

Status Codes:

- 201 Created --

Response JSON Object:

- **filters** (object) -- Metadata filters to apply to search
- **merge** (boolean) -- Whether matching files should be merged
- **metadata_fields** (object) -- Available metadata fields from this search
- **path** (string) -- Directory to search (required)
- **recursive** (boolean) -- Search the target path recursively
- **sites[]** (string) --

`GET /search/metadata_fields/`

API endpoint for file search

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].id** (integer) -- (read only)
- **results[].url** (string) -- (read only)

`GET /search/{id}/`

Get paginated results for a given search id.

Parameters:

- **id** (string) --

Query Parameters:

- **page** (integer) -- Number of the page of results to return
- **page_size** (integer) -- Number of results to return per page
- **sort** (string) -- One or more fields to sort results by

Status Codes:

- 200 OK --

Response JSON Object:

- **href** (string) -- (read only)
- **metadata** (object) -- File metadata
- **name** (string) -- Directory or file name (required)
- **path** (string) -- Directory or file path (required)
- **site** (string) -- Site Name

`DELETE /search/{id}/`

API endpoint for file search

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

`GET /sitelinks/`

API endpoint for managing sitelinks.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].datastore** (string) -- (required)
- **results[].datastore_path** (string) -- (required)
- **results[].id** (integer) -- (read only)
- **results[].site** (string) -- (required)
- **results[].site_path** (string) -- (required)
- **results[].url** (string) -- (read only)

`POST /sitelinks/`

API endpoint for managing sitelinks.

Request JSON Object:

- **datastore** (string) -- (required)
- **datastore_path** (string) -- (required)
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **site_path** (string) -- (required)
- **url** (string) -- (read only)

Status Codes:

- 201 Created --

Response JSON Object:

- **datastore** (string) -- (required)
- **datastore_path** (string) -- (required)
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **site_path** (string) -- (required)
- **url** (string) -- (read only)

`GET /sitelinks/{id}/`

API endpoint for managing sitelinks.

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **datastore** (string) -- (required)
- **datastore_path** (string) -- (required)
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **site_path** (string) -- (required)
- **url** (string) -- (read only)

`PATCH /sitelinks/{id}/`

API endpoint for managing sitelinks.

Parameters:

- **id** (string) --

Request JSON Object:

- **datastore** (string) -- (required)
- **datastore_path** (string) -- (required)
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **site_path** (string) -- (required)
- **url** (string) -- (read only)

Status Codes:

- 200 OK --

Response JSON Object:

- **datastore** (string) -- (required)
- **datastore_path** (string) -- (required)
- **id** (integer) -- (read only)
- **site** (string) -- (required)
- **site_path** (string) -- (required)
- **url** (string) -- (read only)

`DELETE /sitelinks/{id}/`

API endpoint for managing sitelinks.

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

`GET /sites/`

API endpoint for managing sites.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is

empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].id** (integer) -- (read only)
- **results[].name** (string) -- Site Name (required)
- **results[].url** (string) -- (read only)

`POST /sites/`

API endpoint for managing sites.

Request JSON Object:

- **bandwidth** (integer) -- speed for site (in Mb/s)
- **elasticsearch_url** (string) -- URL of the Elasticsearch server to use for the Analytics search backend on this site
- **exclude** (object) -- Global workflow excludes for this site
- **file_batch_gb** (integer) -- File batch GB
- **file_batch_size** (integer) -- File batch size
- **id** (integer) -- (read only)
- **include** (object) -- Global workflow includes for this site
- **lock_threshold** (integer) -- Threshold for soft locking snapshot rotations
- **name** (string) -- Site Name (required)
- **pixstor_search_url** (string) -- The base URL for querying the PixStor API
- **url** (string) -- (read only)

Status Codes:

- 201 Created --

Response JSON Object:

- **bandwidth** (integer) -- speed for site (in Mb/s)
- **elasticsearch_url** (string) -- URL of the Elasticsearch server to use for the Analytics search backend on this site
- **exclude** (object) -- Global workflow excludes for this site
- **file_batch_gb** (integer) -- File batch GB
- **file_batch_size** (integer) -- File batch size
- **id** (integer) -- (read only)
- **include** (object) -- Global workflow includes for this site
- **lock_threshold** (integer) -- Threshold for soft locking snapshot rotations
- **name** (string) -- Site Name (required)
- **pixstor_search_url** (string) -- The base URL for querying the PixStor API
- **url** (string) -- (read only)

`GET /sites/{id}/`

API endpoint for managing sites.

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **bandwidth** (integer) -- speed for site (in Mb/s)
- **elasticsearch_url** (string) -- URL of the Elasticsearch server to use for the Analytics search backend on this site
- **exclude** (object) -- Global workflow excludes for this site
- **file_batch_gb** (integer) -- File batch GB
- **file_batch_size** (integer) -- File batch size
- **id** (integer) -- (read only)
- **include** (object) -- Global workflow includes for this site
- **lock_threshold** (integer) -- Threshold for soft locking snapshot rotations
- **name** (string) -- Site Name (required)
- **pixstor_search_url** (string) -- The base URL for querying the PixStor API
- **url** (string) -- (read only)

`PATCH /sites/{id}/`

API endpoint for managing sites.

Parameters:

- **id** (string) --

Request JSON Object:

- **bandwidth** (integer) -- speed for site (in Mb/s)
- **elasticsearch_url** (string) -- URL of the Elasticsearch server to use for the Analytics search backend on this site
- **exclude** (object) -- Global workflow excludes for this site
- **file_batch_gb** (integer) -- File batch GB
- **file_batch_size** (integer) -- File batch size
- **id** (integer) -- (read only)
- **include** (object) -- Global workflow includes for this site
- **lock_threshold** (integer) -- Threshold for soft locking snapshot rotations
- **name** (string) -- Site Name (required)
- **pixstor_search_url** (string) -- The base URL for querying the PixStor API
- **url** (string) -- (read only)

Status Codes:

- 200 OK --

Response JSON Object:

- **bandwidth** (integer) -- speed for site (in Mb/s)
- **elasticsearch_url** (string) -- URL of the Elasticsearch server to use for the Analytics search backend on this site
- **exclude** (object) -- Global workflow excludes for this site
- **file_batch_gb** (integer) -- File batch GB
- **file_batch_size** (integer) -- File batch size
- **id** (integer) -- (read only)
- **include** (object) -- Global workflow includes for this site
- **lock_threshold** (integer) -- Threshold for soft locking snapshot rotations
- **name** (string) -- Site Name (required)
- **pixstor_search_url** (string) -- The base URL for querying the PixStor API
- **url** (string) -- (read only)

DELETE /sites/{id}/

API endpoint for managing sites.

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

GET /sites/{id}/health/

API endpoint for managing sites.

Parameters:

- **id** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **bandwidth** (integer) -- speed for site (in Mb/s)
- **elasticsearch_url** (string) -- URL of the Elasticsearch server to use for the Analytics search backend on this site
- **exclude** (object) -- Global workflow excludes for this site
- **file_batch_gb** (integer) -- File batch GB
- **file_batch_size** (integer) -- File batch size
- **id** (integer) -- (read only)
- **include** (object) -- Global workflow includes for this site
- **lock_threshold** (integer) -- Threshold for soft locking snapshot rotations
- **name** (string) -- Site Name (required)
- **pixstor_search_url** (string) -- The base URL for querying the PixStor API
- **url** (string) -- (read only)

GET /tasks/

API endpoint for viewing tasks.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.
- **tasktype** (string) -- Task type
- **state** (string) -- Task state
- **jobId** (integer) -- Job ID

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].job** (integer) -- (required)
- **results[].site** (string) -- The target site for the task

- **results[].started** (string) -- Time that the task started running
- **results[].state** (string) -- (required)
- **results[].taskid** (string) -- Job task ID (required)
- **results[].tasktype** (string) -- (required)
- **results[].url** (string) -- (read only)

`GET /tasks/{taskid}/`

API endpoint for viewing tasks.

Parameters:
- **taskid** (string) --

Status Codes:
- 200 OK --

Response JSON Object:
- **completed** (string) -- Time of the task completion
- **job** (integer) -- (required)
- **moved_data** (string) -- (read only)
- **numfailedfiles** (integer) --
- **numfiles** (integer) --
- **numprocessedfiles** (integer) --
- **numskippedfiles** (integer) --
- **paths** (string) -- (read only)
- **results** (string) -- (read only)
- **runtime** (string) -- (read only)
- **site** (string) -- The target site for the task
- **started** (string) -- Time that the task started running
- **state** (string) -- (required)
- **taskid** (string) -- Job task ID (required)
- **tasktype** (string) -- (required)
- **url** (string) -- (read only)

`GET /users/`

API endpoint for managing users.

Query Parameters:
- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:
- 200 OK --

Response JSON Object:
- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].date_joined** (string) --
- **results[].email** (string) --

- **results[].first_name** (string) --
- **results[].groups[].id** (integer) -- (read only)
- **results[].groups[].name** (string) -- (required)
- **results[].groups[].url** (string) -- (read only)
- **results[].id** (integer) -- (read only)
- **results[].is_active** (boolean) -- Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
- **results[].last_login** (string) --
- **results[].last_name** (string) --
- **results[].url** (string) -- (read only)
- **results[].username** (string) -- Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. (required)

`POST /users/`

API endpoint for managing users.

Request JSON Object:

- **email** (string) --
- **first_name** (string) --
- **groups[]** (string) --
- **last_name** (string) --
- **password** (string) -- (required)
- **username** (string) -- Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. (required)

Status Codes:

- 201 Created --

Response JSON Object:

- **email** (string) --
- **first_name** (string) --
- **groups[]** (string) --
- **last_name** (string) --
- **password** (string) -- (required)
- **username** (string) -- Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. (required)

`GET /users/{username}/`

API endpoint for managing users.

Parameters:

- **username** (string) --

Status Codes:

- 200 OK --

Response JSON Object:

- **date_joined** (string) --
- **email** (string) --
- **first_name** (string) --
- **groups[].id** (integer) -- (read only)
- **groups[].name** (string) -- (required)
- **groups[].url** (string) -- (read only)
- **id** (integer) -- (read only)

- **is_active** (boolean) -- Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
- **last_login** (string) --
- **last_name** (string) --
- **url** (string) -- (read only)
- **username** (string) -- Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. (required)

`PATCH /users/{username}/`

API endpoint for managing users.

Parameters:
- **username** (string) --

Request JSON Object:
- **email** (string) --
- **first_name** (string) --
- **groups[]** (string) --
- **last_name** (string) --
- **password** (string) --

Status Codes:
- 200 OK --

Response JSON Object:
- **email** (string) --
- **first_name** (string) --
- **groups[]** (string) --
- **last_name** (string) --
- **password** (string) --

`DELETE /users/{username}/`

API endpoint for managing users.

Parameters:
- **username** (string) --

Status Codes:
- 204 No Content --

`POST /users/{username}/activate/`

Activates user account with given username.

Parameters:
- **username** (string) --

Status Codes:
- 201 Created --

`POST /users/{username}/deactivate/`

Deactivates user account with given username.

Parameters:
- **username** (string) --

Status Codes:
- 201 Created --

`GET /workflows/`

API endpoint for viewing workflows.

Query Parameters:

- **page** (integer) -- A page number within the paginated result set. When not given, first page is retrieved by default.
- **page_size** (integer) -- Number of results to return per page. Page size parameter can be a number between `20` and `100`. For disabling pagination and retrieving all results, `0` should be given. When page size parameter is empty or `<20`, `20` results are returned by default. When page size parameter `>100`, `100` results are returned by default.

Status Codes:

- 200 OK --

Response JSON Object:

- **count** (integer) -- (required)
- **next** (string) --
- **previous** (string) --
- **results[].discovery** (string) --
- **results[].enabled** (boolean) -- Is the workflow available for use?
- **results[].fields** (object) --
- **results[].filter_rules** (object) --
- **results[].icon_classes** (object) --
- **results[].id** (integer) -- (read only)
- **results[].label** (string) -- Friendly name of the workflow (required)
- **results[].name** (string) -- Name of Workflow (required)
- **results[].schedule_only** (boolean) -- Workflow only callable inside of a schedule
- **results[].visible** (boolean) -- Is the workflow visible on the UI?

`POST /workflows/`

API endpoint for viewing workflows.

Request JSON Object:

- **discovery** (string) --
- **enabled** (boolean) -- Is the workflow available for use?
- **fields** (object) --
- **filter_rules** (object) --
- **icon_classes** (object) --
- **id** (integer) -- (read only)
- **label** (string) -- Friendly name of the workflow (required)
- **name** (string) -- Name of Workflow (required)
- **schedule_only** (boolean) -- Workflow only callable inside of a schedule
- **visible** (boolean) -- Is the workflow visible on the UI?

Status Codes:

- 201 Created --

Response JSON Object:

- **discovery** (string) --
- **enabled** (boolean) -- Is the workflow available for use?
- **fields** (object) --
- **filter_rules** (object) --

- **icon_classes** (object) --
- **id** (integer) -- (read only)
- **label** (string) -- Friendly name of the workflow (required)
- **name** (string) -- Name of Workflow (required)
- **schedule_only** (boolean) -- Workflow only callable inside of a schedule
- **visible** (boolean) -- Is the workflow visible on the UI?

`GET /workflows/{id}/`

API endpoint for viewing workflows.

Parameters:
- **id** (string) --

Status Codes:
- 200 OK --

Response JSON Object:
- **discovery** (string) --
- **enabled** (boolean) -- Is the workflow available for use?
- **fields** (object) --
- **filter_rules** (object) --
- **icon_classes** (object) --
- **id** (integer) -- (read only)
- **label** (string) -- Friendly name of the workflow (required)
- **name** (string) -- Name of Workflow (required)
- **schedule_only** (boolean) -- Workflow only callable inside of a schedule
- **visible** (boolean) -- Is the workflow visible on the UI?

`PATCH /workflows/{id}/`

API endpoint for viewing workflows.

Parameters:
- **id** (string) --

Request JSON Object:
- **discovery** (string) --
- **enabled** (boolean) -- Is the workflow available for use?
- **fields** (object) --
- **filter_rules** (object) --
- **icon_classes** (object) --
- **id** (integer) -- (read only)
- **label** (string) -- Friendly name of the workflow (required)
- **name** (string) -- Name of Workflow (required)
- **schedule_only** (boolean) -- Workflow only callable inside of a schedule
- **visible** (boolean) -- Is the workflow visible on the UI?

Status Codes:
- 200 OK --

Response JSON Object:
- **discovery** (string) --
- **enabled** (boolean) -- Is the workflow available for use?
- **fields** (object) --
- **filter_rules** (object) --
- **icon_classes** (object) --

- **id** (integer) -- (read only)
- **label** (string) -- Friendly name of the workflow (required)
- **name** (string) -- Name of Workflow (required)
- **schedule_only** (boolean) -- Workflow only callable inside of a schedule
- **visible** (boolean) -- Is the workflow visible on the UI?

DELETE /workflows/{id}/

API endpoint for viewing workflows.

Parameters:

- **id** (string) --

Status Codes:

- 204 No Content --

Tools

The following sections document add-on tools for Ngenea Hub

ngclient

NGCLIENT

SYNOPSIS

ngclient authenticate (-u USERNAME [-p PASSWORD] | -T TOKEN) [-k NAME]

ngclient migrate path... [-s site] [-r] [-p] [options...]

ngclient recall path... [-s site] [-r] [options...]

ngclient send path... [-s source] -t target [options...]

ngclient workflows COMMAND

ngclient features COMMAND

DESCRIPTION

ngclient is a CLI wrapper for the default Ngenea Hub workflows - migrate, recall, and send. It also provides a mechanism for generating authentication tokens.

ngclient settings can be read from a config file, rather than being passed on the command line. See **ngenea-client.conf(5)** for more information on the configuration format. CLI flags take precedence over config file settings.

The authenticate command can be used to generate a client key from a username or access token. The generated client key will be printed to stdout. That client key

can then be used with the workflow sub-commands, either via the --client-key flag, or by saving it to the **ngenea-client.conf(5)** configuration file.

The workflows command group contains sub-commands for interacting with workflows, such as listing workflows and importing new one. See **ngclient-workflows(1)** for more details.

The features command group contains sub-commands for listing, enabling, or disabling feature flags. See **ngclient-features(1)** for more details.

OPTION SUMMARY

```
      path...                 One paths to call the workflow on

-T, --access-token TOKEN      Access token to authenticate with
-u, --username USERNAME       Username to authenticate with
-p, --password PASSWORD       Password for the authentication username
-k, --key-name NAME           Unique name for the client key

    --base-url                Base URL of the {{ brand_name }} API

-c, --config CONFIG           Alternative configuration file path
    --client-key KEY          Client API key to authenticate with

-s, --site SITE               Site to perform the workflow on
-t, --target TARGET           Site to send files to

-d, --no-wait                 Exit after job is submitted, don't wait
for it to complete
    --timeout SECONDS         wait for completion timeout. If not set,
wait indefinitely

-r, --recursive               Perform task recursively.
-p, --premigrate              Premigrate files from site
-H, --hydrate                 Hydrate files on the send target site

-h, --help                    Print help message and exit
```

OPTIONS

- **-T**, **--access-token**

  An access token to generate a client key with.

- **-u**, **--username**

  Username to generate a client key with.

- **-p**, **--password**

  Password to use in combination with --username

If --username is specified and --password isn't, you will be prompted to enter a password interactively. This may be preferrable so that the password doesn't appear in shell history.

- **-k**, **--key-name**

  A unique name to assign when generating a client key.

  This will be displayed in the Ngenea Hub UI

  If not specified, a random uuid will be generated for the key name.

- **--base-url**

  Base URL of the Ngenea Hub API, which operations will be performed against.

  This can be used to perform Ngenea Hub operations on a remote server.

  If not specified, the default is http://localhost:8000/api

- **-c**, **--config**

  The path to an alternative configuration file.

  If not specified, the default configuration paths will be used. The default paths are in the user's HOME directory $HOME/.config/ngenea/ngenea-client.conf, and the global configuration at /etc/ngenea/ngenea-client.conf

  See **ngenea-client.conf(5)** for more information on the configuration format.

  Command line options take precedence over any corresponding config file settings.

- **--client-key**

  Ngenea Hub authentication client key.

  This can be generated via the Ngenea Hub REST API, or using **ngclient(1)** authenticate

- **-s**, **--site**

  Site to use for workflows.

  For migrate and recall, this is the site where the workflows execute. For send, this is the source site, from which files are sent.

  Site does not have to match the node where **ngclient** is being called. This can be used to migrate/recall/send files from a remote site.

  Note - shell-globbing will be evaluated on the local node. For a remote site, the files that the glob would match may differ.

- **-t**, **--target**

Target site for the send workflow

- **-d**, **--no-wait**

  Don't wait for workflows to complete.

  By default, **ngclient** will wait for the workflow to complete, subject to --timeout.

  With this flag, **ngclient** will exit immediately. The workflow will continue to execute independently. In that case, the workflow can be monitored in the Ngenea Hub UI

- **--timeout**

  How long to wait for workflows to complete, in seconds.

  If not specified, **ngclient** will wait indefinitely.

  If the workflow doesn't complete within the timeout, the client will exit with an error. The workflow itself may continue to execute.

- **-r**, **--recursive**

  Migrate or recall files and directories recursively.

- **-p**, **--premigrate**

  Premigrate files

  Premigrated files are migrated, but the data is kept resident.

- **-H**, **--hydrate**

  Controls whether the send workflow 'hydrates' files on the target.

  If false, files are only reverse stubbed on the target.

- **-h**, **--help**

  Prints the help message.

EXAMPLES

GENERATE A CLIENT KEY

```
$ ngclient authenticate --username pixadmin -k ngclient-pixadmin
pixadmin's password:

jDBh2cRk6.LswQfylT2BtGiqtYUWhMB1iipJmQNgr
```

RECALL A FILE

```
ngclient recall /mmfs1/data/hello.txt -s site1 --client-key
jDBh2cRk6.LswQfylT2BtGiqtYUWhMB1iipJmQNgr
```

For brevity, the site and client-key can be saved to the config file

PREMIGRATE A DIRECTORY RECURSIVELY

Assuming the site and client key has been saved to the config file

```
ngclient migrate /mmfs1/data/sample_data/cats -p -r
```

SEND A FILE TO A REMOTE SITE

```
ngclient send /mmfs1/data/hello.txt -s site1 -t site2 --hydrate
```

AVAILABILITY

Distributed as part of the ngenea-hub-client rpm, or the ngclient wheel (Python) for non-Red Hat based systems.

The ngclient wheel can be installed and run on any operating system.

Note - transparent_recall(1) is packaged along with ngclient, but transparent_recall will only work on Unix-based operating systems.

SEE ALSO

ngenea-client.conf(5), ngclient-workflows(1), ngclient-features(1), transparent_recall(1), ngmigrate(1), ngrecall(1)

LICENSE

---

ngclient-workflows

NGCLIENT-WORKFLOWS

SYNOPSIS

ngclient workflows list [workflow-id] [options...]

ngclient workflows import workflow-file [options...]

ngclient workflows update workflow-id workflow-file [options...]

ngclient workflows delete workflow-id [options...]

DESCRIPTION

The list command is used to list one or more existing workflows from Ngenea Hub. By default, workflows are output in json format, one per line. The --yaml flag can be used to output as yaml.

The import command can be used to import a new, custom workflow from a json or yaml formatted file.

Currently it is not possible to invoke these custom workflows from **ngclient**, once created. They can be invoked from the Ngenea Hub UI or via the REST API.

The update command can be used to update an existing workflow from a json or yaml formatted file. The file can contain only the fields you want to change to perform a partial update, or a whole workflow definition for a full replacement.

NOTE - it's not possible to make partial changes to the fields or filter_rules blocks. They can only be replaced as a whole.

The delete command can be used to delete an existing workflow, by id.

Base URL and API key settings can be read from a config file, rather than being passed on the command line. See **ngenea-client.conf(5)** for more information on the configuration format. CLI flags take precedence over config file settings.

Interacting with workflows requires Ngenea Hub authentication. The **ngclient(1)** authenticate command can be used to generate a client key from a username or access token.

OPTION SUMMARY

```
      workflow-id           Unique workflow identifier
      workflow-file         File containing a custom workflow
definition

    --yaml                  List workflows in yaml format

    --base-url              Base URL of the {{ brand_name }} API

-c, --config CONFIG         Alternative configuration file path
```

```
    --client-key KEY       Client API key to authenticate with

-h, --help                 Print help message and exit
```

OPTIONS

- **workflow-file**

  Path to a json or yaml formatted file, containing a workflow definition.

  If '-' is used, the workflow definition will be read from stdin.

  The workflow format is described in the main documentation, section '4.4. Custom Workflows'

- **--yaml**

  List workflows in yaml format.

  By default, workflows are output in json format, one per line (jsonl).

  The --yaml flag will output the workflows in structured yaml format. If multiple workflows are being listed, each one will be separated by a blank line.

- **--base-url**

  Base URL of the Ngenea Hub API, which operations will be performed against.

  This can be used to perform Ngenea Hub operations on a remote server.

  If not specified, the default is http://localhost:8000/api

- **-c**, **--config**

  The path to an alternative configuration file.

  If not specified, the default configuration paths will be used. The default paths are in the user's HOME directory $HOME/.config/ngenea/ngenea-client.conf, and the global configuration at /etc/ngenea/ngenea-client.conf

  See **ngenea-client.conf(5)** for more information on the configuration format.

  Command line options take precedence over any corresponding config file settings.

- **--client-key**

  Ngenea Hub authentication client key.

  This can be generated via the Ngenea Hub REST API, or using **ngclient(1)** authenticate

- **-h**, **--help**

  Prints the help message.

EXAMPLES

GENERATE A CLIENT KEY

```
$ ngclient authenticate --username pixadmin -k ngclient-pixadmin
pixadmin's password:

jDBh2cRk6.LswQfylT2BtGiqtYUWhMB1iipJmQNgr
```

The following examples assume that the client key has been saved in the default config file.

GET AN EXISTING WORKFLOW

```
$ ngclient workflows list 1
{"id": 1, "name": "migrate", "label": "Migrate", "icon_classes":
["fa fa-cloud fa-stack-2x text-success", "fa fa-angle-up fa-stack-2x
text-light"], "discovery": "recursive", "enabled": true, "visible":
true, "fields": [], "filter_rules": [{"type": "all", "state": "all",
"action": [{"name": "dynamo.tasks.migrate"}], "description":
"Migrates a file off from a given path"}]}
```

LIST ALL EXISTING WORKFLOWS IN YAML FORMAT

```
$ ngclient workflows list --yaml
id: 1
name: migrate
label: Migrate
discovery: recursive
...
enabled: true
visible: true

id: 2
name: premigrate
label: Premigrate
discovery: recursive
...
enabled: true
visible: true


...
```

(the above example output has been truncated)

IMPORT A CUSTOM WORKFLOW

Using the following workflow definition in json format

```
$ cat overwrite_workflow.json
{"name": "recall_overwrite", "label": "Overwrite On Recall",
```

```
"icon_classes": ["fa fa-cloud fa-stack-2x text-primary", "fa fa-
caret-down fa-stack-2x text-light"], "filter_rules": [{"type": "all",
"state": "all", "action": [{"name": "dynamo.tasks.reverse_stub",
"site": "*destinationsite", "overwrite": true}]}], "fields": [{"name":
"destinationsite", "type": "enum[site]", "label": "Destination Site",
"value": "site"}]}
```

Import the workflow as follows

```
ngclient workflows import overwrite_workflow.json
```

RENAME A WORKFLOW

With the change in yaml format, using '-' to read from stdin

```
echo "name: overwrite_on_recall" | ngclient workflows update 6 -
```

DELETE A WORKFLOW

```
ngclient workflows delete 6
```

AVAILABILITY

Distributed as part of the ngenea-hub-client rpm, or the ngclient wheel (Python) for non-Red Hat based systems.

The ngclient wheel can be installed and run on any operating system.

SEE ALSO

ngclient(1), ngenea-client.conf(5)

LICENSE

---

ngclient-features

NGCLIENT-FEATURES

SYNOPSIS

ngclient features list [options...]

ngclient features enable name [options...]

ngclient features disable name [options...]

DESCRIPTION

The list command is used to list available feature flags for .

The enable and disable commands can be used to enable a named feature in Ngenea Hub.

Base URL and API key settings can be read from a config file, rather than being passed on the command line. See **ngenea-client.conf(5)** for more information on the configuration format. CLI flags take precedence over config file settings.

Interacting with features requires Ngenea Hub authentication. The **ngclient(1)** authenticate command can be used to generate a client key from a username or access token.

OPTION SUMMARY

```
      name                  Name of the feature to enable or disable

    --json                  List features in json format

    --base-url              Base URL of the {{ brand_name }} API

-c, --config CONFIG         Alternative configuration file path
    --client-key KEY        Client API key to authenticate with

-h, --help                  Print help message and exit
```

OPTIONS

- **--json**

  List features in json format.

  By default, the list command will report features in a table-based format. The --json flag will report features in json format instead, one per line.

- **--base-url**

  Base URL of the Ngenea Hub API, which operations will be performed against.

  This can be used to perform Ngenea Hub operations on a remote server.

If not specified, the default is http://localhost:8000/api

- **-c**, **--config**

  The path to an alternative configuration file.

  If not specified, the default configuration paths will be used. The default paths are in the user's HOME directory $HOME/.config/ngenea/ngenea-client.conf, and the global configuration at /etc/ngenea/ngenea-client.conf

  See **ngenea-client.conf(5)** for more information on the configuration format.

  Command line options take precedence over any corresponding config file settings.

- **--client-key**

  Ngenea Hub authentication client key.

  This can be generated via the Ngenea Hub REST API, or using **ngclient(1)** authenticate

- **-h**, **--help**

  Prints the help message.

EXAMPLES

The following examples assume that the client key has been saved in the default config file.

LIST AVAILABLE FEATURES

```
$ ngclient features list
 [X]  searchui              Enable search features in the UI
 [ ]  bandwidth_controls    Enable bandwidth controls in the UI
 [ ]  rbac                  Enable role-based access controls
```

(The above are just examples and may not reflect actual feature flags)

ENABLE A FEATURE

```
ngclient features enable rbac
```

DISABLE A FEATURE

```
ngclient features disable searchui
```

AVAILABILITY

Distributed as part of the ngenea-hub-client rpm, or the ngclient wheel (Python) for non-Red Hat based systems.

The ngclient wheel can be installed and run on any operating system.

SEE ALSO

ngclient(1), ngenea-client.conf(5)

LICENSE

---

ngenea-client.conf

NGENEACLIENTCONF

SYNOPSIS

The Ngenea Hub client configuration files is used to configure **ngclient(1)** and **transparent_recall(1)**

The default config file locations are in the user's HOME directory $HOME/.config/ngenea/ngenea-client.conf, with a global config at /etc/ngenea/ngenea-client.conf.

If both configuration files exist, the user config will take precedence, with the global config used for any values not specified in the user config.

For example

```
# global config
[settings]
base_url = http://10.172.0.23:8000/api
site = default

# user config
[settings]
site = mysite
```

would result in base_url = http://10.172.0.23:8000/api, since it's not specified in the user config, and site = mysite since the value from the user config takes precedence.

NOTE - unless explicitly specified with the --config flag, both ngclient(1) and transparent_recall(1) will use this same default config files.

If a config file is explicitly specified with --config, the default configs will not be considered at all.

Command line options take precedence over any corresponding config file settings.

FILE FORMAT

ngenea-client.conf(5) uses an ini-style format.

It is made up of key = value lines under the [settings] section header.

```
[settings]
client_key = mykey
```

Boolean type values can be either true, false, yes, or no (case-insensitive)

Additional sections, or unrecognised keys are ignored.

PARAMETERS

- **base_url**

  Base URL of the Ngenea Hub API, which operations will be performed against.

  This can be used to perform Ngenea Hub operations on a remote server.

  If not specified, the default is http://localhost:8000/api

- **client_key**

  Ngenea Hub authentication client key.

  This can be generated via the Ngenea Hub REST API, or using **ngclient(1)** authenticate

- **site**

  The default site to use for workflows.

  For migrate and recall, this is the site where the workflows execute. For send, this is the source site, from which files are sent.

- **wait**

  Whether to wait for workflows to complete.

  If true (default), tools will wait for the workflow to complete, subject to timeout.

If false, tools will exit immediately. The workflow will continue to execute independently. In that case, the workflow can be monitored in the Ngenea Hub UI

- **timeout**

  How long to wait for workflows to complete, in seconds.

  If not set, tools will wait indefinitely.

  If the workflow doesn't complete within the timeout, the client will exit with an error. The workflow itself may continue to execute.

- **hydrate**

  For **ngclient(1)** send, controls whether sent files are hydrated on the target.

  If false, files are only reverse stubbed on the target.

EXAMPLE

```
[settings]
base_url = http://mypixserver:8000/api
client_key = ...
site = mysite
wait = true
timeout = 180
hydrate = true
```

SEE ALSO

ngclient(1), transparent_recall(1)

LICENSE

---

transparent_recall

TRANSPARENT_RECALL

SYNOPSIS

transparent_recall file [--config CONF]

DESCRIPTION

transparent_recall is a tool for recalling individual files via Ngenea Hub

Performing recalls via Ngenea Hub allows for monitoring progress via the Ngenea Hub UI. Individual recall tasks performed on demand, but for reporting are grouped together into one job per hour.

transparent_recall can be called directly to recall files, but typically would be installed as a filesystem policy rule. See **TRANSPARENT RECALL POLICY** for more info.

OPTION SUMMARY

```
      file           One or more directory to export events from

    --config CONF    Alternative configuration file location
-h, --help           Print help message and exit
```

OPTIONS

- **--config**

  The path to an alternative configuration file.

  If not specified, the default configuration paths will be used. The default paths are in the user's HOME directory $HOME/.config/ngenea/ngenea-client.conf, and the global configuration at /etc/ngenea/ngenea-client.conf

  See the **CONFIGURATION** section below and ngenea-client.conf(5) for more information.

  Command line options take precedence over any corresponding config file settings.

- **-h**, **--help**

  Prints the help message.

CONFIGURATION

transparent_recall requires authentication to be able to perform recalls via Ngenea Hub. To authenticate, a valid client_key must be placed in the configuration file.

A client key can be generated via the Ngenea Hub REST API, or using the **ngclient(1)** authenticate command.

Minimally, the configuration must include this client_key, as well as the site where recalls are performed.

```
[settings]
client_key = ...
site = thissite
```

The site must match the the node where the recall was triggered.

transparent_recall will respect any **ngclient(1)** recall configuration options, except for recursive. This includes timeout; by default it will wait indefinitely for the recall to complete.

See **ngenea-client.conf(5)** for more information on the configuration format and additional options.

TRANSPARENT RECALL POLICY

Transparent recall, by definition, is intended to be triggered automatically when an offline file is opened for reading or writing.

To enable transparent recall functionality using **transparent_recall**, the following rules can be added to the filesystem placement policy.

```
define(xattr_stbsz,[INTEGER(XATTR('dmapi.APXstbsz'))])

RULE FileOpen EVENT 'OPEN'
     ACTION(SetDataEvent(0, OP_READ, CASE WHEN xattr_stbsz IS NULL
THEN 0 ELSE xattr_stbsz END) AND
            SetDataEvent(1, OP_WRITE+OP_TRUNC, 0))
     WHERE XATTR('dmapi.APXguuid') IS NOT NULL AND
CountSubstr(MISC_ATTRIBUTES,'V')>0

RULE FileOpen_else EVENT 'OPEN' DIRECTORIES_PLUS

RULE FileData EVENT 'DATA'
     ACTION(system('/usr/bin/python3 /usr/bin/transparent_recall '''
|| getDetail('path_name') || '''')=0)
     WHERE XATTR('dmapi.APXguuid') IS NOT NULL AND
CountSubstr(MISC_ATTRIBUTES,'V')>0

RULE FileData_else EVENT 'DATA' DIRECTORIES_PLUS
```

If transparent recall (EVENT) rules are already installed for ngenea (native), these rules should replace those equivalent rules.

Don't replace any rules besides the ngenea EVENT rules, e.g. don't replace any SET POOL rules.

See **mmchpolicy(1)** for how to change the filesystem placement policy

TROUBLESHOOTING

When attempting to read an offline file, if the read process reports "Operation not permitted", and it's not due to permissions, the most likely cause is that the recall failed.

Logs for the **transparent_recall** command invocation can be found at /var/adm/ras/mmfs.log.latest

Logs for the transparent recall job can be viewed via Ngenea Hub.

WARNING - if reading a file triggers a recall, the read request will block until recall exits; it can't be interrupted (Ctrl+C) or killed (kill -9). If the recall job is 'stuck' and no timeout is set, the only way to make the read process exit is to kill the recall job via Ngenea Hub.

AVAILABILITY

Distributed as part of the ngenea-hub-client rpm, or the ngclient wheel (Python) for non-Red Hat based systems.

Note - **transparent_recall** makes use of flock(2), so can only be used on Unix-base operating systems.

SEE ALSO

ngclient(1), ngenea-client.conf(5), ngrecall(1), mmchpolicy(1)

LICENSE

Contact

Ngenea Hub is provided and supported by:

Pixit Media: https://pixitmedia.com/contact-us/

Arcastream: https://www.arcastream.com/contact/

Ngenea Hub Changelog

```
Ngenea Hub 1.14.1 (2022-07-27)
==============================

( This release only contains changes to ngenea-worker )

Ngenea Hub 1.14.0 (2022-07-15)
==============================

Highlights
----------

- It's now possible to cancel running or stuck jobs via the Jobs
Details page.
- Introducing a new and improved workflow: bi-directional site-sync.
A schedule-only workflow that allows 2 sites to stay in sync.
- New heath status page - showing the status of ngenea hub and it's
workers.
- It's now possible to have site-wide include and exclude lists.

Improvement
-----------

- Autocomplete selection for task types is added to filter tasks
modal. (DYNAMOHUB-522)
- Search filters are displayed as an autocomplete input box
(DYNAMOHUB-546)
- Site selection for search UI is enabled. (DYNAMOHUB-859)
- Sidebar toggler icon is animated (DYNAMOHUB-874)
- Jobs can be filtered by completion time. (DYNAMOHUB-887)
- Path picker component is added for schedule management.
(DYNAMOHUB-889)
- Limit bidirectional_snapdiff discovery schedules to one
bidirectional_sync subscribed workflow (DYNAMOHUB-894)
- Tasks are sorted by their start time by default. (DYNAMOHUB-895)
- bidirectional_snapdiff discovery schedules can be created in the
UI (DYNAMOHUB-899)
- Only allow one bi-directional sync run at a time in a schedule
(DYNAMOHUB-901)
- Jobs can be cancelled on the UI (DYNAMOHUB-911)


Bugfixes
--------

- Fixed users deactivating themselves. (DYNAMOHUB-578)
- Halting the search issue is fixed when the search filters are
changed (DYNAMOHUB-647)
- Fixed jobs filter not syncing with recent jobs issue.
(DYNAMOHUB-821)
- Possible web socket connection flaws are resolved. (DYNAMOHUB-885)
- Fix intermittent transparent recall hangs (DYNAMOHUB-902)
```

- RPM creates a blank config file on new install (DYNAMOHUB-947)

Features

- Added user configurable port to access hub (DYNAMOHUB-198)
- Support for site-global includes and excludes (DYNAMOHUB-613)
- Removed HubSchedule model from DB (DYNAMOHUB-769)
- Added validation to prevent call to workflow/schedule when no work is done (DYNAMOHUB-798)
- Added endpoint for both site specific and hub wide health status reporting (DYNAMOHUB-838)
- Added Site Health Feature (DYNAMOHUB-840)
- Display cron schedule in human readable format (DYNAMOHUB-868)
- Snapdiff task results are now tracked within the hub database (DYNAMOHUB-870)
- Background task to clean up old sync events (DYNAMOHUB-871)
- Workflows can now be set for use only within a schedule (DYNAMOHUB-873)
- New workflow added for two way syncing between sites within schedules (DYNAMOHUB-877)

Documentation

- Documentation on how to set up site sync (DYNAMOHUB-574)
- Search set-up documentation (DYNAMOHUB-781)
- Document using ngclient to interact with feature flags (DYNAMOHUB-843)
- Fixed worker service name in upgrade guide (DYNAMOHUB-882)
- Documentation on how to set up bidirectional site sync (DYNAMOHUB-908)
- Troubleshooting guide (DYNAMOHUB-948)

Ngenea Hub 1.13.0 (2022-06-13)
==============================

Bugfixes

- Timeout is handled on login. (DYNAMOHUB-348)
- If the token is expired, redirect to original target page after login. (DYNAMOHUB-611)
- Removed non-deterministic behaviour while displaying the search bar. (DYNAMOHUB-646)
- Long paths are wrapped to a new line in job details page. (DYNAMOHUB-659)
- Missing ngclient sub-command man pages (DYNAMOHUB-692)
- Fixed Api-Key issues in Resubmit API (DYNAMOHUB-703)
- Added new celery beat that runs every one hour to update the state

from STARTED to FAILURE for inactive tasks (DYNAMOHUB-704)
- Single search web socket connection is managed against working intermittently. (DYNAMOHUB-729)
- Show jobs with no processed files filter is configurable for all job tables. (DYNAMOHUB-744)
- Ensures that a snapshot rotation task is created when a job is resolved regardless of the task chain (DYNAMOHUB-747)
- Ensured that a clearer message is provided when a GPFS error occurs on worker for a stat (DYNAMOHUB-761)
- Recent job counts are corrected for both API and web socket consumer. (DYNAMOHUB-783)
- Fixed schedule job resulting in pending state for snapdiff task (DYNAMOHUB-790)
- Fixed the issue which chases when sumitting bandwidth form with no changes. (DYNAMOHUB-800)
- Bug is fixed when managed_paths is null for a schedule. (DYNAMOHUB-817)
- Removed extra file batch field (DYNAMOHUB-852)
- Sort actions in alphabetical order (DYNAMOHUB-752)

Features

- Feature added for limiting fields in metadata returned (DYNAMOHUB-376)
- feature added for search REST API to provide more_results key if elasticsearch backend has more results than max_results (DYNAMOHUB-550)
- Added support for overriding includes and excludes at runtime (DYNAMOHUB-640)
- Added reverse_stub in default workflows (DYNAMOHUB-681)
- Added recursive flag for move_paths_on_gpfs task in docs workflow_steps.md (DYNAMOHUB-685)
- Feature added for chunk_size to chunk paths in snapdiff tasks (DYNAMOHUB-711)
- Feature added for Id based endpoint for IpAddress Apis GET, DELETE and PATCH (DYNAMOHUB-712)
- Added support for pixstor_search backend in hub (DYNAMOHUB-738)
- Add Datastore managemant and IP address management feature. (DYNAMOHUB-774)
- Updated sync_preference type to choices for site_sync workflow (DYNAMOHUB-787)
- NgeneaHub UI supports having new themes now. (DYNAMOHUB-842)

Documentation

- Upgrade guide documentation (DYNAMOHUB-687)
- Add site-specific configurations to the docs (DYNAMOHUB-739)
- Documentation for default workflows (DYNAMOHUB-741)
- Document how to use search metadata field limiting (DYNAMOHUB-756)

```
Removals
--------

- Removed the site field for IP address objects as they were once
depreciated (DYNAMOHUB-657)


Improvement
-----------

- Job runtime field is introduced and becomes sortable on the UI.
(DYNAMOHUB-714)


Ngenea Hub 1.12.0 (2022-05-20)
==============================

Improvement
-----------

- Hash is used instead of content hash for cache busting.
(DYNAMOHUB-743)
- Fixed not displaying error or failure state jobs with no
operations (DYNAMOHUB-745)
- File browser performance improvement for file status updates.
(DYNAMOHUB-766)
- Log PID instead of process name within the celery logs.
(DYNAMOHUB-784)
- List actions in alphabetical order. (DYNAMOHUB-752)
- Allow deleting of /ipaddresses/{ipaddr}/ if an ip address is
associated with multiple datastores. (DYNAMOHUB-712)
- Add new reverse stub workflow as part of deployment.
(DYNAMOHUB-681)
- Provide mechanism to limit fields of search metadata returned.
(DYNAMOHUB-376)


Bugfixes
--------

- Fixed migration bug with mis-matching schedule IDs during
migration. (DYNAMOHUB-824)
- UI now handles managed_paths being null within a schedule.
(DYNAMOHUB-817)
- Ensured rotate task do not remain stuck in a PENDING state.
(DYNAMOHUB-747)
- Users can now show no-op jobs across all job tables.
(DYNAMOHUB-744)
- Search no longer works intermittently when previous search is
stopped. (DYNAMOHUB-729)
- Searching by gpfs.filesetname now works correctly. (DYNAMOHUB-727)
- When task gets killed, it is no longer marked as "STARTED" in
celery. (DYNAMOHUB-704)
- Jobs can now be resubmited using an API Key. (DYNAMOHUB-703)
```

- Search bar now always shows when you first log into UI. (DYNAMOHUB-646)

Ngenea Hub 1.11.0 (2022-05-09)
==============================

Improvement

- Descriptive messages for all tables (DYNAMOHUB-541)
- Default search filter is changed to core.pathname (DYNAMOHUB-545)
- Workflow details can be inspected on job details page. (DYNAMOHUB-648)
- Table row highlighting is improved visually. (DYNAMOHUB-656)
- Make number of files diplay dialog nicer (DYNAMOHUB-661)
- Elasticsearch url management for sites (DYNAMOHUB-693)
- Added a message when there are more searched results to be displayed (DYNAMOHUB-697)
- Task details preview is disabled when task details JSON is too large. (DYNAMOHUB-719)
- file_size_gb field of sites became managable from UI. (DYNAMOHUB-740)
- Update UI for new discovery schedules, to which multiple workflows can be subscribed. (DYNAMOHUB-772)

Ngenea Hub 1.10.0 (2022-04-01)
==============================

Improvement

- Added user friendly log viewer (DYNAMOHUB-436)
- feature improvement added for filter_results to return clear results for the user (DYNAMOHUB-491)
- Workflow configuration is visible in job details and job list pages (DYNAMOHUB-626)
- Table optional fields are managed by switches in Show/Hide column dialog. (DYNAMOHUB-631)
- Converted runtime value into human readable value (DYNAMOHUB-644)
- Resolved UI Typescript compile issues & unit test issues (DYNAMOHUB-665)
- Tab view for Administration, SiteDetails, ScheduleDetails pages (DYNAMOHUB-666)
- Refactoring on the file browser (DYNAMOHUB-667)
- Refactoring forms on the UI (DYNAMOHUB-668)
- Refactoring tables & sticky table controls (DYNAMOHUB-669)
- UI test coverage is increased. (DYNAMOHUB-670)
- Refactoring on websocket serialization (DYNAMOHUB-680)
- Bandwidth controls are hidden behind a feature flag. (DYNAMOHUB-750)

```
Features
--------

- Support for setting search-related configurations via the REST API
(DYNAMOHUB-474)
- feature added for supporting default filter rules for snapdiff and
recursive discovery tasks (DYNAMOHUB-533)
- API endpoint for setting speed for site (DYNAMOHUB-568)
- Support for setting the IP address for a DataStore (DYNAMOHUB-569)
- Feature added for auto update mechanism of cloud hosted storage
target ips for s3,azure,gcs (DYNAMOHUB-570)
- UI for setting bandwidth controls (DYNAMOHUB-571)
- Updated docs with endpoint flag for migrate tasks (DYNAMOHUB-580)
- Snapdiff discovery tasks now only rotate their snapshot on no
errors in other tasks. (DYNAMOHUB-630)
- Ability to scan for worker nodes via the REST API (DYNAMOHUB-638)


Bugfixes
--------

- Display workflow name as the tooltip text (DYNAMOHUB-554)
- fixed issue job fails while other tasks are running by refreshing
the job when there are running/pending remaining tasks
(DYNAMOHUB-577)
- Fixed clicking to search result path for hidden items
(DYNAMOHUB-683)


Deprecations and Removals
-------------------------

- Setting IP address for a site is deprecated as of release 1.9 and
will be removed in 1.10 (DYNAMOHUB-569)


Ngenea Hub 1.9.0 (2022-02-23)
=============================

Features
--------

- Model for object stores (S3, GCS, etc.) (DYNAMOHUB-207)
- SiteLink model for connecting sites to datastores (DYNAMOHUB-224)
- Worker node model and monitoring (DYNAMOHUB-563)
- Feature added for configuring file batch size in site model
(DYNAMOHUB-572)
- Job started time, completed time, runtime fields are accessible on
the UI. (DYNAMOHUB-573)
- Modifications done for file count for all tasks to report job
statistics (DYNAMOHUB-587)
- Added endpoint to job API to cancel all on-going tasks
(DYNAMOHUB-71)
```

```
Improvement
-----------

- Allow user to search job id from the job table (DYNAMOHUB-411)
- Complex queries can be made with the search bar (DYNAMOHUB-547)
- changed tooltip to display workflow name (DYNAMOHUB-554)
- Update frontend depedancy: follow-redirects to address
CVE-2022-0536 (DYNAMOHUB-637)


Bugfixes
--------

- Job filter preferences are remembered on the jobs page
(DYNAMOHUB-427)
- Display correct page size after applying the filters.
(DYNAMOHUB-531)
- Jobs page table glitch is fixed. (DYNAMOHUB-632)


Ngenea Hub 1.8.0 (2022-02-08)
=============================

Improvement
-----------

- Usability improvements on task detail modal (DYNAMOHUB-420)
- Automatically create hub environment auth file if it does not
exist (DYNAMOHUB-576)


Features
--------

- Added UI element for "choices" field type (DYNAMOGUB-551)
- Added ngenea locking mode support for default workflow operations
(DYNAMOHUB-537)


Bugfixes
--------

- fixed misleading error messages on login page (DYNAMOHUB-348)
- changed return code of /api/file/workflow endpoint from 200 to 201
(DYNAMOHUB-488)
- Informative messages when the worker is shutdown (DYNAMOHUB-514)
- Usability improvements on tables and schedule pages (DYNAMOHUB-517)
- Ensure debug mode isn't enabled in production (DYNAMOHUB-518)
- fixed the warning issue raised for timezone in task started
(DYNAMOHUB-519)
- Removed default option as recursive set for discovery for /api/
file/workflow (DYNAMOHUB-534)
- Added jobid in signatures for recursive discovery task
```

(DYNAMOHUB-543)
- Ensured that the detault value of a field is respected in the UI (DYNAMOHUB-558)
- Ensure secure file permissions on hub auth configuration file (DYNAMOHUB-561)
- Ensured that the workflow API route has full validation on JSON fields (DYNAMOHUB-562)
- More detailed messages for file browser when the worker is not available (DYNAMOHUB-566)

Ngenea Hub 1.7.0 (2022-01-24)
=============================

Improvement

- feature improvement added choices as field type (DYNAMOHUB-154)
- Record and report errors when performing searches (DYNAMOHUB-365)
- Search UI has now filtering option. Before submitting a search, users can add more filters to narrow down the search. (DYNAMOHUB-374)
- Time range selection for job filtering (DYNAMOHUB-403)
- Multiselect functionality on the job state field for job filtering is added. (DYNAMOHUB-449)
- Owner selection while filtering jobs (DYNAMOHUB-457)
- All alerts are manually closed now. (DYNAMOHUB-458)
- Job state and job type selection is swapped visually on the job filtering dialog (DYNAMOHUB-468)
- Replaced "In progress" label with "Started" for jobs (DYNAMOHUB-470)
- Performance improvement on job table and auto refresh toggle for web socket live updates (DYNAMOHUB-480)
- feature improvement added for snapdiff tasks numfiles (DYNAMOHUB-510)

Features

- feature improvement added for runtime fields to have default values for choices type (DYNAMOHUB-153)
- feature added for choices runtime field to support list of objects (DYNAMOHUB-154)
- Added support for providing file states for non-discovery workflows (DYNAMOHUB-344)
- Allowed the use of multiple generic rules within the recursive discovery task (DYNAMOHUB-476)
- API endpoint for feature flags (DYNAMOHUB-482)
- Search UI is a configurable feature now. (DYNAMOHUB-483)
- ngclient sub-commands for listing and setting feature flags (DYNAMOHUB-484)
- Feature API is used for enabling and disabling the usage of UI elements. (DYNAMOHUB-485)

```
Bugfixes
--------

- Number of search results per site is 200 results by default
(DYNAMOHUB-374)
- fixed issue by adding fields in job model instance for resubmit
tasks (DYNAMOHUB-465)
- Performance improvements on the Jobs page (DYNAMOHUB-466)
- A timeout is added for failed search operations. If no results are
found in 100 seconds, the error will show up and the search will
halt. (DYNAMOHUB-498)
- Search results can be sorted by metadata fields. (DYNAMOHUB-557)


Ngenea Hub 1.6.0 (2021-12-21)
=============================

Improvement
-----------

- return extended attributes in file API (DYNAMOHUB-201)
- Add support for `--endpoint` flag on recall and reverse stub tasks
(DYNAMOHUB-361)
- Support for workflows without a discovery task, through the API
(DYNAMOHUB-402)
- limit the number of hub celery worker threads to 2 (DYNAMOHUB-405)
- Include error type when reporting task errors (DYNAMOHUB-418)
- Apply dynamo.tasks.remove_location_xattrs_for_moved to send and
snapdiff workflows to better handle moving files (DYNAMOHUB-422)
- Improve handling of file system xattr "stale nfs handle" message
scenarios (DYNAMOHUB-439)
- Disable validation of non-included fields when patching workflows
via api (DYNAMOHUB-456)

Features
--------

- Set graceful timeout on stat requests, default to 10 seconds.
(DYNAMOHUB-338)
- JWTs are created by using RS256 algorithm (DYNAMOHUB-353)
- Use search result to jump to directory in file browser
(DYNAMOHUB-380)
- Expose JWK set used in token verification via API endpoint
(DYNAMOHUB-430)
- Use Hub authentication for Grafana access (DYNAMOHUB-431)
- Initial framework and metrics for Grafana dashboads (DYNAMOHUB-432)
- ngclient command to import and export custom workflows
(DYNAMOHUB-434)
- NgeneaHub UI management of scheduled workflows (DYNAMOHUB-435)


Bugfixes
--------
```

- Migration path not correct when upgrading from 1.3.0 (DYNAMOHUB-397)
- Fix incorrect PDF Documention download link (DYNAMOHUB-440)
- Fixed issue with task start time formatting (DYNAMOHUB-441)
- Custom workflow steps can get wiped on upgrade (DYNAMOHUB-442)
- Fix incorrect download link for ngenea client (DYNAMOHUB-444)
- Mark tasks as completed when skipped (DYNAMOHUB-477)
- Fix date string parsing error in update_task celery task (DYNAMOHUB-508)

Ngenea Hub 1.5.0 (2021-11-22)
==========

Features

- Workflows can be enabled/disabled. Also, workflows can be available to use via the API only. (DYNAMOHUB-167)
- File browser can be filtered by filetype, size, change date, accessed date. (DYNAMOHUB-196)
- Job list filtering options are extended. Jobs can be filtered by site and path prefix information too. (DYNAMOHUB-299)
- Endpoint to submit search requests (DYNAMOHUB-319)
- Endpoint for retrieving search results (DYNAMOHUB-320)
- Submit async search tasks and store results (DYNAMOHUB-321)
- Periodically remove old search results (DYNAMOHUB-328)
- Support for search backend configurations (DYNAMOHUB-329)
- Added the option to provide the django secret key via the environment (DYNAMOHUB-342)
- Include available metadata fields with search results (DYNAMOHUB-382)
- Add job completion and run time to metadata (DYNAMOHUB-63)

Bugfixes

- Set task status to ERROR if any file failed (DYNAMOHUB-161)
- Fixed job processed and failed files incorrectly returning empty lists (DYNAMOHUB-264)
- Fix transparent recall policy to handle paths with whitespace (DYNAMOHUB-324)
- Retry on error when waiting for transparent recall (DYNAMOHUB-336)

Improved Documentation

- Clarify transparent recall documentation (DYNAMOHUB-324)

Ngenea Hub 1.4.0 (2021-10-15)
==========

```
Features
--------

- Consolidate the 3 ngenea-worker systemd services to a single
service for easier management. (DYNAMOHUB-149)


Dynamohub 1.3.0 (2021-09-22)
============================

Features
--------

- “live” changes to file status (DYNAMOHUB-105)
- Transparent Recall logging (DYNAMOHUB-100)
- Expose Groups model via REST API (DYNAMOHUB-277)
- Option to show/hide hidden objects in the filebrowser
(DYNAMOHUB-262)
- UI for setting IP addresses against Site (DYNAMOHUB-232)
- Extend site model to store it's IP addresses (DYNAMOHUB-229)
- ClientKey UI (DYNAMOHUB-143)
- PATCH support for /workflow/ endpoint (DYNAMOHUB-285)
- Spinners on UI components waiting for API response (DYNAMOHUB-281)
- Having visible alerts or redirects according to the websocket
warnings (DYNAMOHUB-276)
- Websocket request should wait for response before sending another
request (DYNAMOHUB-272)
- Reconsider websockets broadcast approach (DYNAMOHUB-271)
- Revamp 404 page (DYNAMOHUB-252)
- Allow the client-key to access all routes excluding user and
client-key (DYNAMOHUB-243)
- Control number of worker threads (DYNAMOHUB-237)
- Change colours for paginated table (DYNAMOHUB-218)
- Make action dropdown wider (DYNAMOHUB-212)
- Make task table sortable (DYNAMOHUB-202)
- Execute the production code via gunicorn (DYNAMOHUB-129)
- Task for checking the current filesets residing on a site
(DYNAMOHUB-251)
- Site worker snapdiff celery discovery task (DYNAMOHUB-245)
- Ability to submit tasks to an existing job (DYNAMOHUB-215)
- Ngenea Hub remote client (DYNAMOHUB-213)


Bugfixes
--------

- Workers can timeout on waiting for response from rabbitmq
(DYNAMOHUB-291)
- Fails to open directories with strange file names (DYNAMOHUB-288)
- Swagger logs exception (DYNAMOHUB-283)
- Prevent resubmitting a job multiple times in quick succession
(DYNAMOHUB-274)
- "Failure" job filter only shows failures for one site
```

(DYNAMOHUB-273)
- Actions button visible on the user profile page (DYNAMOHUB-270)
- Unable to deactivate users (DYNAMOHUB-269)
- Update user profile "save" button is always enabled (DYNAMOHUB-268)
- Inaccurate "last login" info (DYNAMOHUB-267)
- Sort order of jobs (DYNAMOHUB-266)
- Browser tries to show contents of a non-directory (DYNAMOHUB-260)
- Fix frontend issue when submitting large number of files to a workflow (DYNAMOHUB-259)
- Clicking on "Owner" for a clientkey gives a 404 (DYNAMOHUB-253)
- Include ngeneahub-frontend in ngeneahub-images RPM (DYNAMOHUB-242)
- Resubmitting a job doesn't use the same discovery method (DYNAMOHUB-239)
- Some overall job stats don't update until the dir walk is complete (DYNAMOHUB-210)
- Files without a file extension are listed as both directory and folder (DYNAMOHUB-200)

Dynamohub 1.2.0 (2021-08-11)
============================

Features

- Update RabbitMQ to 3.9 (DYNAMOHUB-223)
- paginate tasks in /api/jobs/<id> (DYNAMOHUB-204)
- Group transparent recall tasks (DYNAMOHUB-216)
- Ability to submit non-recursive tasks (DYNAMOHUB-214)

Bugfixes

- Rest API can not be authenticated with an access token (DYNAMOHUB-227)
- Fix client side pagination for task list in job details (DYNAMOHUB-219)
- List of files is job details is empty (DYNAMOHUB-211)
- Using a boolean as a workflow step parameter fails (DYNAMOHUB-208)

Dynamohub 1.1.0 (2021-06-19)
============================

Features

- Send data between different sites via the UI (DYNAMOHUB-124)
- Rework ngenea output parsing (DYNAMOHUB-115)
- Rebuild UI using React (DYNAMOHUB-103)
- Improve stat task to make directory handling explicit (DYNAMOHUB-188)
- step argument: skip-check-hash (DYNAMOHUB-176)
- step argument: overwrite-remote (DYNAMOHUB-175)
- step argument: overwrite-local (DYNAMOHUB-174)
- step argument: stub-size (DYNAMOHUB-173)

```
- React 404 should not change URL (DYNAMOHUB-166)
- Support file delete workflow (DYNAMOHUB-164)
- Recent jobs list should show the most recent jobs, nevermind what
state they are in (DYNAMOHUB-147)
- Job reporting scaling improvements (DYNAMOHUB-84)

Bugfixes
--------

- `api/users/` endpoint throwing errror when a username containing
"." exists (DYNAMOHUB-187)
- "failed to acquire a DMAPI lock EXCL immediately; keeping
trying..." is treated as an error (DYNAMOHUB-186)
- Job reporting not providing correct counts (DYNAMOHUB-185)
- Resubmit button should not be enabled when not usable
(DYNAMOHUB-184)
- Using UI to trigger workflow fails (DYNAMOHUB-177)
- Attempting to post to /api/file/workflow with only a token causes
500 (DYNAMOHUB-171)
- Bug when a directory is in intermediate selection state and closed
(DYNAMOHUB-170)

Dynamohub 1.0.4 (2021-06-10)
============================

Features
--------

- Allow overriding the site on a per-step basis (DYNAMOHUB-151)
- Submission time arguments to workflow support (DYNAMOHUB-150)
- Validate runtime fields (DYNAMOHUB-136)
- Show verbose error information on failure (DYNAMOHUB-156)
- New task: reverse stub (DYNAMOHUB-155)
- Report task types with more meaningful names (DYNAMOHUB-152)

Bugfixes
--------

- Fix bug where sub-directories can fail to be processed
(DYNAMOHUB-148)
- recursive_action failures report as success (DYNAMOHUB-142)

Dynamohub 1.0.3 (2021-05-19)
============================

Features
--------

- Refactor existing ngenea hub tasks to message passing format
(DYNAMOHUB-140)
- Support static arguments to a step (DYNAMOHUB-137)
- Provide existing actions as default workflows (DYNAMOHUB-131)
- enable easy access to dbshell (DYNAMOHUB-134)
- Workflows are now defined dynamically (DYNAMOHUB-132)
```

```
- Allow users to create (API) client keys (DYNAMOHUB-141)

Bugfixes
--------

- Fixed an issue with submitting workflows with Client Key
(DYNAMOHUB-144)
- Fixed warnings generated from auto field (DYNAMOHUB-133)

Dynamohub 1.0.2 (2021-04-26)
============================

Features
--------

- Expose JWT Login API endpoints (DYNAMOHUB-90)
- Add Swagger API Documentation (DYNAMOHUB-94)
- Validate that site names do not end with the suffix we use to
identify queue types (DYNAMOHUB-85)
- Allow middle-click/right-click to open navbar links in new tabs
(DYNAMOHUB-83)
- Ship ngeneahub cli tool as a venv with docker-compose included
(DYNAMOHUB-81)
- Improve performance of job page by doing pagination server-side
(DYNAMOHUB-56)

Dynamohub 1.0.1 (2021-04-01)
============================

Bugfixes
--------

- Symlinks cause file browser to fail (DYNAMOHUB-86)
- "Creation time" should be "ctime" - "Last changed
time" (DYNAMOHUB-78)
- Files and folders with newlines in their names fails, with console
error (DYNAMOHUB-55)

Features
--------

- Standalone UI providing interfaces for core Ngenea/Dynamo
workflows (DYNAMOHUB-1)
- Positioning and content of UI buttons (DYNAMOHUB-38)
- Show percentage completion of job based on # of files transferred
vs remaining (DYNAMOHUB-34)
- Remove redundant environment settings (DYNAMOHUB-82)
- Show a user view page when clicking on a user link (DYNAMOHUB-79)
- Pop-up confirmation of job actions (DYNAMOHUB-57)
- Support for premigrate task (DYNAMOHUB-50)
- Allow viewing directories from different sites (DYNAMOHUB-12)
```

Ngenea Worker Changelog

```
Ngenea Worker 1.14.2 (2022-08-12)
=================================

Bugfixes
--------

- Optimised snapdiff event processing for initial sync runs
(DYNAMOHUB-1014)


Ngenea Worker 1.14.1 (2022-07-27)
=================================

Bugfixes
--------

- Ensure snapdiff soft-lock is removed in the event of an error
(DYNAMOHUB-975)


Ngenea Worker 1.14.0 (2022-07-15)
=================================

Improvement
-----------

- New tasks are introduced for retrieving search metadata fields
(DYNAMOHUB-546)
- Support for different fileset and pool name format in Analytics
2.X (DYNAMOHUB-797)
- Move snapdiff lock file location to the .rotate directory
(DYNAMOHUB-931)


Bugfixes
--------

- Skip moves which have already been applied (DYNAMOHUB-926)


Features
--------

- Added celery command to return the version information of software
running on the current node (DYNAMOHUB-795)
- Condense directory move events from snapdiff (DYNAMOHUB-878)
- The snapdiff task now reports file ctimes with resulting object
changes (DYNAMOHUB-884)


Ngenea Worker 1.13.0 (2022-06-13)
```

```
================================

Improvement
-----------

- feature improvement done in stats to add extended attributes as a
dict (DYNAMOHUB-201)
- feature improvement added for allowing directory deletes with
kwargs (DYNAMOHUB-345)
- Add support for `--endpoint` flag on recall and reverse stub tasks
(DYNAMOHUB-361)
- Workaround for stale file handles when performing reverse stub
(DYNAMOHUB-439)
- Add support for worker auto-scaling (DYNAMOHUB-461)
- Record exact ngenea command in task results (DYNAMOHUB-462)
- Adjusted recursive task to allow for multiple operations based of
generic rules (DYNAMOHUB-476)
- feature improvement added for failure scenario for
remove_location_xattrs_moved (DYNAMOHUB-509)
- feature improvement added for snapdiff tasks to allow only single
path else raise an exception with error message (DYNAMOHUB-515)
- Add Ngenea server and client as rpm package dependencies for
installing the ngenea worker, and require at least version 1.15.
(DYNAMOHUB-694)
- Ensured that ngenea binary stderr capture is not blocked by
reporting the status of a job (DYNAMOHUB-765)

Features
--------

- Add list of available metadata fields to results of Hub searches
performed against Pixstor Search (DYNAMOHUB-734)
- Add mechanism to return specified metadata fields only, in Hub
searches against Pixstor Search (DYNAMOHUB-736)


Bugfixes
--------

- Fixed issue on UUID mismatch using ngrecall overwrite-recall by
comparing local and remote UUID (DYNAMOHUB-464)
- Added interface type dummy to virtual drivers list to exclude it
from physical interface (DYNAMOHUB-757)
- Correctly caught GPFS errors to prevent task from returning stack
trace. (DYNAMOHUB-761)
- Directory moves in move_paths_on_gpfs now move all files along
with the directory themselves (DYNAMOHUB-782)
- Removed the strict format for snapshots within the snapdiff
tracking file (DYANMOHUB-699)


Ngenea Worker 1.12.0 (2022-05-20)
================================
```

```
Features
--------

- Feature added for limiting metadata fields in search_analytics
(DYNAMOHUB-376)


Bugfixes
--------

- Fixed migrate task issue when migrating same filepath to multi
targets (DYNAMOHUB-726)


Ngenea Worker 1.11.0 (2022-05-09)
==============================

Features
--------

- Feature added in search_analytics to hint users if there are
more_results than max_results in elasticsearch backend
(DYNAMOHUB-550)
- Added flag to recall and reverse_stub tasks for gid and uid
(DYNAMOHUB-614)
- Added elasticsearch credentials for analytics search backend
(DYNAMOHUB-689)
- Support for AP-Analytics 2.X search backend (DYNAMOHUB-760)


Bugfixes
--------

- Fix upgrade generating spurious systemd services (DYNAMOHUB-359)
- Added support for CredentialsJSON in GCS ngenea configs
(DYNAMOHUB-655)
- Adjusted the start date of all tasks to return UTC times for
consistency (DYNAMOHUB-663)


Ngenea Worker 1.10.0 (2022-04-01)
==============================

Features
--------

- Feature added to allow directory move in move_paths_on_gpfs
(DYNAMOHUB-542)
- Feature improvement added endpoint flag for migrate task
(DYNAMOHUB-580)
- Adjusted the parsing method to new json output for ngenea 1.15
(DYNAMOHUB-606)
- Added worker task for rotating snapshots outside of the snapdiff
task (DYNAMOHUB-630)
```

```
Bugfixes
--------

- Fix inconsistencies between PixStor Search and Analyitcs search
backends (DYNAMOHUB-556)
- Fixed ngenea error reporting by adding proper status logs
(DYNAMOHUB-624)
- fixed issue for moving files between different filesets
(DYNAMOHUB542)


Ngenea Worker 1.9.0 (2022-02-23)
================================

Features
--------

- changes added for chunk size in recursive action (DYNAMOHUB-472)


Ngenea Worker 1.8.0 (2022-02-23)
================================

Features
--------

- changes added for chunk size in recursive action (DYNAMOHUB-472)


Ngenea Worker 1.8.0 (2022-02-08)
================================

Features
--------

- feature added for migrating empty directories (DYNAMOHUB-490)
- Added flag to recall, reverse_stub and migrate tasks for
controlling ngenea locking levels (DYNAMOHUB-537)


Bugfixes
--------

- No results returned for date range search (DYNAMOHUB-489)
- Number of search results differs for path with trailing slash
(DYNAMOHUB-500)
- added try/except block for get_xattrs for gpfs.FileHeat key error
(DYNAMOHUB-525)
- Fixed an issue where if a file was skipped by a step in a
workflow, they were not being processed by a migrate or recall step
later in the workflow. (DYNAMOHUB-540)
```

```
Ngenea Worker 1.7.0 (2022-01-24)
================================

Features
--------

- Tasks for moving files and directories within cloud storage
(DYNAMOHUB-355)


Bugfixes
--------

- added exception for permission errors (DYNAMOHUB-345)
- Policy error in snapdiff (DYNAMOHUB-450)
- fixed issue raised when submitting a workflow with multiple
filters in recursive task (DYNAMOHUB-476)


Ngenea Worker 1.6.0 (2021-12-21)
================================

Features
--------

- Added task for checking the existence of files in remote storage
(DYNAMOHUB-421)
- Task to remove remote location xattrs if a premigrated file was
moved (DYNAMOHUB-422)


Bugfixes
--------

- Don't require ArcaPix policy driver to be executable
(DYNAMOHUB-419)
- Check for no paths passed to recall and reverse_stub tasks
(DYNAMOHUB-438)
- fixed the status of snapdiff tasks processed files to created
instead of created and deleted (DYNAMOHUB-492)
- Fix issue with parsing ngenea config file when the [General]
section is present (DYNAMOHUB-506)
- Ensured that all chained tasks parse results correctly
(DYNAMOHUB-511)


Ngenea Worker 1.5.0 (2021-11-22)
================================

Features
--------

- Added support for ngenea-hub to delete GPFS files (DYNAMOHUB-164)
```

```
- Added support for ngrecall --skip-check-hash in workflows
(DYNAMOHUB-176)
- Snapdiff discovery task (DYNAMOHUB-245)
- Task to enumerate independent filesets on a site (DYNAMOHUB-251)
- AP-Analytics based search backend task (DYNAMOHUB-321)
- Accept configuration settings for search backend tasks
(DYNAMOHUB-329)
- Compatibility with ngenea 1.12 (DYNAMOHUB-333)
- Added argument "delete_remote_xattrs" to the move_paths task to
remove remote location metadata (DYNAMOHUB-346)
- Provide available metadata fields with search results
(DYNAMOHUB-382)
- Size-aware file list chunking in recursive_action (DYNAMOHUB-53)
- Track size of files being processed for reporting (DYNAMOHUB-63)


Bugfixes
--------

- Don't treat individual file failures as a task failure
(DYNAMOHUB-161)
- Don't treat ngenea warnings as errors (DYNAMOHUB-186)
- Disable task late ack to try to mitigate consumer timeouts
(DYNAMOHUB-291)
- Friendlier error response when Elasticsearch is unavailable
(DYNAMOHUB-364)


Ngenea Worker 0.6.0 (2021-03-26)
================================

Features
--------

- DYNAMO-99 - Dynamo now deletes provided empty folders
- DYNAMO-119 - Ngenea 1.9 character escaping support
- Updated EULA
- Rename dynamod service to ngenea-worker
- Support for use as an ngenea-hub worker agent

Bugfixes
--------

- DYNAMO-112 - Dynamo now uses unique names for each transient .acl
and .link file

Ngenea Worker 0.5.2 (2020-12-22)
================================

Feature
-------

- DYNAMO-111 - Ngenea 1.9 stdout results are now supported
```

```
Ngenea Worker 0.5.1 (2020-12-18)
================================

Bugfixes
--------

- DYNAMO-110 - AWS Credentials not correctly pass signature_version
correctly


Ngenea Worker 0.5.0 (2020-12-16)
================================

Features
--------

- DYNAMO-107 - AWS credentials can now specify endpoint_url, verify
and signature_version

Bugfixes
--------

- DYNAMO-108 - Dynamo now uses migration arguments when only the
source queue is specified

Ngenea Worker 0.4.0 (2020-12-11)
================================

- DYNAMO-94 - Ability to specify AWS credentials

Ngenea Worker 0.3.0 (2020-12-01)
================================

Features
--------

- DYNAMO-81 - Migrate-only workflow
- DYNAMO-92 - Ability to specify GCP credentials
- DYNAMO-93 - Delete-from-remote workflow


Ngenea Worker 0.2.0 (2020-11-09)
================================

- Refactored GA release


Ngenea Worker 0.1.0 (2020-04-06)
================================

- Initial Release
```

License

Ngenea Hub is licensed under the ArcaPix EULA: https://www.arcapix.com/licenses/EULA.txt

```
ArcaPix EULA
January 2021
http://www.arcapix.com/licenses/EULA.txt
Copyright 2021 ArcaPix Limited

This end user licence agreement (EULA) is a legal agreement between
you (the entity or individual who is using the software) and us (as
applicable either Pixit Media Limited, company number 07298805 or
ArcaStream Ltd, company number 08346283) in respect of (as
applicable):

* the computer software described in the order form, proof of
entitlement (POE), invoice or other document linking to this EULA
(in each case as issued by or agreed in writing with us), or which
otherwise incorporates or is governed by this EULA and the data
supplied with that software (collectively, Software);

* the application programming interface described in the order form,
proof of entitlement (POE), invoice or other document linking to
this EULA (in each case as issued by or agreed in writing with us),
or which otherwise incorporates or is governed by this EULA and the
data supplied with that application programming interface, in each
case whether provided on a standalone basis or alongside the
Software (collectively, API) and

* printed materials and electronic documents associated with that
Software or API (Documents).

The Software, API and Documents are collectively the "Work" and such
term includes each of the Software, API and Documents as applicable.

We license use of the Work to you on the basis of this EULA. We do
not sell the Work to you. We remain the owners of the Work at all
times. By using the Work, or otherwise by clicking "accept" or
otherwise indicating acceptance of this EULA, you confirm you accept
the terms of this EULA. If you do not accept this EULA you may not
use the Work.

Any services we provide, including but not limited to maintenance,
support, development and hosting, will be under separate terms of
business, but any software we provide to you incidentally in the
course of those services, will also be governed by this EULA unless
expressly stated that other licence terms will apply.

You should print a copy of this EULA for future reference.

1 Grant and scope of licence
```

1.1 In consideration of payment by you of the agreed licence fee and you agreeing to abide by the terms of this EULA, we grant to you a non-exclusive, non-transferable licence to use the Work on the terms of this EULA for the duration of your subscription. Your subscription will only be valid during the period for which you have a valid POE from us to use the Software and when your subscription expires this EULA will automatically terminate without the need for notice. Any termination of this EULA will also terminate your subscription and your POE will be invalidated.

1.2 You may download, install and use the Software for your own internal business purposes only on the systems, either physical or virtual detailed in the accompanying POE as identified in your purchase order (if applicable) or otherwise approved by us or our authorised representatives.

1.3 You may not use the Work for the purposes of making its functionality available to third parties as a service, whether directly or indirectly, without our express written agreement.

1.4 You may access and use the API solely for the purposes of

1.4.1 internally developing applications which communicate and interoperate with software or systems detailed in (and for the purposes detailed in) the accompanying POE as identified in your purchase order (if applicable) or otherwise approved by us or our authorised representatives; and

1.4.2 making calls to the systems or software permitted under clause 1.4.1, subject to any limits detailed in the accompanying POE as identified in your purchase order (if applicable) or otherwise agreed with us or our authorised representatives.

1.5 Your display and use information received through the API or data derived from that information is in each case subject to any limits detailed in the accompanying POE as identified in your purchase order (if applicable) or otherwise agreed with us or our authorised representatives).

1.6 You may use any Documents in support of the use permitted under condition 1.2 and make copies of the Documents as are reasonably necessary for their lawful use.

1.7 This EULA does not grant you permission to use the trade names, trademarks, service marks, or product names of us or our contributors or licensors, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of any "licence" files.

2 Restrictions

2.1 Except as expressly set out in this EULA or as permitted by any local law, you undertake:

2.1.1 not to copy the Work except where such copying is incidental to normal use of the Software, or where it is strictly necessary for the purpose of back-up or operational security;

2.1.2 not to rent, lease, sub-license, loan, translate, merge, adapt, vary or modify the Work;

2.1.3 not to make alterations to, or modifications of, the whole or any part of the Work, nor permit the Work or any part of them to be combined with, or become incorporated in, any other programs or other documentation as applicable, other than as expressly permitted in writing by us;

2.1.4 not to disassemble, decompile, reverse-engineer or create derivative works based on the whole or any part of the Software or API (except as expressly permitted by us in writing or clearly provided for within the functionality of the Software or any accompanying API we provide) nor attempt to do any such thing except to the extent that (by virtue of section 296A of the Copyright, Designs and Patents Act 1988) such actions cannot be prohibited because they are essential for the purpose of achieving inter-operability of the Software or API with another software program, and provided that the information obtained by you during such activities:

2.1.4.1 is used only for the purpose of achieving inter-operability of the Software or API with another software program; and

2.1.4.2 is not unnecessarily disclosed or communicated without our prior written consent to any third party; and

2.1.4.3 is not used to create any software which is substantially similar to the Software or API;

2.1.5 to keep all copies of the Work secure and to maintain accurate and up-to-date records of the number and locations of all copies of the Work;

2.1.6 to supervise and control use of the Work and ensure that the Work are only used by your employees (or such other individuals or entities are you may be expressly permitted in writing by us to allow to access or use the Work) in accordance with the terms of this EULA;

2.1.7 to include our copyright notice on and any "licence" text files in all entire and partial copies you make of the Work on any medium, however you may not use any component parts of the Work outside of or separately from the Work;

2.1.8 not to provide or otherwise make available the Work in whole or in part (including but not limited to program listings, object and source program listings, object code and source code), in any form to any person other than your employees without prior written consent from us; and

2.1.9 to comply with all applicable technology control or export laws and regulations.

2.2 Without prejudice to the restrictions in this EULA on copying, modifying or creating derivative works from the Work, where you (or someone on your behalf) creates (solely or in conjunction with others, and whether in object or source code form) any software or other work which is based on or derived from the Work (Derivative Work) in breach of this EULA or otherwise, then in consideration of the sum of 1 GBP (receipt and sufficiency of which you acknowledge), you hereby:

2.2.1 assign to us (by way of present assignment of future rights) all intellectual property rights in such Derivative Work and waive (and shall procure a waiver of) all moral rights arising under the Copyright, Designs and Patents Act 1988 in relation to the Derivative Work and, so far as is legally possible, any broadly equivalent rights that may exist in any territory of the world; and

2.2.2 In the event that any rights in such Derivative Work are not assigned to us pursuant to clause 2.2.1, you hereby grant to us an exclusive, royalty-free, worldwide, transferrable, irrevocable, perpetual licence (together with the right to grant sub-licences) to use in any manner as we determine, any such Derivative Work.

2.3 For the avoidance of doubt, for the purposes of clause 2.2, Derivative Work shall not include works which merely link or bind by name an existing third party application to the interfaces of the Software or API but does include works which are created to integrate with, or to be processed using, the interface of the Software or any API which we provide.

2.4 You agree not to (by your act or omission) do, or permit to be done, any act that will or may weaken, damage or be detrimental to the Work or any of our intellectual property rights or our or any of our contributors or licensors' rights in such, or seek to register any rights in the Work or any part of it or seek to commence litigation against any third party in respect of any intellectual property infringement in relation to the Work or any part of it.

3 Intellectual property rights

3.1 You acknowledge that all intellectual property rights in the Work anywhere in the world belong to us or our licensors or contributors, that rights in the Work are licensed (not sold) to you, and that you have no rights in, or to, the Work other than the right to use them in accordance with the terms of this EULA.

3.2 You acknowledge (unless explicitly agreed in writing by us) that you have no right to have access to the Software in source code form.

4 Liability

4.1 You acknowledge that the Work has not been developed to meet your individual requirements, including any particular cybersecurity requirements you might be subject to under law or otherwise, and that it is therefore your responsibility to ensure that the facilities and functions of the Software and API as described in the Documents meet your requirements.

4.2 We only supply the Work for internal use by your business, and you agree not to use the Work for any other purposes unless expressly permitted in writing by us.

4.3 We shall not in any circumstances whatever be liable to you, whether in contract, tort (including negligence), breach of statutory duty, or otherwise, arising under or in connection with this EULA for:

4.3.1 loss of profits, sales, business, or revenue;

4.3.2 business interruption;

4.3.3 loss of anticipated savings;

4.3.4 loss or corruption of data or information or any loss arising from misconfiguration or incorrect implementation or use of any API;

4.3.5 loss of business opportunity, goodwill or reputation;

where any of the losses set out in condition 4.3.1 to condition 4.3.5 are direct or indirect; or

4.3.6 any special, indirect or consequential loss, damage, charges or expenses.

4.4 Other than the losses set out in condition 4.3 (for which we are not liable), our maximum aggregate liability under or in connection with this EULA whether in contract, tort (including negligence) or otherwise, shall in all circumstances not exceed a sum equal to the Licence Fee paid in the 12 months prior to the event first giving rise to any liability. This maximum cap does not apply to condition 4.5.

4.5 Nothing in this EULA shall limit or exclude our liability for:

4.5.1 death or personal injury resulting from our negligence;

4.5.2 fraud or fraudulent misrepresentation;

4.5.3 any other liability that cannot be excluded or limited by English law.

4.6 Save as required by applicable law or agreed to in writing, we provide the Work on an "AS IS" basis, without conditions, warranties, representations or other terms of any kind, either express or implied (and any such implied conditions, warranties,

representations or other terms, whether implied by statute, common law or otherwise, are excluded to the fullest extent permitted by law), including, without limitation, any conditions, warranties, representations or other terms relating to title, non-infringement, merchantability, or fitness for a particular purpose. You are solely responsible for determining the appropriateness of using the Work and for any configuration or interface necessary for you to effectively use the Work and assume any risks associated with your exercise of permissions under this EULA.

4.7 Without prejudice to clause 4.6, where the API interacts with any software or system which is not provided by us, we are not responsible and shall have no liability in any way for such software or system.

5 Termination

5.1 We may terminate this EULA immediately by written notice to you if you commit a breach of this EULA which you fail to remedy (if remediable) within 14 days after the service of written notice requiring you to do so. Without prejudice to our rights under this clause 5.1, your rights under this EULA will terminate automatically without the need for notice if you commit a material breach of any of the terms of this EULA.

5.2 On termination for any reason:

5.2.1 all rights granted to you under this EULA shall cease;

5.2.2 you must immediately cease all activities authorised by this EULA; and

5.2.3 you must immediately and permanently delete or remove the Work from all computer equipment in your possession, and immediately destroy or return to us (at our option) all copies of the Work then in your possession, custody or control and, in the case of destruction, certify to us that you have done so.

6 Communications between us

6.1 We may update the terms of this EULA at any time on notice to you in accordance with this condition 6. Your continued use of the Work following the deemed receipt and service of the notice under condition 6.3 shall constitute your acceptance to the terms of this EULA, as varied. If you do not wish to accept the terms of the EULA (as varied) you must immediately stop using and accessing the Work on the deemed receipt and service of the notice.

6.2 If we have to contact you, we will do so by email or by pre-paid post to the address you provided in accordance with your order for or registration of the Work.

6.3 Note that any notice:

6.3.1 given by us to you will be deemed received and properly served 24 hours after it is first posted on our website, 24 hours after an email is sent, or three days after the date of posting of any letter; and

6.3.2 given by you to us will be deemed received and properly served 24 hours after an email is sent, or three days after the date of posting of any letter.

6.4 In proving the service of any notice, it will be sufficient to prove, in the case of posting on our website, that the website was generally accessible to the public for a period of 24 hours after the first posting of the notice; in the case of a letter, that such letter was properly addressed, stamped and placed in the post to the address of the recipient given for these purposes; and, in the case of an email, that such email was sent to the email address of the recipient given for these purposes.

7 Events outside our control

7.1 We will not be liable or responsible for any failure to perform, or delay in performance of, any of our obligations under this EULA that is caused by an Event Outside Our Control. An Event Outside Our Control is defined below in condition 7.2.

7.2 An Event Outside Our Control means any act or event beyond our reasonable control, including without limitation failure of public or private telecommunications networks.

7.3 If an Event Outside Our Control takes place that affects the performance of our obligations under this EULA:

7.3.1 our obligations under this EULA will be suspended and the time for performance of our obligations will be extended for the duration of the Event Outside Our Control; and

7.3.2 we will use our reasonable endeavours to find a solution by which our obligations under this EULA may be performed despite the Event Outside Our Control.

8 Third Party Software

8.1 Any part or component of the Software which has been contributed or created by any third party (including any open-source software) and which is not owned by us (Third Party Software) shall be deemed to be incorporated within the Software for the purposes of this EULA (except where expressly provided to the contrary) and use of the Third Party Software shall be subject to (and you shall comply with) such additional terms as relate to such Third Party Software from time to time (Third Party Additional Terms), and such Third Party Additional terms shall take precedence over this EULA in relation to such Third Party Software. You shall indemnify and hold us harmless against any loss or damage which we may suffer or incur as a result of your breach of any Third Party Additional Terms howsoever

arising, and we may treat your breach of any Third Party Additional Terms as a material breach of this EULA.

8.2 For the avoidance of doubt, the performance of, and any issues caused by or arising from, any Third Party Software shall be considered an Event Outside Our Control and (without prejudice to the provisions of this EULA in relation to warranties regarding the Software generally) all Third Party Software is provided on an "AS IS" basis and without conditions, warranties, representations or other terms of any kind, either express or implied (and any such implied conditions, warranties, representations or other terms, whether implied by statute, common law or otherwise, are excluded to the fullest extent permitted by law), including, without limitation, any conditions, warranties, representations or other terms relating to title, non-infringement, merchantability, or fitness for a particular purpose.

9 Other important terms

9.1 We may transfer our rights and obligations under this EULA to another organisation, but this will not affect your rights or our obligations under this EULA.

9.2 You may only transfer your rights or your obligations under this EULA to another person if we agree in writing.

9.3 This EULA and any document expressly referred to in it constitutes the entire agreement between us and supersedes and extinguishes all previous agreements, promises, assurances, warranties, representations and understandings between us, whether written or oral, relating to its subject matter. You agree that you shall have no remedies in respect of any statement, representation, assurance or warranty (whether made innocently or negligently) that is not set out in this EULA or any document expressly referred to in it. You agree that you shall have no claim for innocent or negligent misrepresentation or negligent misstatement based on any statement in this EULA or any document expressly referred to in it.

9.4 If we fail to insist that you perform any of your obligations under this EULA, or if we do not enforce our rights against you, or if we delay in doing so, that will not mean that we have waived our rights against you and will not mean that you do not have to comply with those obligations. If we do waive a default by you, we will only do so in writing signed by us, and that will not mean that we will automatically waive any later default by you.

9.5 Each of the conditions of this EULA operates separately. If any court or competent authority decides that any of them are unlawful or unenforceable, the remaining conditions will remain in full force and effect.

9.6 This EULA, its subject matter and its formation (and any non-contractual disputes or claims) are governed by English law. We both

```
irrevocably agree to the exclusive jurisdiction of the courts of
England and Wales.
```