Ngenea Hub

Ngenea Hub is a centralized control system designed to manage the movement of data across diverse storage environments. It simplifies the synchronization of files, migration to more efficient storage, and ensures data is moved according to business needs.

Features

- **Web-Based Interface**: A user-friendly dashboard that allows easy management of data across different storage systems, monitors synchronization, and sets migration policies without requiring technical expertise.
- **CLI and API**: Command-line tools and APIs provide advanced users the flexibility to automate tasks and integrate them into larger systems, offering greater control and customization of workflows.
- **Pixstor Integration**: Enables users to create Spaces for datasets across multiple Pixstor environments, managing synchronization, data manipulation through workflows, and system settings from a unified interface.

Uses

- **Data Synchronization**: Effortlessly synchronize files across multiple storage environments, ensuring data consistency and accessibility.
- **Data Migration**: Migrate data to more cost-effective storage resources, including cloud, object storage, NAS, and tape resources, based on usage and value.
- **Business-Driven Data Movement**: Automatically move data into the most suitable storage resource, ensuring efficient allocation and reducing operational costs based on business needs.
- **Centralized Control**: Streamline system settings and manage all Pixstor environments from one administrative interface, enhancing operational efficiency and control.

Ngenea Hub 2

Ngenea Hub 2 expands on the original Hub by giving you even more control, especially if you’re managing multiple Pixstor systems. It helps you create “Spaces”—logical groupings of datasets—so you can organize your data more effectively. With Ngenea Hub 2, you can synchronize data across different locations and automatically migrate it to the most cost-efficient storage, ensuring that your storage solution always aligns with your business goals.

What is the Role of the Ngenea Worker?

The Ngenea Worker is the engine that orchestrates the actual movement and synchronization of data behind the scenes. It carries out tasks like migrating data

between storage tiers and ensuring that files are synchronized between Pixstor and other storage systems, including cloud platforms. By automating these tasks, Ngenea Worker saves time and ensures that your data is always in the right place, ready for when you need it.

How can I Interact with Ngenea Hub and Manage Data Easily?

You can use the **Web UI** or **ngclient**, a command-line tool designed to make managing data migrations simple and efficient. With either, you can:

- **Migrate**: Move data between storage locations, such as from your local system to cloud storage.
- **Recall**: Retrieve data from a lower-cost storage tier back to high-speed storage when needed.
- **Send**: Transfer data to other endpoints like cloud platforms or long-term archives.
- **Send**: Transfer data to other Hub sites.

To keep everything secure, token-based authentication is used, ensuring that only authorized users can execute data management tasks.

What does Ngenea Hub Orchestrate?

Ngenea HSM

Ngenea HSM (Hierarchical Storage Management) is an important feature that helps you manage where your data is stored. Not all data is the same—some is accessed frequently, and some is rarely used. Ngenea HSM moves less critical data to more cost-effective storage options, like cloud storage or archives, freeing up your high-performance storage for the data you need most.

It integrates with popular cloud storage platforms like Amazon S3, Google Cloud, and Microsoft Azure, as well as on-premises systems, so you can easily manage your data across different environments.

Pixstor

Pixstor is a powerful and flexible storage platform designed to manage large amounts of data in high-demand environments. It offers fast, reliable, and scalable storage that adapts to your needs, whether you’re working on-premises, in the cloud, or both. With Pixstor, you get a comprehensive solution to organize and control your data effectively.

How does Pixstor work?

Pixstor is built on two key components:

- **Infrastructure Management with Saltstack**: Saltstack is an advanced tool that automates and simplifies complex tasks like setting up servers, updating

software, and scaling resources. It helps keep your storage environment running smoothly and efficiently, without requiring much manual effort.

- **Configuration CLI (Command Line Interface)**: The Pixstor CLI is a tool that allows administrators to manage everything from setting up storage pools to controlling access and monitoring the health of the system. It also supports automation, so routine tasks can be integrated into your workflows easily. This means you can manage your storage with minimal hassle and greater precision.

Ngenea Hub Architecture Configurations

Ngenea Hub can be configured in different architectural setups, depending on the infrastructure and deployment environment. Each setup influences how Ngenea Hub and Workers interact with Pixstor nodes and other system components.

Pixstor-Centric Architecture: In this setup, the Ngenea Hub is hosted on a Pixstor Management Node. However, for most environments, the Worker is typically run on separate Ngenea nodes, rather than on the management node itself. This ensures better scalability and performance.

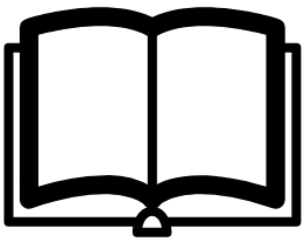
This configuration is ideal for environments where Pixstor serves as the primary storage infrastructure, and high-performance data access is required directly from the storage nodes.

Independent Hub Architecture: In this architecture, the Ngenea Hub is installed on a separate node running Enterprise Linux 8/9 (EL8/EL9), independent of the Pixstor storage system. Workers are deployed exclusively on the Pixstor nodes to handle the actual data processing tasks, allowing for separation between control and execution layers.

This configuration is suitable for organizations preferring to decouple the management layer from the processing layer, especially in cloud or hybrid environments.

Pick your area of interest:

Please refer to the provided links below, which will direct you to both the Administration and User Guides for Pixstor-Centric Architecture.



User Guide

Administration Guide

Release Notes

2.11.0

Warning: Ngenea Hub has been upgraded to Python 3.12. As a result of this internal change all plugins must upgrade their celery dependencies and be re-installed, more details for this can be found in the plugin dependency page.

What you need to know

- Ngenea Hub has been internally upgraded with newest security fixes.
- Ngenea Hub now runs on Python 3.12.
- DAG and SnapDiff workflows now support configurable `pre_action` and `post_action` tasks executed before and after workflow processing.
- `pre_action` and `post_action` definitions can be overridden at runtime through the workflow API.
- Schedules now support defining `pre_action` and `post_action` directly on subscribed workflows.
- Workflow-level actions are ignored when the workflow is scheduled. Pre/post actions must be defined directly on the schedule.
- SnapDiff workflows now support streamed input handling for `post_action` tasks by default.
- Introduced a standard `templated_email` task for workflow notifications, with reusable email templates and plugin integration.
- Alerts now support maintenance mode. When maintenance mode is enabled for a site, all alerts for that site are muted. Maintenance mode automatically expires after a configured timeout.
- The policy creation wizard has received an overhaul with extra supported execution options
- A new “Iris Metadata Policy” schedule type has been introduced to migrate and recall files based on file metadata in Iris.
- Site-level excludes can now be manipulated directly from the GUI.

Changes that may affect tools

- Ngenea Hub plugins must be upgraded and re-installed to match new dependencies, details on how to perform that can be found in the plugin dependency page.

What’s Fixed

- Consult the changelog for individual fixes

2.10.0

Note: The internal version of the PostgreSQL database has been upgraded from version 13 to 17. Currently existing data will be automatically translated to the new

database version upon first boot after a PixStor upgrade. Logs for this process can be retrieved from the `ngeneahub-dbmigration` systemd service.

What you need to know

- Ngenea Hub now backs up database data and configuration on startup
- It is now possible to include job history in database backups. Note that including job history will increase the size of the backup file and will take longer to complete. Details for this are in the backup configuration settings.
- New Job Alerting functionality can send email alerts if Jobs error or are not yet finished after an expected time period
- Ngenea Hub is now prepared for failover by allowing multiple Ngenea Hub instances with shared high-availability (HA) services such as PostgreSQL, Redis and RabbitMQ. See the Ngenea Hub configuration guide for further information.
- All share types can now be configured per-Site on each Space

Changes that may affect tools

- The `/health` endpoint now reports on infrastructure component health The previous Sites health response is now available at `/health/sites`
- Docker service names change to a new format, E.G. `ngeneahub_backend_1`, is now `ngeneahub-backend-1`
- The output format of `ngeneahubctl status` now only reports service status and uptime The “command” and “port” columns have been removed. Use `docker ps` to retrieve detailed service information.
- Ngenea Hub provides an updated Docker version. The updated Docker version is incompatible with `DJANGO_SECRETS` containing a `$` or `#` character. Hub automatically converts the Django secrets where such characters are present. Upon conversion existing JWT tokens are invalidated. Hub native API keys are not affected.

What’s Fixed

- Consult the changelog for individual fixes

2.9.4

What you need to know

Resolves an indentified vulnerability in Ngenea Hub versions 2.7.0 through 2.9.3 inclusive that could allow a low-privileged user to elevate their privileges by changing their group membership or performing user management via remote API requests.

What’s Fixed

- Consult the changelog for individual fixes

2.9.3

What you need to know

- Adds a default workflow for manual Iris notification. Import of a custom workflow is no longer required.
- Introduces various quality of life improvements to the Hub UI. Refer to the changelog for individual improvements.
- Ngenea HSM 1.33 support
- Cloud provider-specific tasks will be deprecated in a future release. The following tasks are affected:
    - `delete_from_s3`
    - `remove_from_s3_bucket`
    - `copy_between_s3_buckets`
    - `remove_from_gcs_bucket`
    - `copy_between_gcs_buckets`

What’s Fixed

- Consult the changelog for individual fixes

2.9.2

What you need to know

- Adds support for triggering Iris file notification manually. This requires the import of a custom workflow. Contact support for more details.

What’s Fixed

- This release includes no bug fixes.

2.9.1

What you need to know

- Resolves an issue whereby upgrades from versions prior to 2.8.0 causes pre-upgrade schedules running workflows with snapdiff, such as sync, to reprocess all data. New schedules post-upgrade are unaffected.

What’s Fixed

- Consult the changelog for individual fixes

2.9.0

Whats new

- Iris enablement of spaces
- Iris Single Sign On (SSO)
- External service support for Postgres, RabbitMQ and Redis
- Ngenea HSM 1.32 support

What you need to know

- The configuration value LDAP_MIRROR_GROUPS is no longer json loaded, please consult the LDAP documentation for more details.
- The configuration value “celery_broker” will be automatically replaced with BROKER_PROVIDER and RESULTS_PROVIDER on first hub startup.
- Any configuration file older than version 2.9.0 will be backed up with this upgrade
- SSO and LDAP cannot be enabled simultaneously, please consult the LDAP and Single Sign-On documentation for more details.
- SSO is not enabled by default and must be configured and enabled, details for this process are in the Single Sign-On documentation.

What’s Fixed

- Consult the changelog for individual fixes

2.8.3

What you need to know

Resolves an indentified vulnerability in Ngenea Hub versions 2.7.0 through 2.9.3 inclusive that could allow a low-privileged user to elevate their privileges by changing their group membership or performing user management via remote API requests.

What’s Fixed

- Consult the changelog for individual fixes

2.8.2

What you need to know

- Resolves an issue whereby upgrades from versions prior to 2.8.0 causes pre-upgrade schedules running workflows with snapdiff, such as sync, to reprocess all data. New schedules post-upgrade are unaffected.
- Support all ngenea configuration paramaters for HSM External Targets

2.8.1

What you need to know

- Ngenea targets are no longer renamed on update, the new format is only applied to new targets
- Ngenea HSM 1.32 support has been added to targets

What’s Fixed

- Consult the Changelog for individual fixes

2.8.0

What’s new

- Iris beta (feature flagged)
- Initial release of Programming Guide, with improved, user-friendly REST API documentation

What you need to know

- API only workflows can now be selected when editing the Space Schedules and Schedule settings
- Hub containers now auto-restart when encountering an issue which causes container shutdown

What’s Fixed

- Consult the Changelog for individual fixes

2.7.1

What you need to know

- Ngenea targets are no longer renamed on update, the new format is only applied to new targets

What’s Fixed

- Consult the Changelog for individual fixes

2.7.0

What’s new

- Spaces can now be viewed in table layout
- Schedules are now specified relative to the site timezones rather than UTC+0
- Site workflow batch size options to control workflow task processing performance
- Policies support more file systems than mmfs1
- Global settings now provides Timeout controls for Hub behaviours
- API performance can be scaled with the new GUNICORN_THREADS setting

What you need to know

- Where NAS user(s) and group(s) exist prior to additional sites being joined, automatic creation of existing NAS users and groups as part of setup on the new Site may fail if the Site is still undergoing joining. Should this occur, re-submit the failed ‘Creating NAS group’ and/or ‘Creating NAS user’ job after a few minutes.

What’s Fixed

- Consult the Changelog for individual fixes

User Guide

Overview

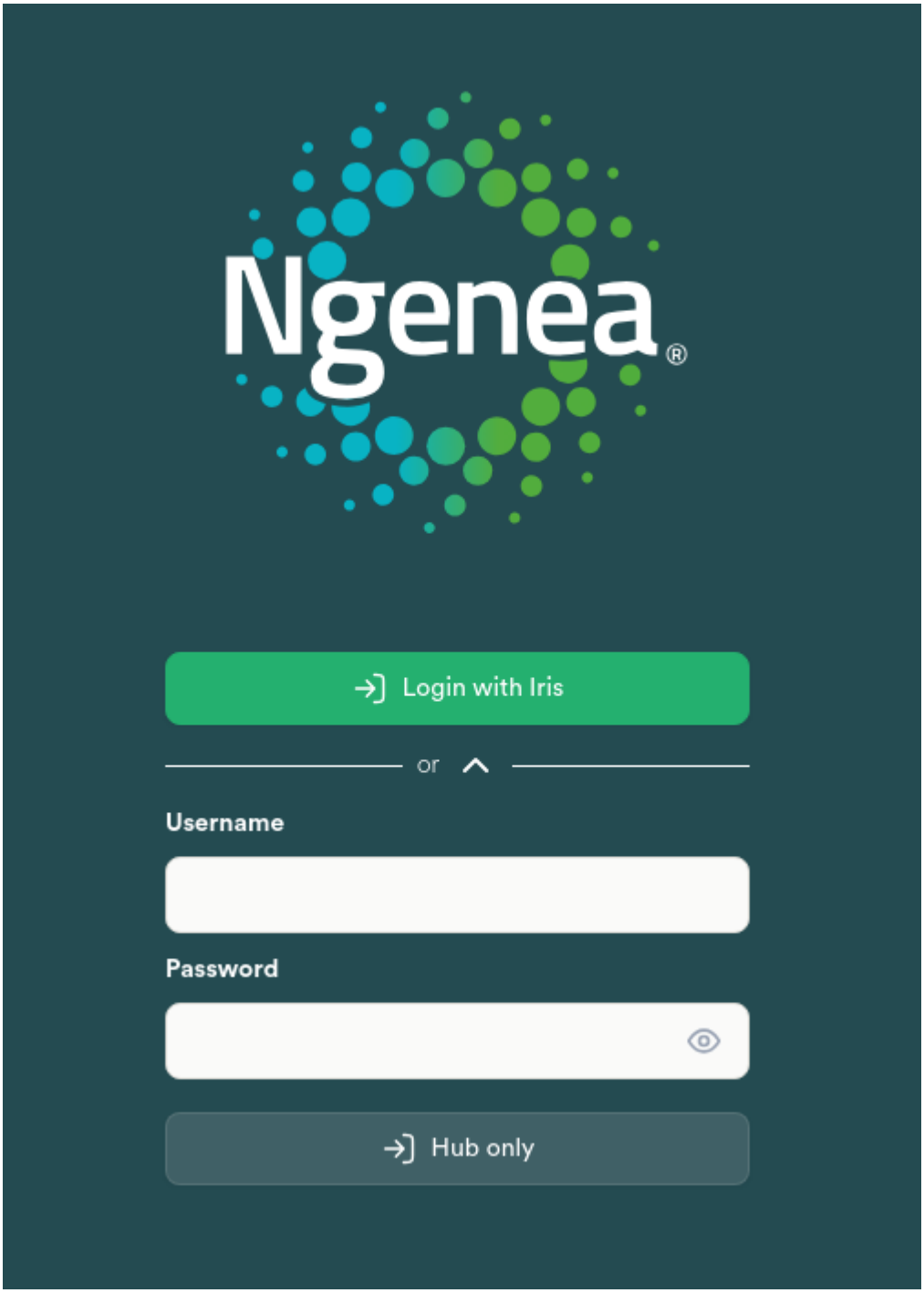
Ngenea Hub provides a management and control interface for data and systems across a global pixstor estate.

Using Ngenea Hub, users can create Spaces for datasets across multiple pixstors, enable data synchronisation, perform data manipulation through workflows and control system settings for all pixstors from one administrative interface.

Accessing Hub

To login to Hub enter the URL of the hub into the browser address bar. E.G. https://myhub

On successful connection to the Hub a login page is presented.



Where Ngenea Hub has been deployed with the wider Iris product suite an option to choose to login with Iris **Single Sign On** is presented. Choose this option if you are also using Iris products.

Alternatively, choose **Hub only** login to only authenticate locally to the Hub.

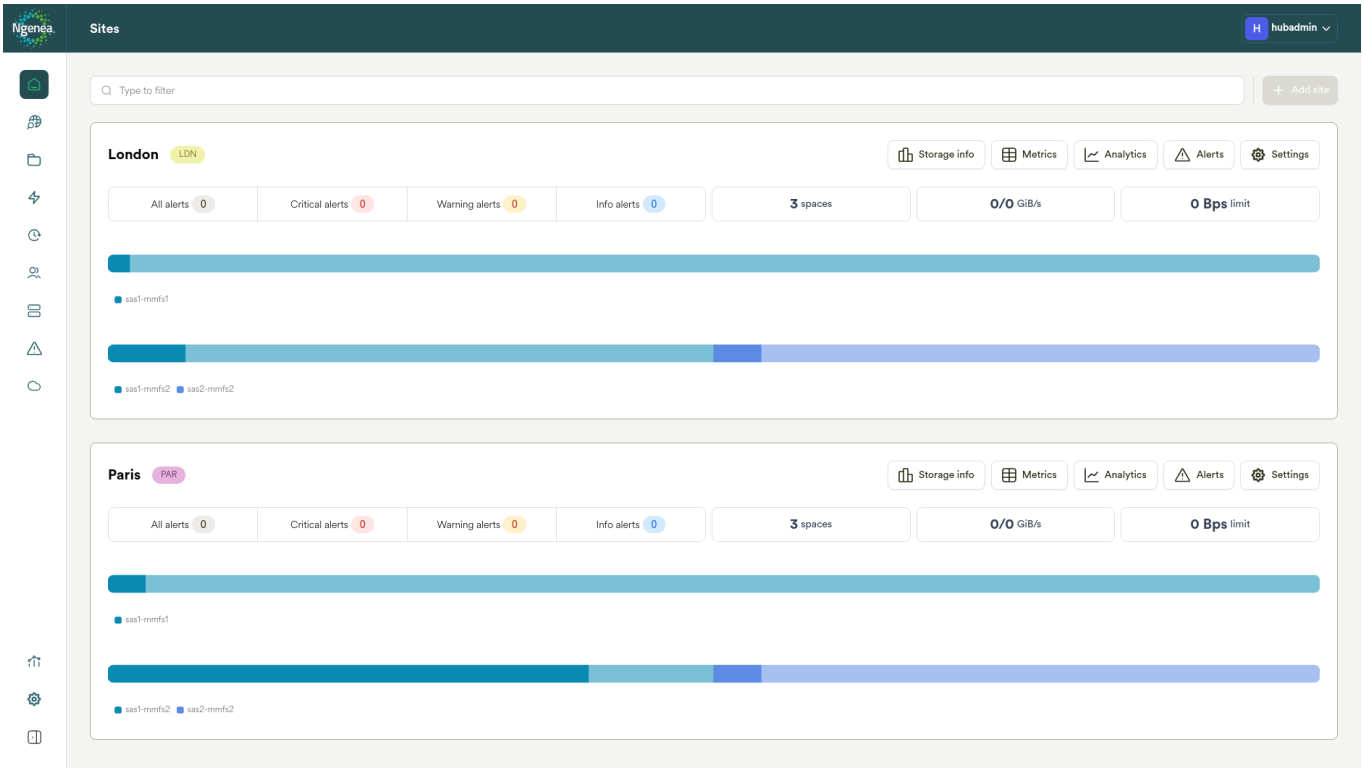
- Choose the appropriate option, enter your Username and Password to authenticate and login

Logging Out Of Hub

After successful login, from the User Profile control in the top right of the screen select the Log out option.

The Main Screen

The main Hub screen is comprised of several areas.



Menu Bar

The menu bar is positioned on the far left of the Hub screen.

Menu Buttons

The menu buttons perform the following functions when clicked:

Home

	Navigates to the page first presented after Login

Spaces

	Navigates to the Spaces screen

Jobs

	Navigates to the Jobs screen

Groups & Users

	Navigates to the Groups & Users screen

Sites

	Navigates to the Sites screen

Alerts

	Navigates to the Global Alerts screen

Targets

	Navigates to the Targets screen

Hub Metrics

	Navigates to the External Hub Grafana Metrics. Only visible to Hub Administrators when Hub is operating in `BROKER_PROVIDER=redis` mode. Typically used to see service availablity and queue task length.

Global Settings

	Navigates to the Global Settings screen

Enlarge/Reduce

	Clicking the toggle widens the menu bar to show:	• Menu button descriptions	• Hub version

Clicking the toggle button again shrinks the menu bar.

Page Location

The page location is positioned to the top of the Hub screen.

Main page content

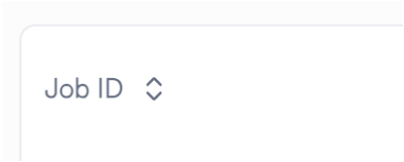
The main page content is positioned to the center of the Hub screen.

Tables

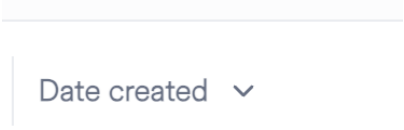
Most tables offer the following common functions:

Sorting Values In a Column

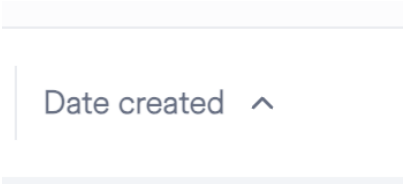
When it is possible to sort the values in a column, a control button is provided for you near the name of the column:



Indicates that the values in this column are sortable

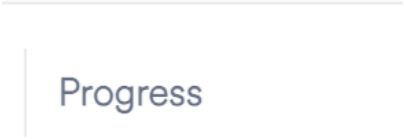


Indicates that the values in this column are sorted in descending order



Indicates that the values in this column are sorted in ascending order

If the values in a column are not sortable, then no control button is provided:



Indicates that the values in this column are not sortable

Pagination

Tables do not show all of the items. The User interface offers controls to see more results when these controls are used.

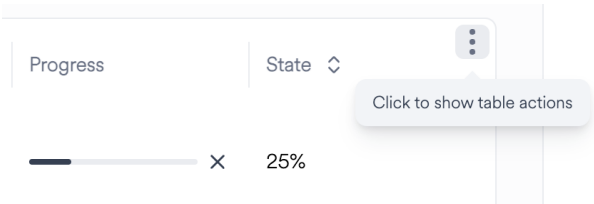
The users are able to view 20, 50 or 100 items on the current page.

The users can change the current page by clicking to a page number.

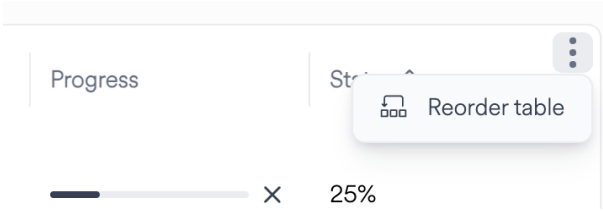
The users are able to switch to previous or next page.

Reordering Columns

Table columns can be reordered:



The users are able to open the table controls menu by clicking this



The users are able to open the table controls menu by clicking this

The table field organizer is presented for table customization:

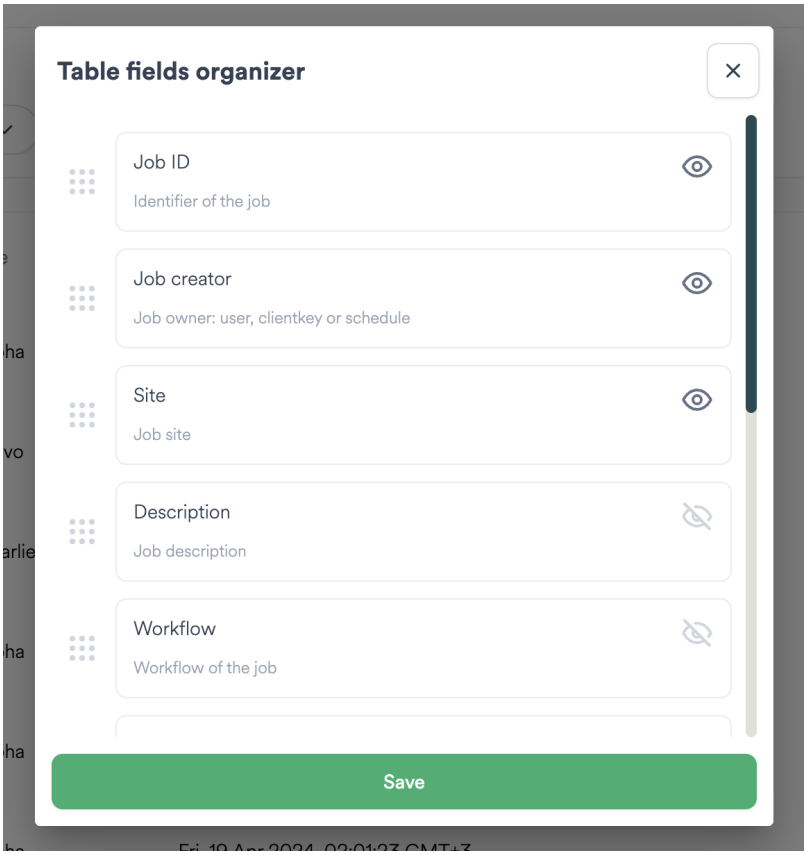


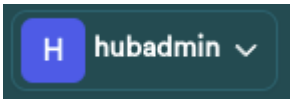
Table fields can be organized using the following controls:

	By clicking and dragging this button, users are able to change the appearance order of the columns.
Job ID Identifier of the job	When a table column is visible, it is represented as open eye icon and clicking it will hide the column.
Description Job description	When a table column is hidden, it is represented as closed eye icon and clicking it will show the column.

User Profile

The User Profile control is positioned to the top right of the Hub screen.

The User Profile control displays the name of the user logged into Hub using the current browser session.

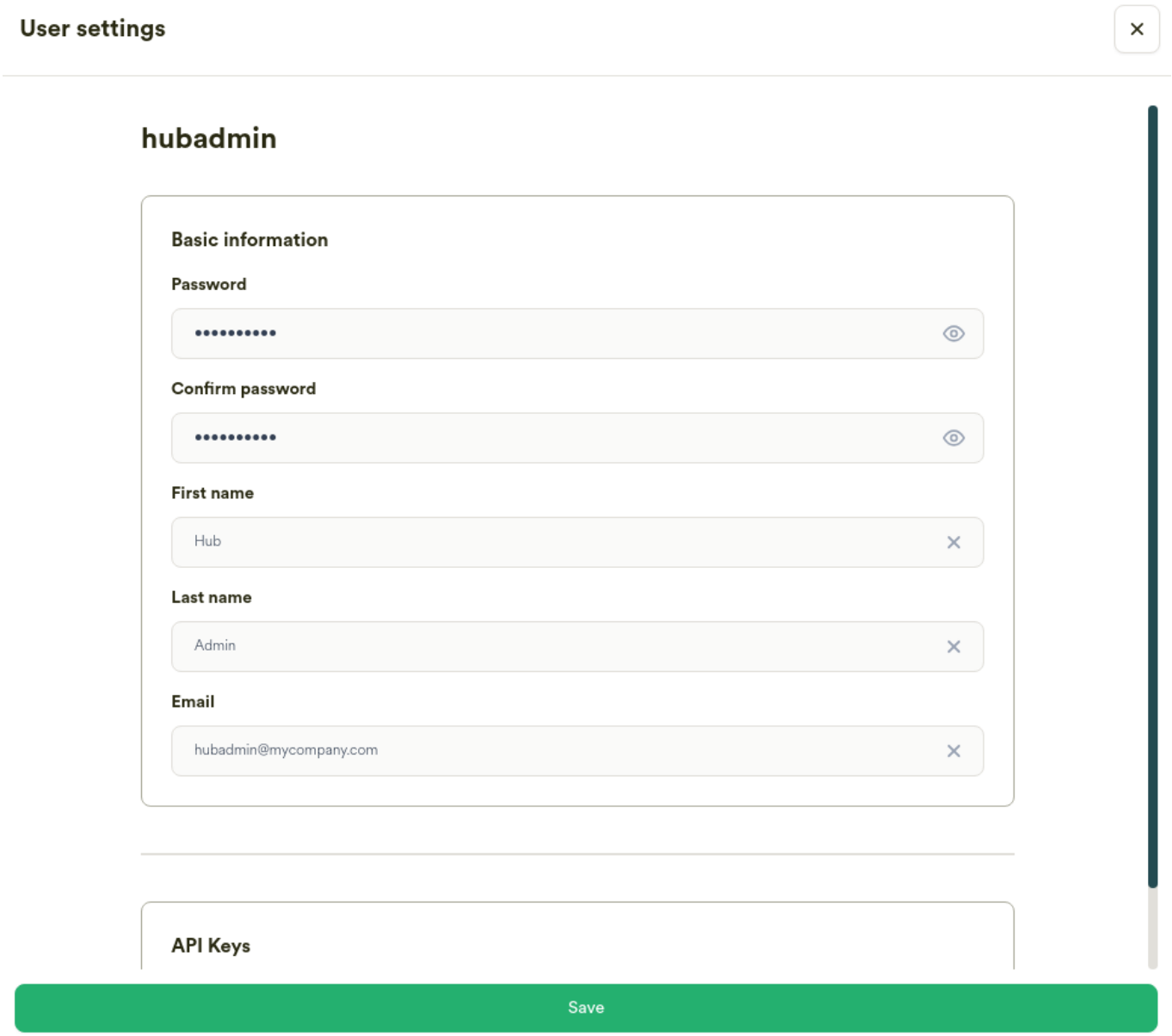


Use this control to Log out of Hub and update your User settings.

User Profile

Selecting the Profile option from the drop down menu presents the User settings dialog of the logged in user.

Clicking the Save button at the bottom of the User settings dialog saves any changes made.



Tip: Changing your password will automatically log you out from the Hub UI when the profile is saved.

Password

Enter a new password to change your password, satisfying the password rules.

Confirm Password

Enter the new password identically.

First name

Enter a new first name to change your First name.

Last name

Enter a new last name to change your Last name.

Email

Enter a new valid email address to change your email address.

Groups

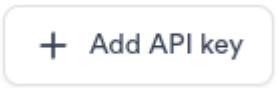
A list of groups your user account belongs to is displayed.

API Keys

Provides the capability to add and remove API keys.

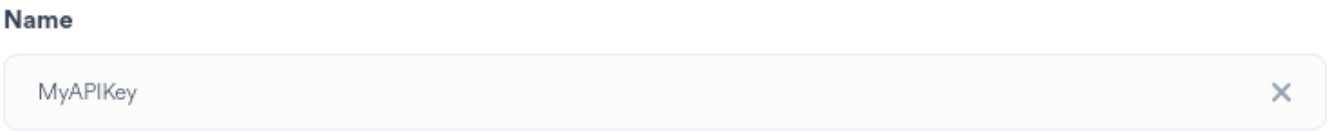
An API key is typically used for by 3rd party software to connect to Hub to perform automation.

Adding an API Key



To add a new API key, click the Add API key button.

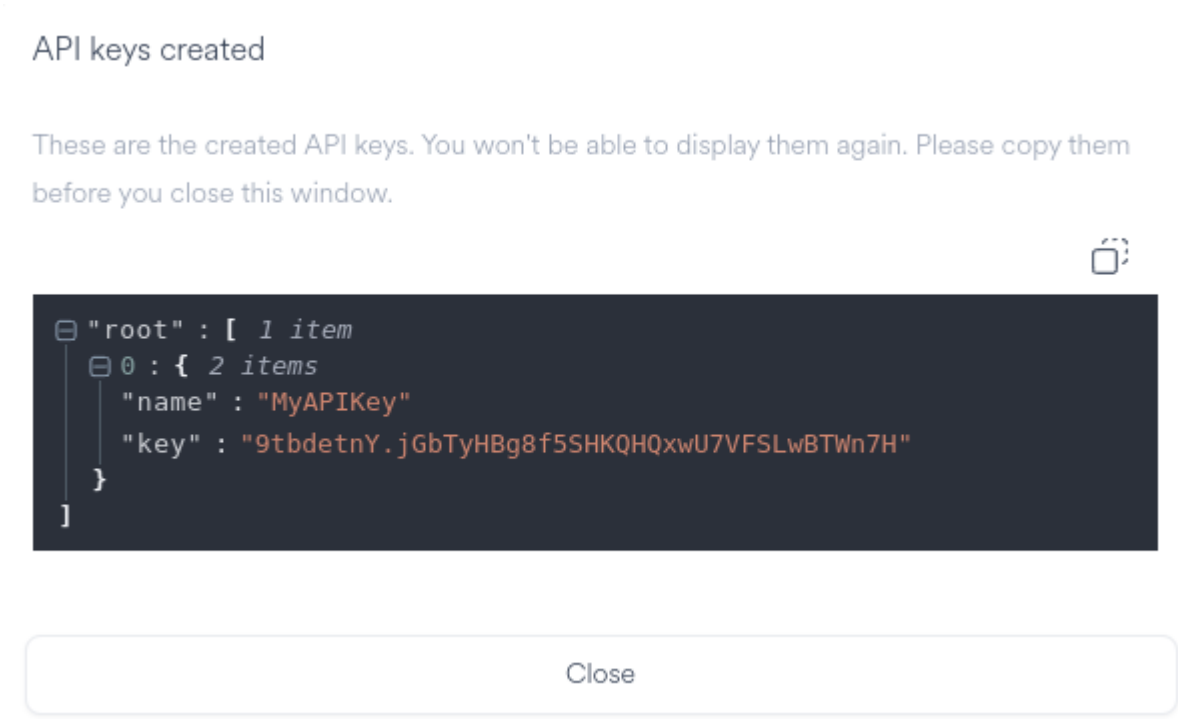
Enter a name for the API Key .



Tip: It is not possible to change the name of an API key once the key has been created.

Add additional API keys as required.

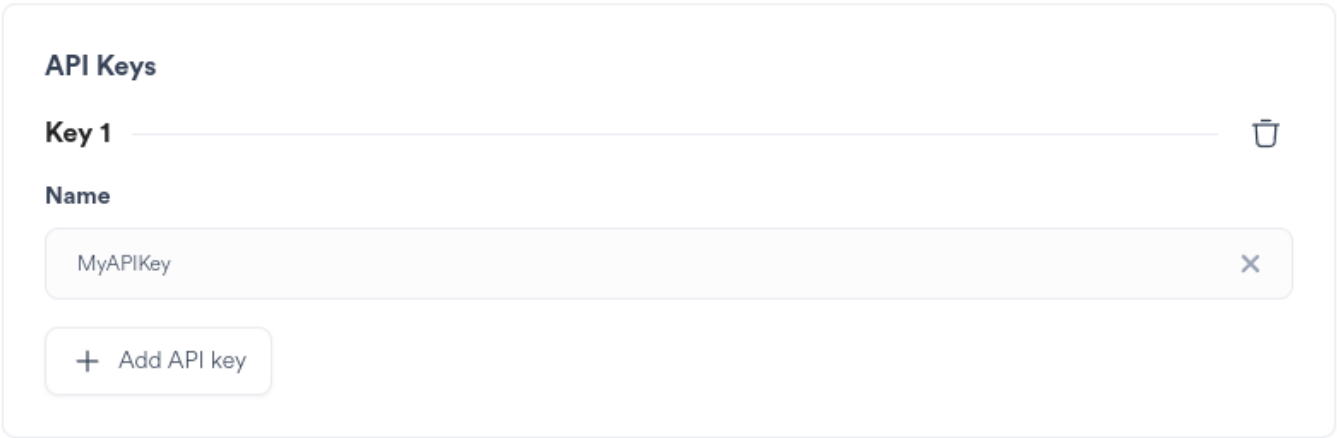
Upon pressing Save the API Key dialog is raised displaying the API Keys created.



Ensure to save the generated keys safely as once the API Key dialog is closed the keys cannot be viewed again.

Removing an API key

Created API Keys are listed



To remove an API key, click the dustbin icon next to the API Key.

Tip: Deleted API Keys are non-recoverable. If an API Key has been inadvertently removed, do not press the Save button, instead click off the User settings dialog to the main area of the screen.

Clicking the Save button at the bottom of the User settings dialog saves any changes made.

Search

Hub provides the ability to Search for assets across all Hub managed Sites.

Where proxies are enabled for Search supported asset types, thumbnails and proxies are displayed.

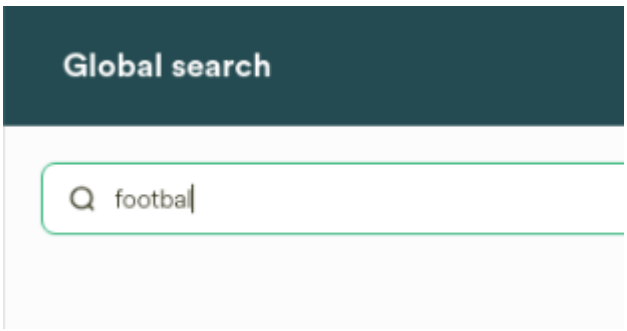
Asset locations, Spaces, metadata and Tags are searchable providing rich criteria to locate assets.

Located assets can be processed using the Jobs Panel to perform Workflow actions.

Searching

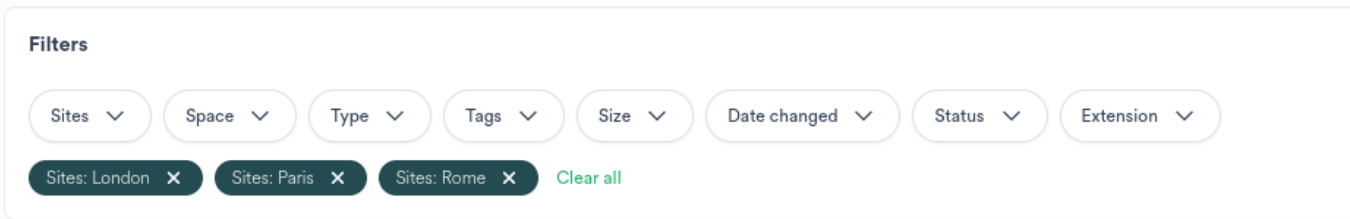
Enter a term in the Search bar to start a Search. E.G.: *football*.

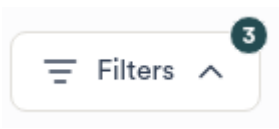
Terms are automatically wildcarded to enable wide Searching. E.G.: **football**.



Filters

Search provides several high level filters each of which apply additional criteria to narrow the Search to return only the required assets.





Click the filter button to display the available filters. The number of applied filters is displayed on the filter button.

Sidebar

Clicking the toggle displays the side bar to show:

- Item Info
- Tags
- Metadata

Clicking the toggle button again hides the side bar.

Item info —

Name	football hd001.png
Path	/mmfs1/data/datasync/footbal...
Size	3.2 MB

Tags —

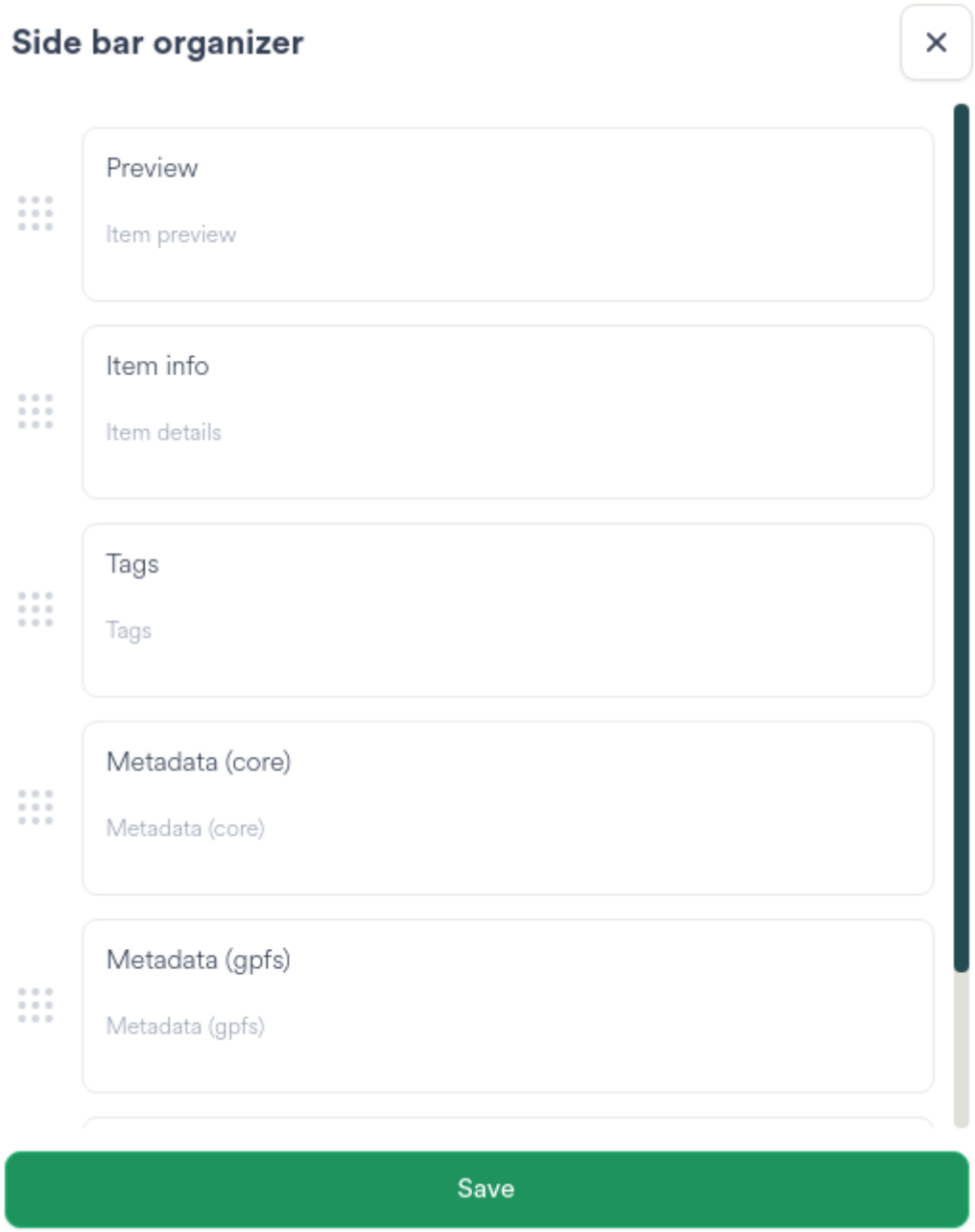
Empty (add tags)

Metadata (core) —

accesstime	24.01.2024, 10:25 GMT
blocksize	8388608
changetime	24.01.2024, 10:25 GMT
deviceid	250609814
directory	/mmfs1/data/datasync/footbal...

Reordering

Re-arrange items in top to bottom order by dragging vertically up or down.



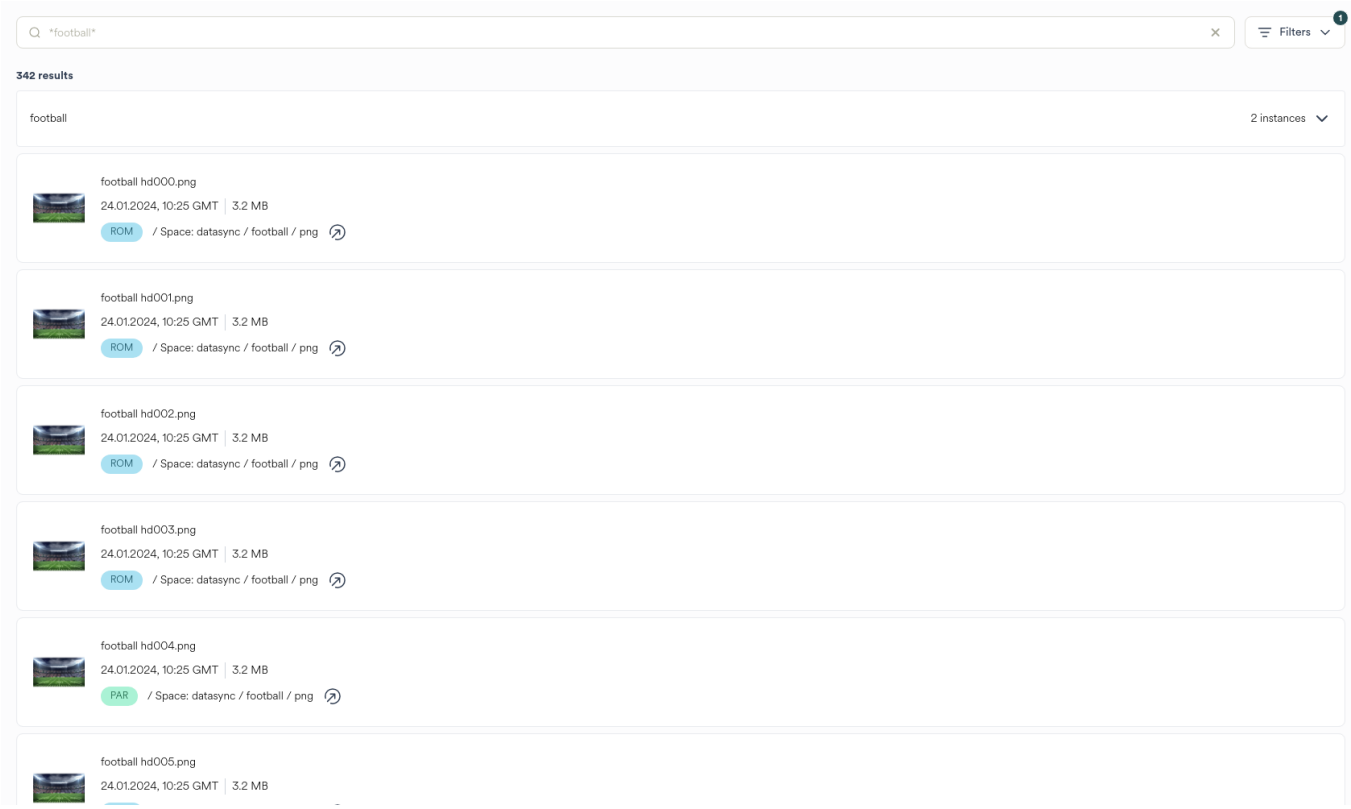
Click Save to store the sequence which is reflected in the Job side bar.

Results

Search matches assets to the Search input returning collated results to screen.

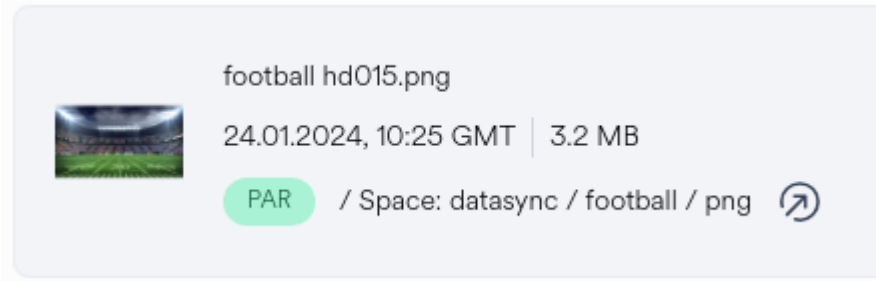
Searches are processed asynchronously, therefore some results may take longer to return to screen than others.

Where a Site is unavailable or takes too long to return results, information messages appear at the top of the screen allowing for deselection of the Site or Site(s).



Result Card

Returned assets appear as a Result Card.



Where multiple instances of the same asset are present on more than one Site, the results are collated.

2 instances	Open the instances drop down to display all the assets from all Sites

The Result Card displays:

Item	Description
Thumbnail	The generated thumbnail or a default icon where no thumbnail exists
Name	The name of the asset
Date and Size	The date and time of last modification, the size of the file on the pixstor file system
Site chip	The site on which the asset is found

Item	Description
Space Location	The location of the asset in the Space
	The filebrowser navigation button

Tags

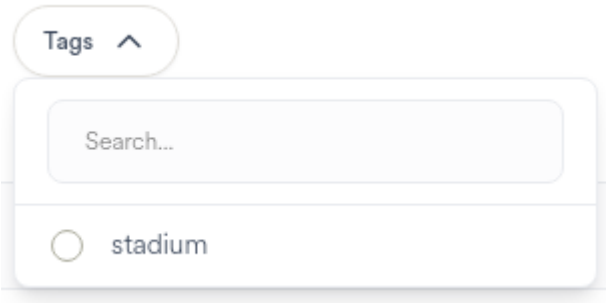
Search provides the ability to Tag assets.

Apply Search Tags to assets in order to collate assets into collections or curate as required.

Tip: Applied Tags are added to Site Search services asynchronously. Dependent on workload there may be a small delay between the addition of the Tag in the UI and the ability to Search for the asset by the Tag.

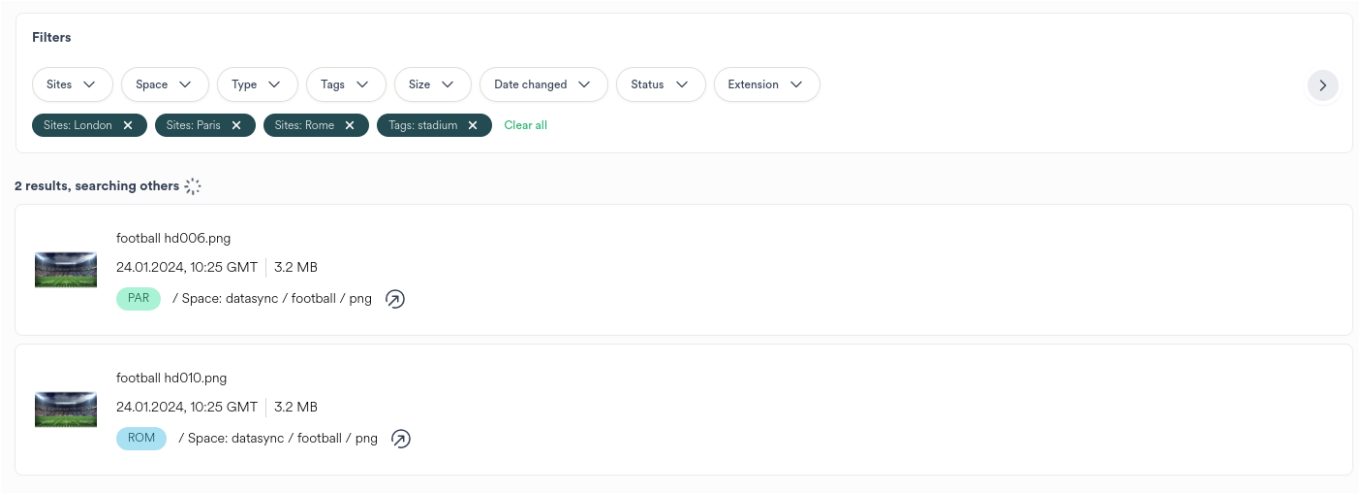
Searching By Tag

- To Search for an asset by Tag, open the Search filter bar and select the Tag from the drop down menu
- Entering text into the Tag filter matches available Tags to provide a shorter selection list



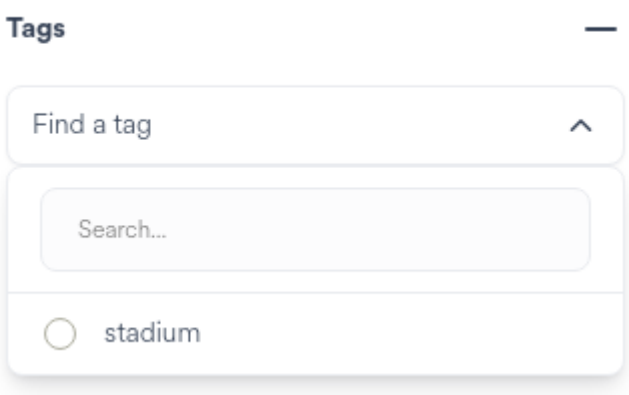
Applied Tags are denoted with a chip below the filter bar. Click the *x* to remove the Tag from the filter.

Example of Searching for assets where the Tag *stadium* is applied:

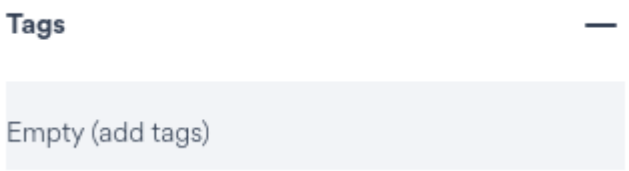


Tagging An Asset

To add a Tag to an asset, open the side bar and locate the Tags section in the bar.

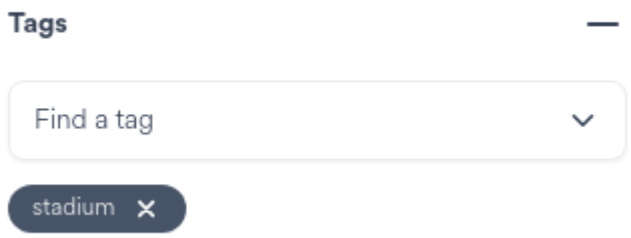


Where no Tags are set on the asset, a prompt to add Tags is shown. Click the entry box to start.

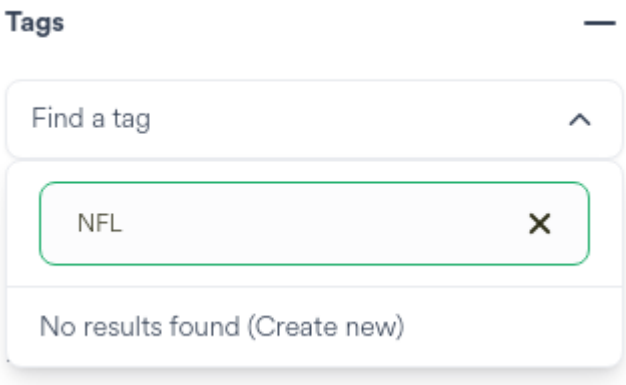


To apply an existing Tag to the asset, select the Tag.

Applied Tags are denoted with a chip below the filter bar. Click the *x* to remove the Tag from the filter.



If no Tags exist, enter a Tag (case sensitive) in the Tag entry box and press Enter to create and apply the new Tag.



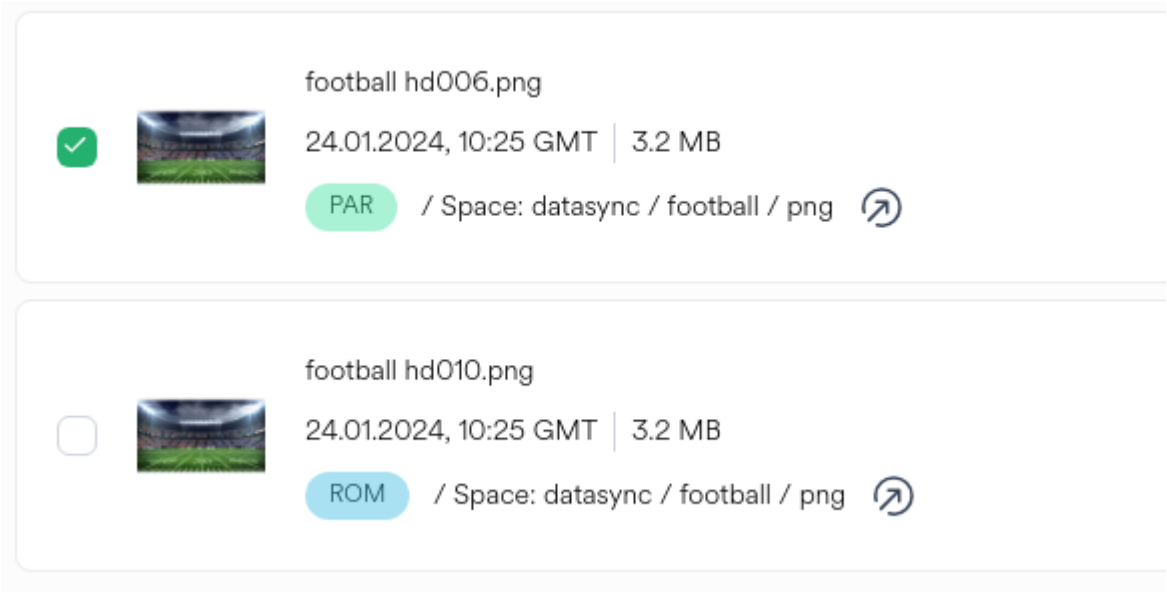
Actions

Post Search, assets can be actioned by Workflows using the Job Panel.

To enable asset selection first select an individual Space from the Search filters.

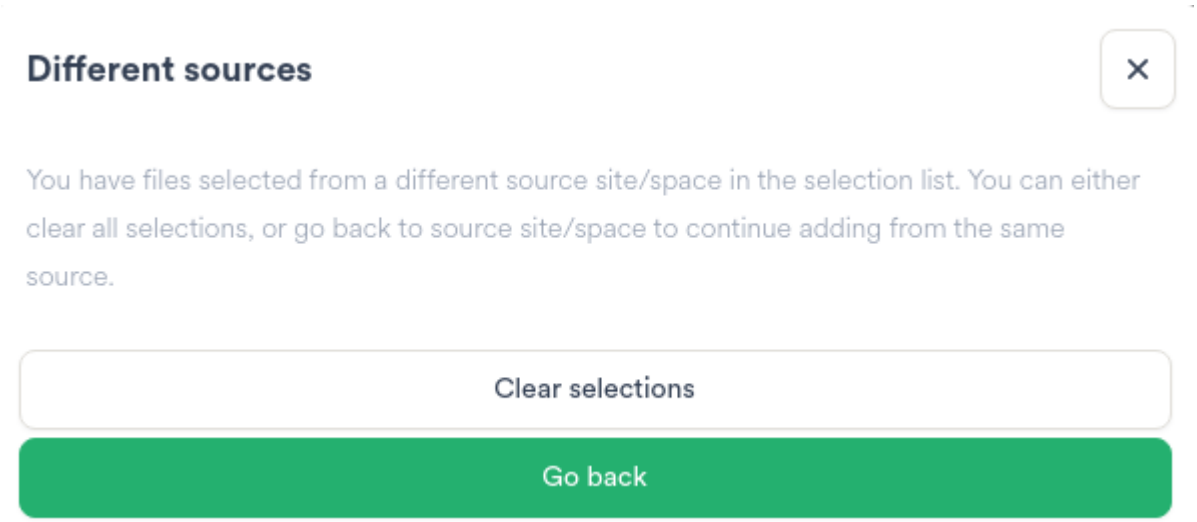
Selecting an individual Space enables asset selection.

- Click the checkbox to the left of an asset to add the asset to the Job Panel selection list



Assets in the Job Panel are processed per Space, per Site. Selecting assets from additional Sites raises a confirmation dialog.

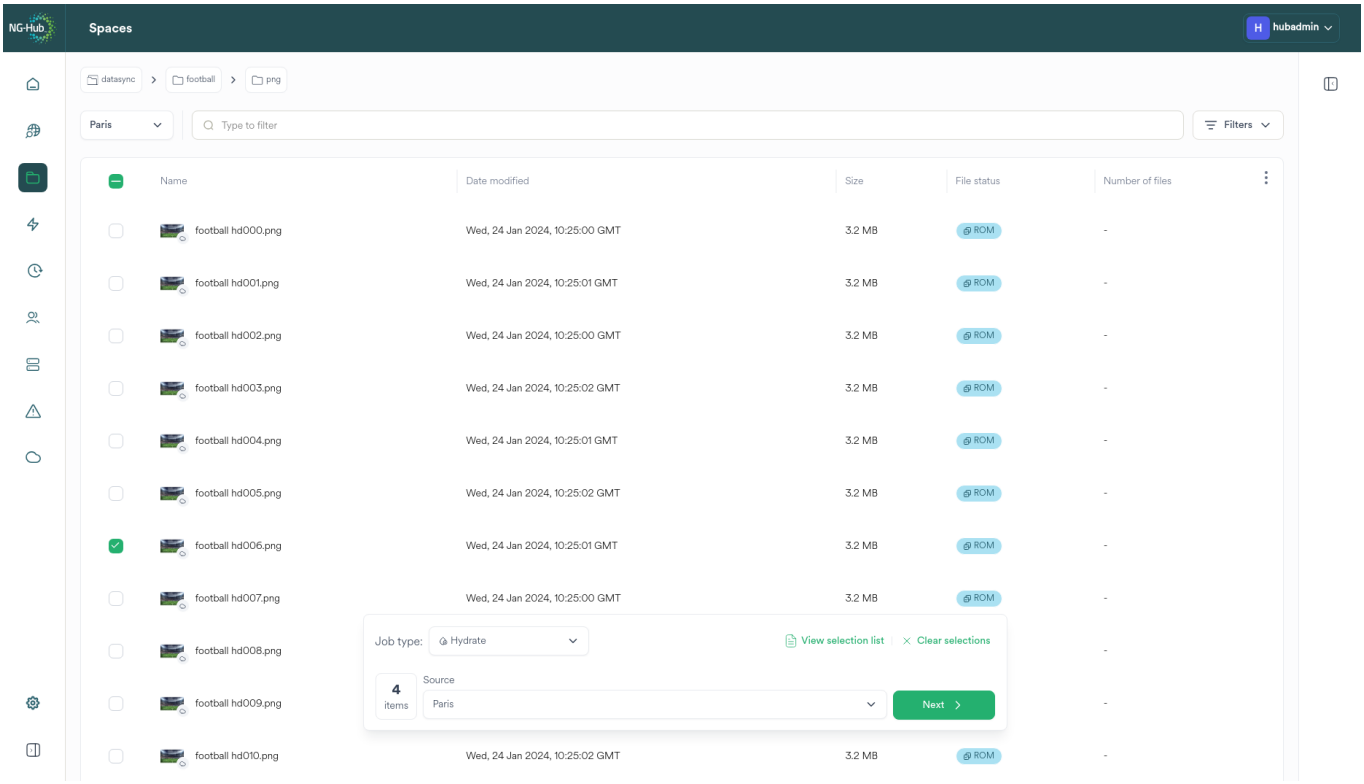
- Choose whether to retain the assets from the current Site or to clear all the current selections in order to add new selections from other Site(s)



When the Job Panel selection contains selected items, navigating to the Space on the selected Site retains the selections in the Job Panel selection list.

	Click the filebrowser navigation button on the Space card to navigate to the selected Space, filtered by Site. Any assets already selected remain in the Job Panel selection list.

Example of asset selections after navigating to the Space on the Site with assets selected from the Search results:



Spaces

A Space is an area of storage present on or across one or more pixstor which comprises the following capabilities:

- name
- location
- size
- data protection
- performance
- file share

Viewing Spaces

To view available spaces, click the Spaces menu button.

	Navigates to the Spaces screen

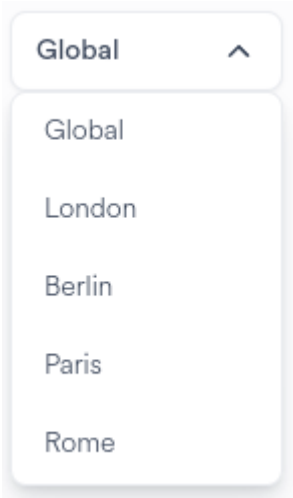
The ability to view spaces is restricted by group membership.

- Administrators can view and administrate all Spaces across all Sites
- Users can view and use all Spaces across all Sites
- Restricted users can view and use a defined subset of Spaces on associated Sites

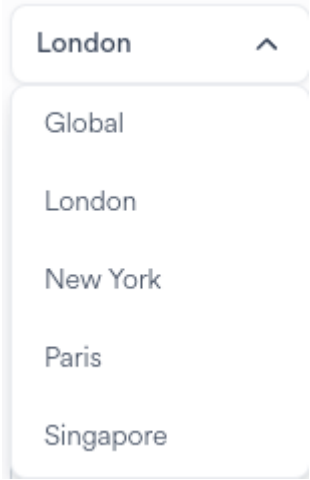
Hub provides two views of Spaces - Global and Local.

- Global displays all Spaces on all Sites
- Local displays the Spaces on a specific site

The default view of Spaces is Global.



To switch to a site-centric view, select the specific site from the Spaces drop-down menu.

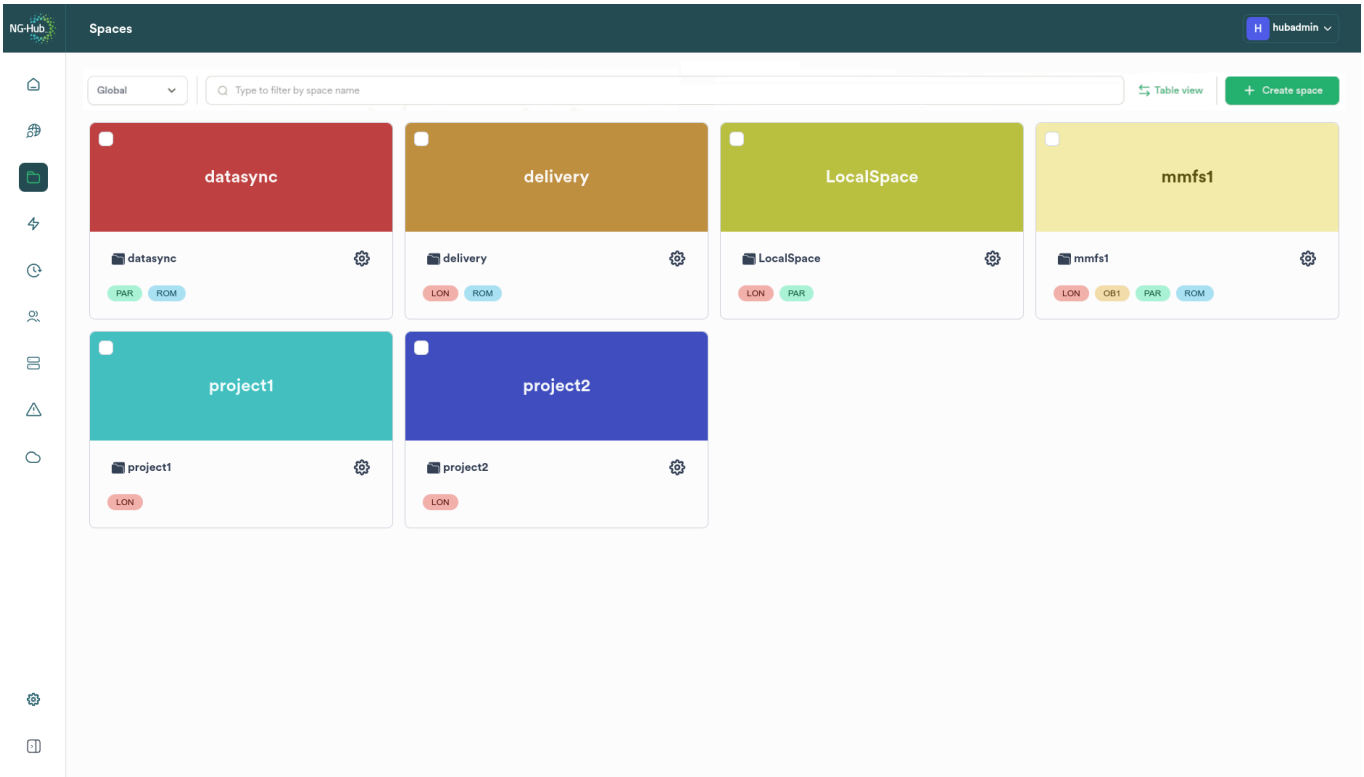


To switch to Global view, select Global from the Spaces drop-down menu.

Global View

Global view displays all Spaces on all Sites.

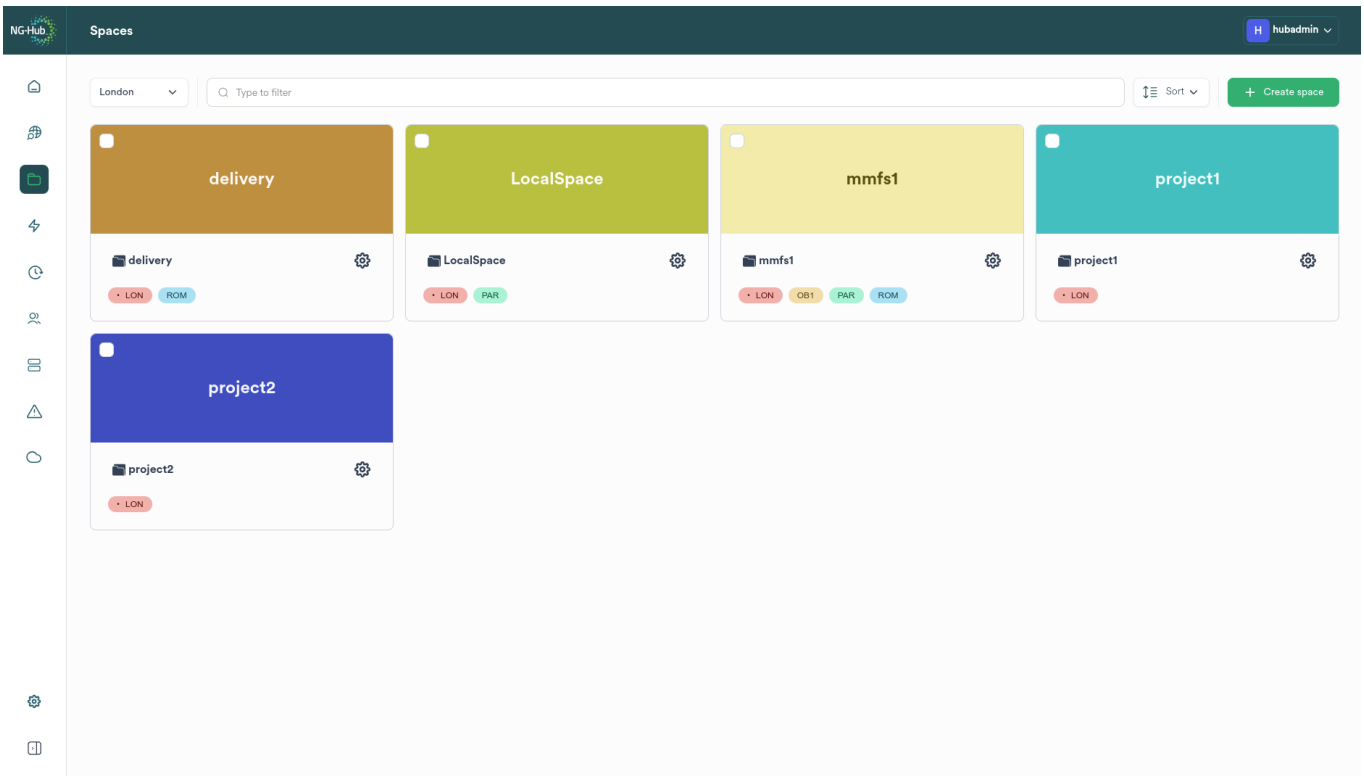
The example below displays all the Spaces across all Sites.



Local View

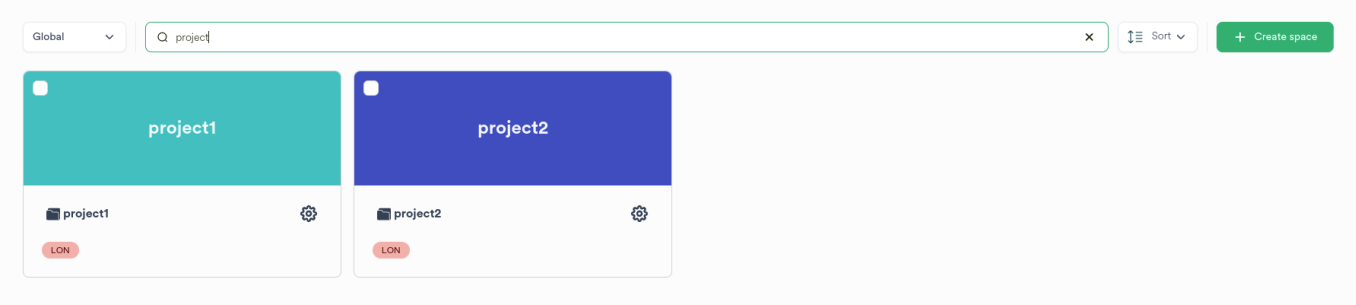
Local view displays the Spaces on a specific Site.

The example below displays less Spaces than Global as the site is participating in fewer Spaces.



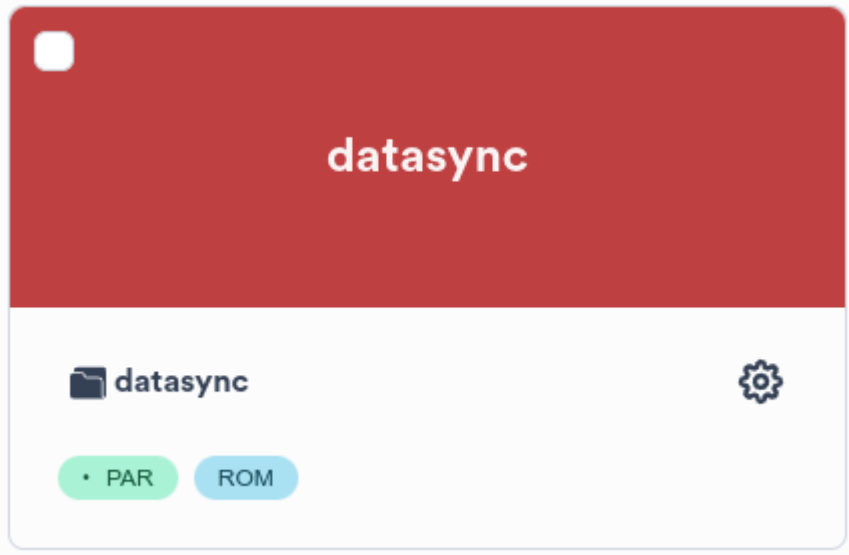
Filtering the Space View

To display Spaces matching keywords, enter the keywords in the filter bar.



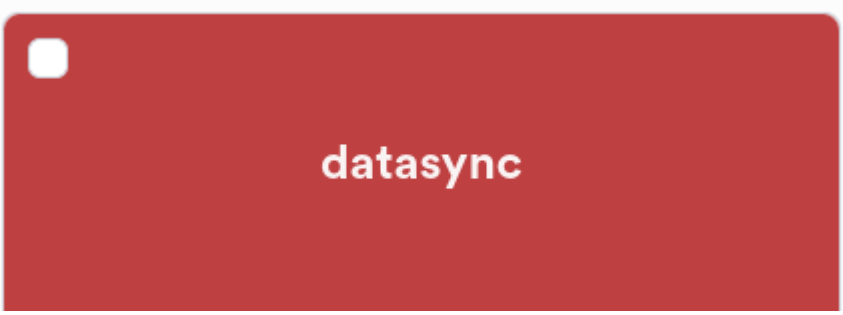
The Space Card

A Space is displayed as a card in the Space view.



A Space Card comprises:

Space Card Header



Denotes the name of the Space.

Click the Header to view the Space File Browser.

Space Name

Denotes the name of the Space.

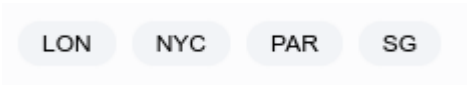
Click the Space Name to view the Space File Browser.

For more information on the Space File Browser, see Browsing Spaces

Space Site Chips

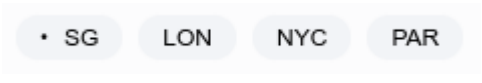
Site Chips denote the sites on which a Space is present.

Click a Site Chip to change the view to the Local View of the Space File Browser for a specific Site.



In Global view all sites are listed in alphabetical order.

In Local View, the selected site is identified with a dot.



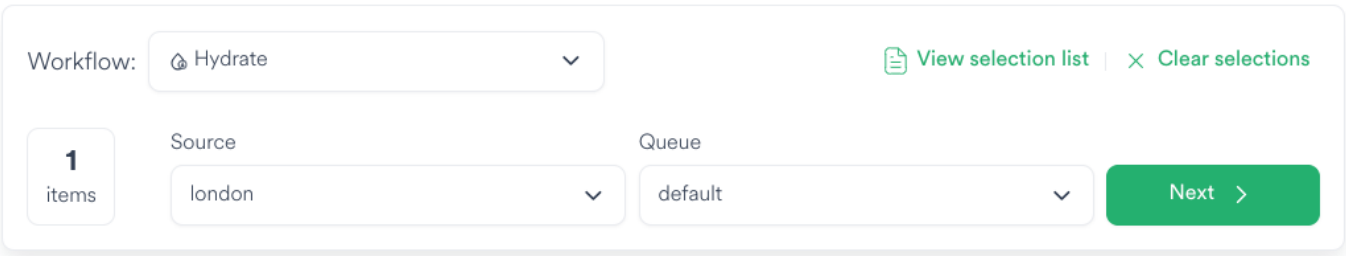
All other sites are listed in alphabetical order after the local site.

Space Selector Checkbox

Selecting the Space Selector Checkbox raises the Job Creator Panel to perform data operations on the entire Space.

Tip: To perform more granular operations on data inside the Space, refer to Selecting files and folders

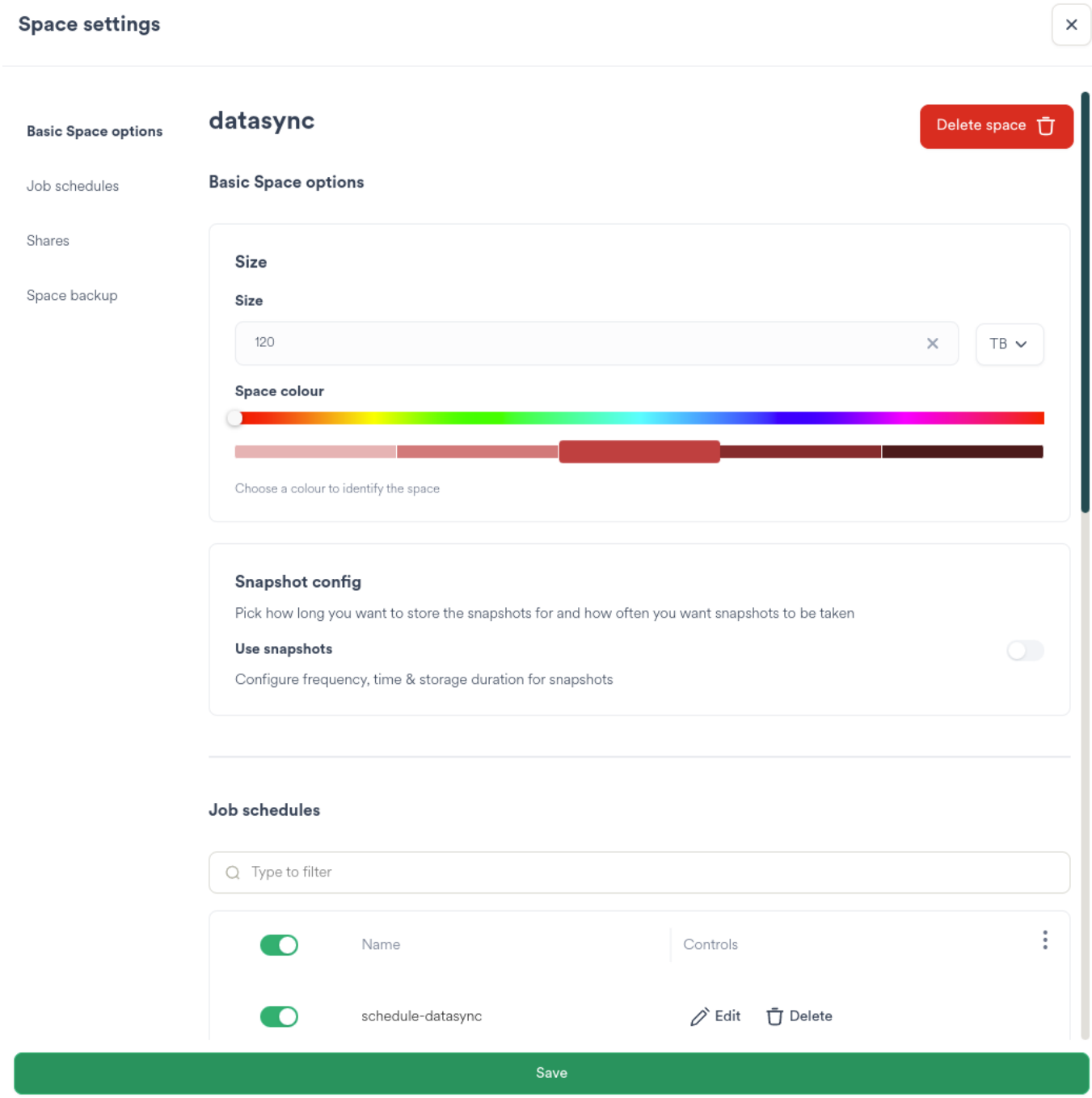
The Job Creator Panel allows data operations on the Space as a whole.



For further information refer to Using the Job Creator Panel.

Space Settings

Clicking the Settings button on the Space card raises the Settings dialog for the Space.



Important: Adding or Configuring a Space can only be performed by a Hub Administrator or User with Space Administration rights granted through group management.

Job Schedules

Important: In the schedule settings, hidden workflows types are also available for selection in the Workflow drop down menu. These are workflows defined with `visible:false`.

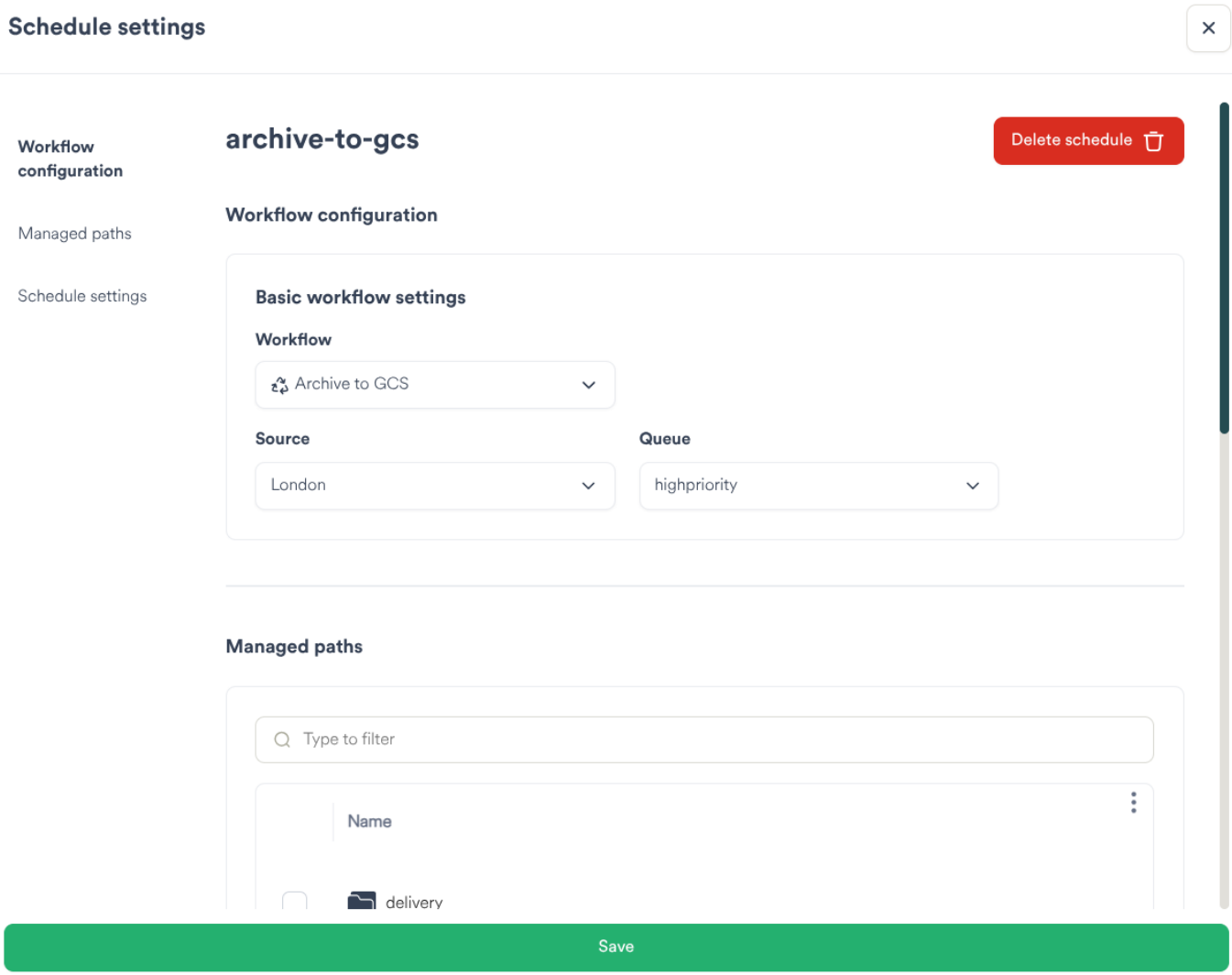


Table View

Hub provides an alternative table layout for Spaces.

Table layouts can be preferable for viewing many Spaces or extended high level Space information.

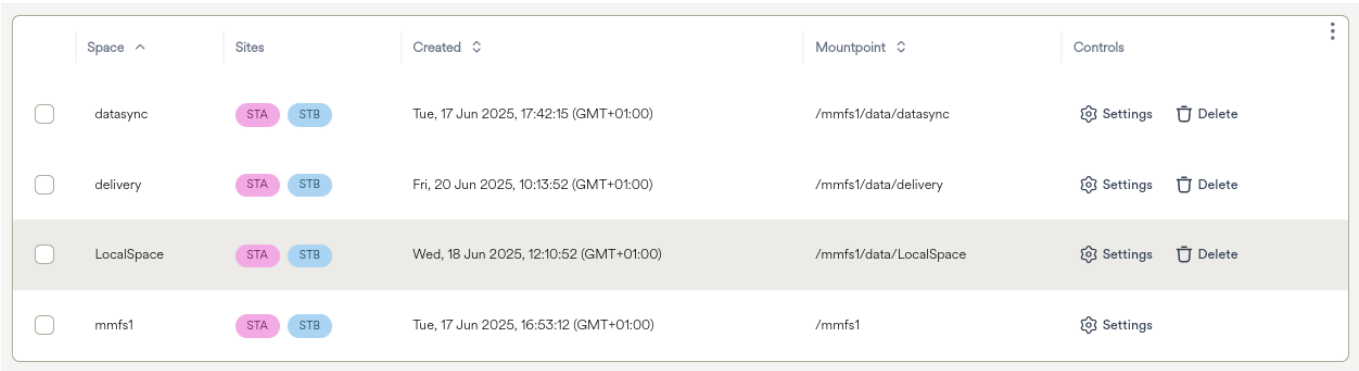


Table view	Clicking the Table view button to switch to Table Layout
Cards view	Click the Cards view button to switch to Cards Layout

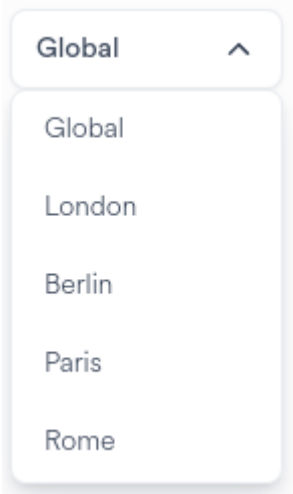
Browsing Spaces

The Spaces File Browser displays the file and folder contents of a Space across multiple or specific sites including additional contextual information such as file counts, size, metadata and status.

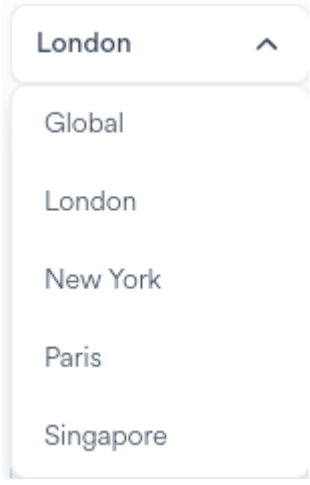
The Space File Browser provides two views of Spaces file and folder content - Global and Local.

- Global displays the file and folder content for a Space across all Spaces on all associated Sites
- Local displays the file and folder content for a Space on a specific Site

The default view of the Spaces File Browser is Global.



To switch to a site-centric view, select the specific site from the Spaces drop-down menu.

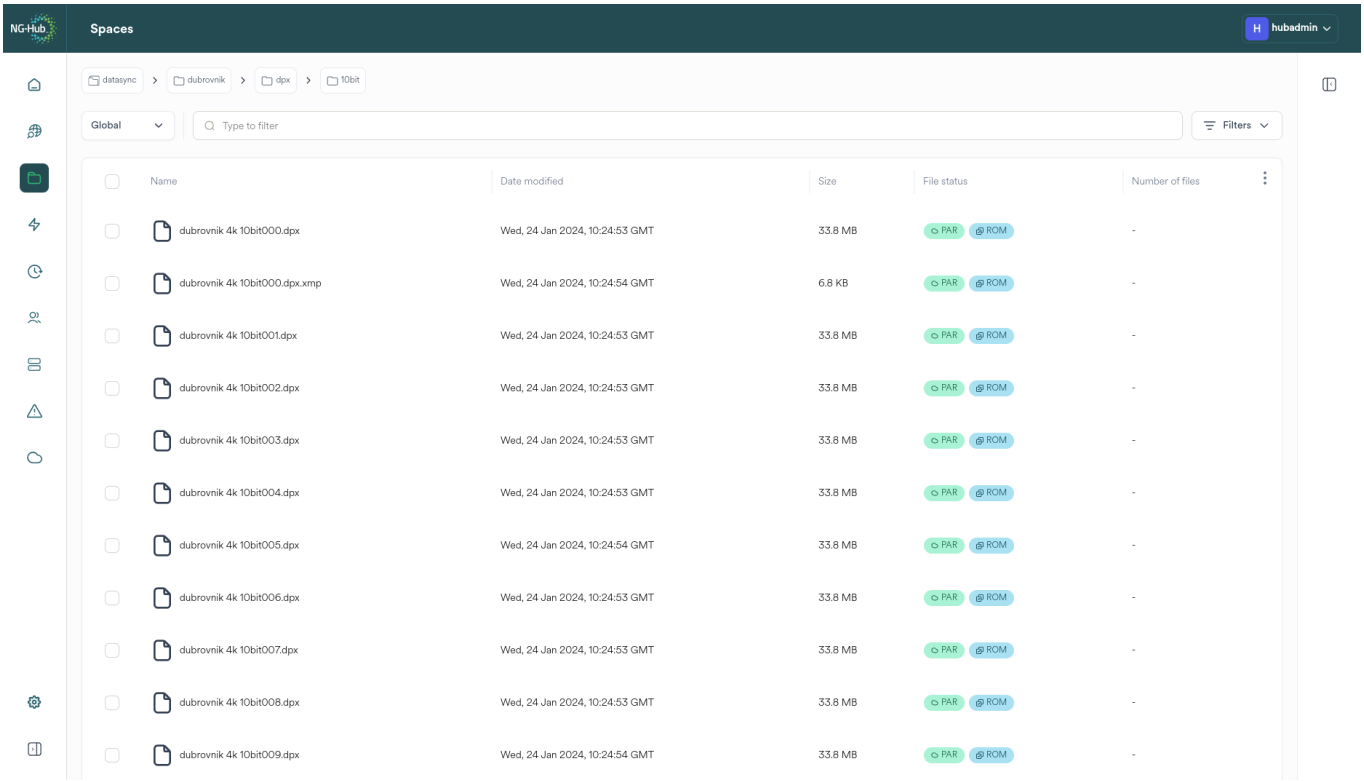


To switch to Global view, select Global from the Spaces drop-down menu.

Spaces File Browser Global View

Global view displays the file and folder content for a Space across all Spaces on all associated Sites

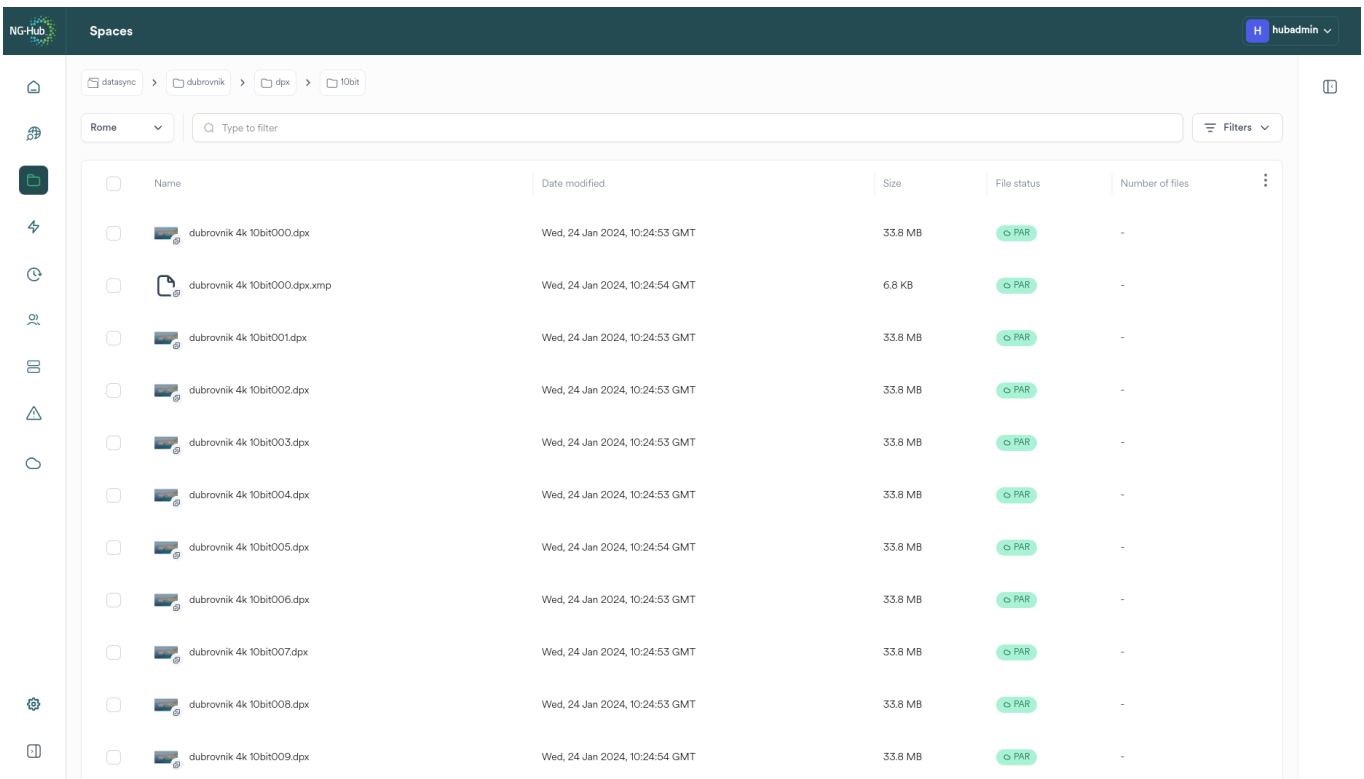
The example below displays the file and folder content for the Spaces across all Sites.



Spaces File Browser Local View

Local view displays the file and folder content for a Space on a specific Site

The example below displays the file and folder content for a Spaces across all Sites with a site-centric view.

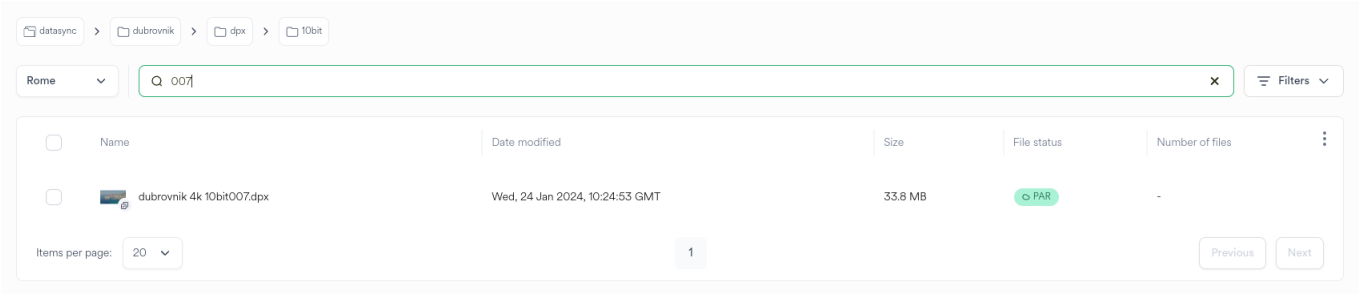


The Space File Browser Screen

The Spaces File Browser screen is comprised of several areas.

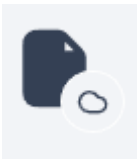
Filtering the Space File Browser

To display files or folders matching keywords, enter the keywords in the filter bar.



Space File Browser Icon Statuses

In Local View the Spaces File Browser displays additional information to designate the status of the file on the Site's pixstor file system.

	Spaces File Browser Local View displays the default file icon if the file or folder is unmanaged. An unmanaged file has not yet been processed by Ngenea operations.
	Spaces File Browser Local View displays the cloud icon if the file or folder is Pre-Staged. A pre-staged file is present on the Site's pixstor file system and has an identical copy in the Ngenea target (E.G. AWS cloud bucket).
	Spaces File Browser Local View displays the cloud icon if the file or folder is dehydrated. A dehydrated file has only metadata present on the Site's pixstor file system and has a fully hydrated identical copy in the Ngenea target (E.G. AWS cloud bucket).

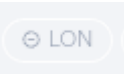
Space File Browser File Statuses

Site Chips denote the sites on which a Space file or folder is present and their status.

In Global view all sites are listed in alphabetical order.

In Local View, the selected site is identified with a dot.

All other sites are listed in alphabetical order after the local site.

	Spaces File Browser Local View displays the default file icon if the file or folder is unmanaged. An unmanaged file has not yet been processed by Ngenea operations.
	Spaces File Browser Local View displays the cloud icon if the file or folder is pre-staged. A pre-staged file is present on the Site's pixstor file system and has an identical copy in the Ngenea target (E.G. AWS cloud bucket).
	Spaces File Browser Local View displays the cloud icon if the file or folder is dehydrated. A dehydrated file has only metadata present on the Site's pixstor file system and has a fully hydrated identical copy in the Ngenea target (E.G. AWS cloud bucket).
	Spaces File Browser Local View the file or folder displays a greyed out site with a circled dash where the file is not present on the associated site.

Space File Browser File Attributes

The Space File Browser displays information regarding the files and folders within the Space, globally on all Sites or local to a Site.

Option	Description
Name	The file or directory name
Date Modified	The date and time of last modification
Size	The size of the file on the pixstor file system
File Status	The file status per-site
Number of Files	If the item is a directory, displays the total file count within the directory tree thereunder

Viewing File or Folder Metadata

Metadata Panel

Click a file or directory name to view the associated metadata:

Item info

Name	dubrovnik 4k 10bit000.dpx
Type	file
Path	/mmfs1/data/datasync/dubrov...
Site names	Rome
Size	33.8 MB
Date changed	24.01.2024, 10:24 GMT
Date accessed	24.01.2024, 10:24 GMT
Date modified	24.01.2024, 10:24 GMT

Hydration status

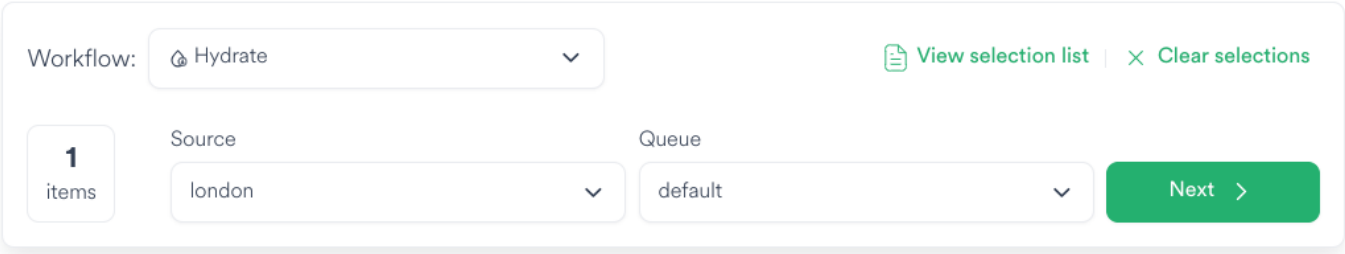
Rome	Premigrated
Paris	Migrated

The metadata panel provides extended metadata, including a summary of the file status on each site participating within a Space.

Where an asset has been ingested via Search thumbnails are displayed for supported file types.

Using the Job Creator Panel

Explain panel areas, buttons and operations

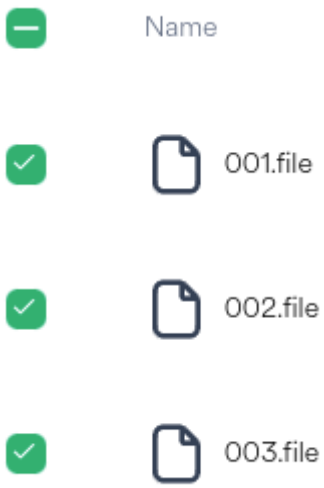


Selecting files and directories

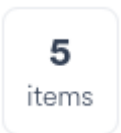
Selecting an individual file or directory populates the selection list with the item. Selecting a directory populates all items within the chosen directory tree.

Selections of the entire Space or a directory are each counted as 1 item.

Select the ‘select all’ checkbox next to the Name field in order to select the entire Space.



As selections are added or removed, the selected item count is updated:

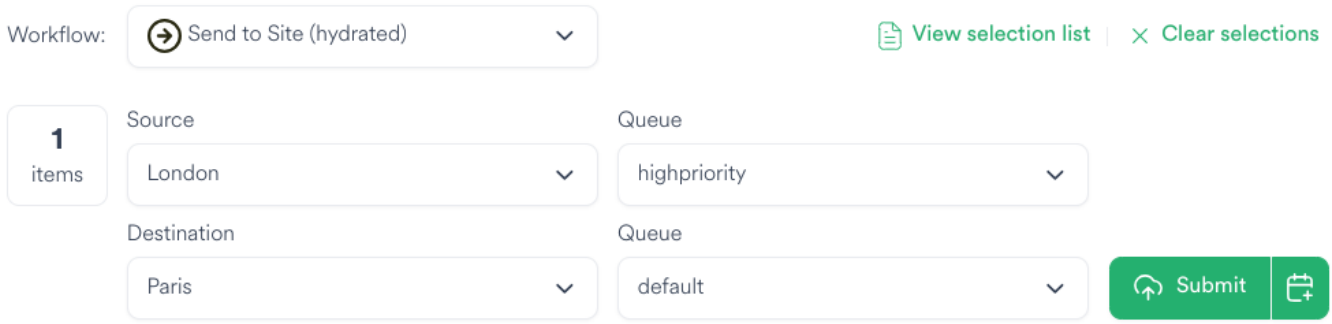


View selection list	Click the View Selection List button to show the files and directories selected to be processed by the chosen workflow
× Clear selections	Click the Clear selections to remove all files and directories currently selected from the selection list
Next >	Click the Next button to enact the Job

Sites and queues

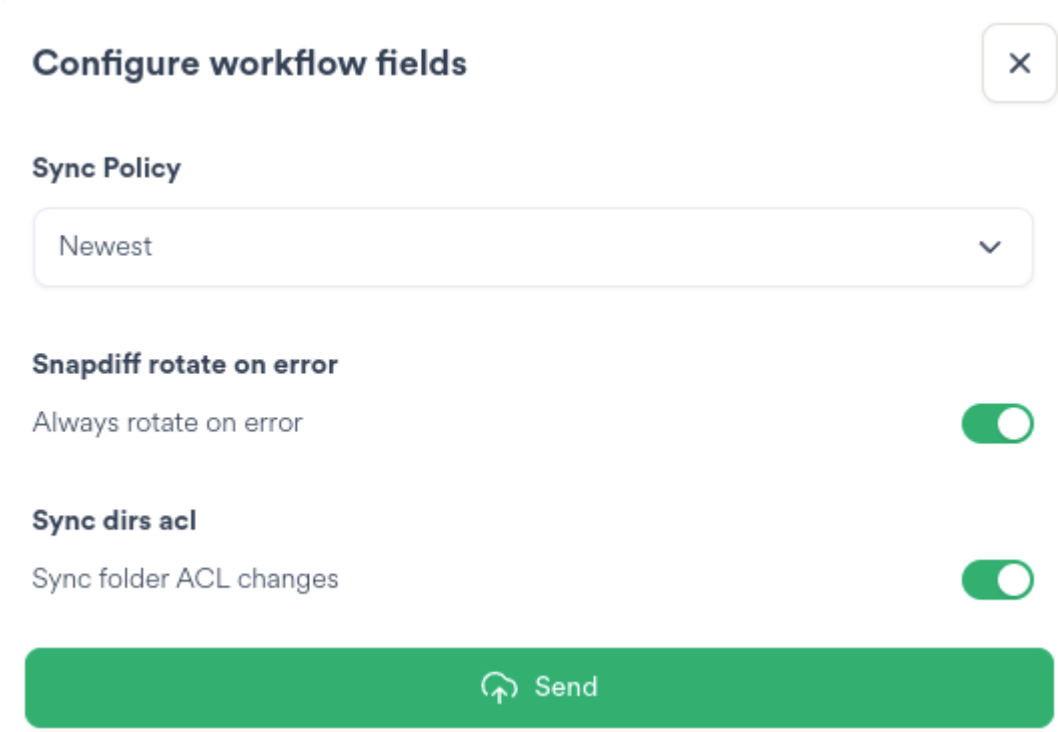
Source london	Select the site where the job will run.
Queue default	Select the queue the job will be assigned to.

If the workflow runs across multiple sites, such as “Send to Site”, select a source site and queue, and a destination site and queue.

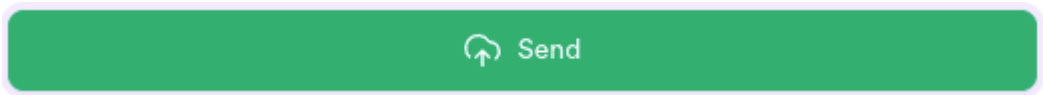


Workflow fields

Jobs with Workflows which require additional user decisions prior to enacting the Job raise the Configure workflow fields dialog:



After entering the field information [if required], select the button at the bottom of the dialog to submit the Job.



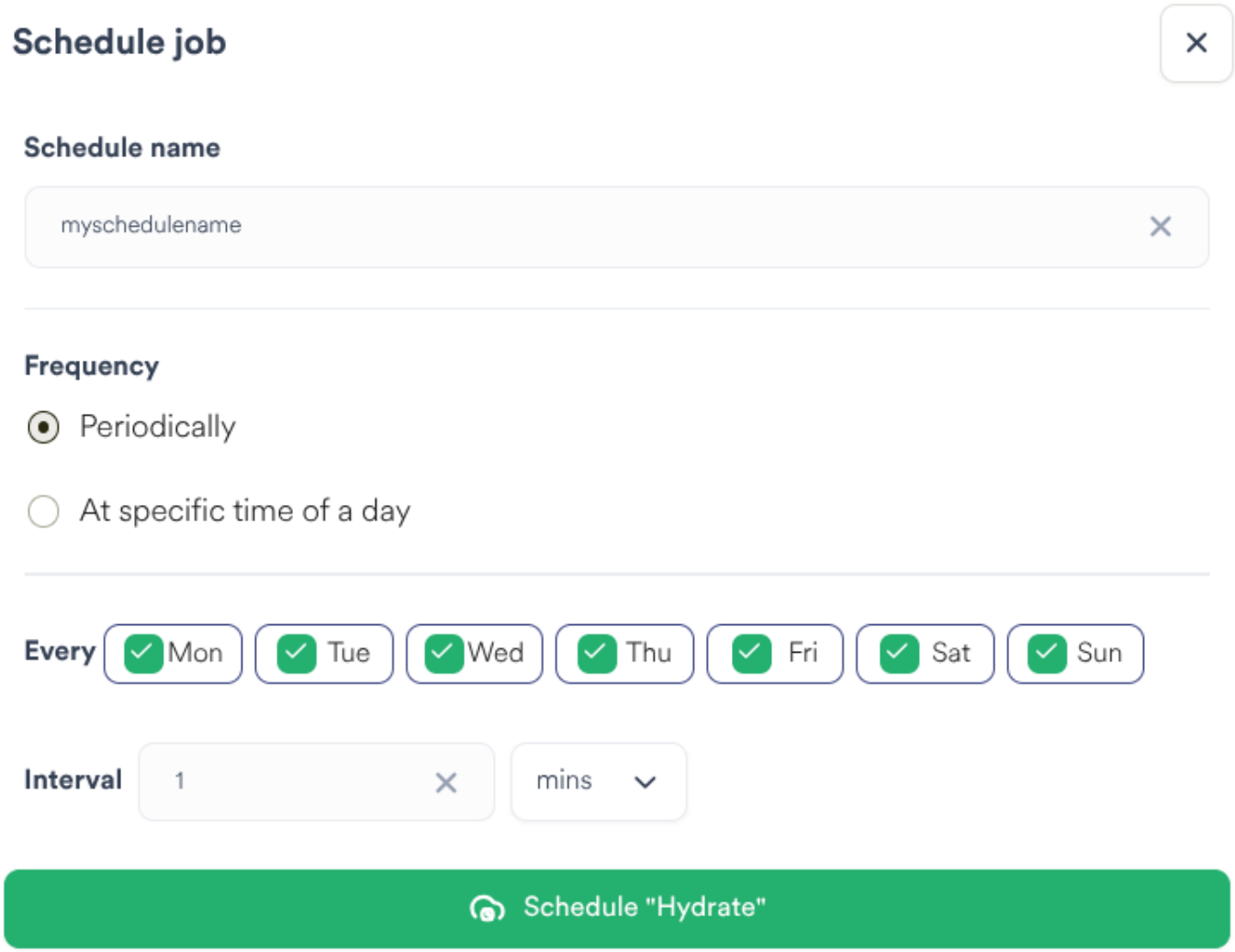
Click the button at the bottom of the dialog to submit the Job immediately. The button label is changed dependent on the workflow chosen.

Scheduling a workflow

To Schedule the Job to run later, click the Schedule button.

Click the Schedule button to Schedule the Job for later

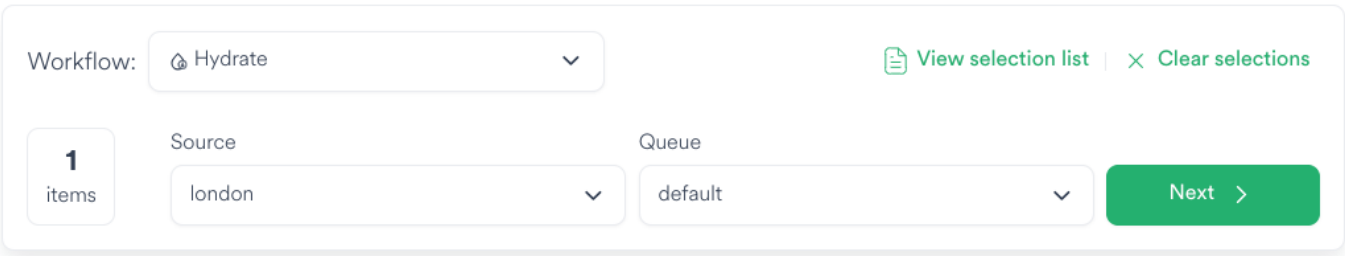
The Schedule button raises the Schedule Job dialog.



- Enter the name of the Schedule
- Select the desired Schedule settings
- Click the Schedule button to create the Schedule

See Schedules for more details for schedule workflows.

Job Types (Workflows)



Default Workflows

Dehydrate

The Dehydrate workflow transfers the file metadata and data to an Ngenea target (E.G. an AWS cloud bucket). After dehydration the file appears to be normally present alike any other file on the pixstor file system, but consumes no space. Reading the file automatically hydrates the file with data content allowing the user to read the file as normal.

Pre-Stage

The Pre-Stage workflow transfers the file metadata and data to an Ngenea target (E.G. an AWS cloud bucket). After migration the file on Ngenea target is an identical instance of the file on the pixstor file system. The file on the pixstor file system is not dehydrated. Reading the file allows the user to read the file as normal. Pre-Staging can reduce the total time to Dehydrate the same data in future.

Hydrate

The Hydrate workflow retrieves the file data from an Ngenea target (E.G. an AWS cloud bucket). After successful transfer the metadata and data content of the file is present on the destination site. Reading the file allows the user to read the file as normal.

Sync Space to Site

The Sync Space to Site uses file system snapshots to disover changes between the last file system snapshot and when the workflow was run. 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.

Send to Site (hydrated)

The Send to Site (hydrated) workflow transfers data from a source site to a destination site. After successful transfer the metadata and data content of the file is present on the destination site. Reading the file allows the user to read the file as normal.

Send to Site (dehydrated)

The Send to Site (dehydrated) workflow transfers data from a source site to a destination site. After successful transfer the metadata of the file is present on the destination site. To a user the file appears to be normally present alike any other file on the pixstor file system. Reading the file automatically hydrates the file with data content allowing the user to read the file as normal.

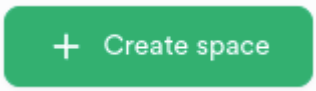
Additional Custom Workflows

Hub can support additional Custom Workflows providing custom operations to data through multiple task steps. Custom workflows are configured and provisioned by a Hub Administrator using the Hub CLI. When provisioned Hub Custom Workflows will appear in the list of workflows available for users to use with their data.

Managing Spaces

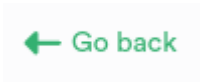
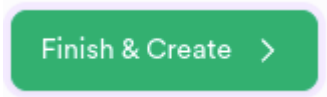
Create a Space Wizard

Click the Create Space button to display a dialog to configure the selected space.



Important: This function can only be performed by a Hub Administrator

Navigating the Wizard

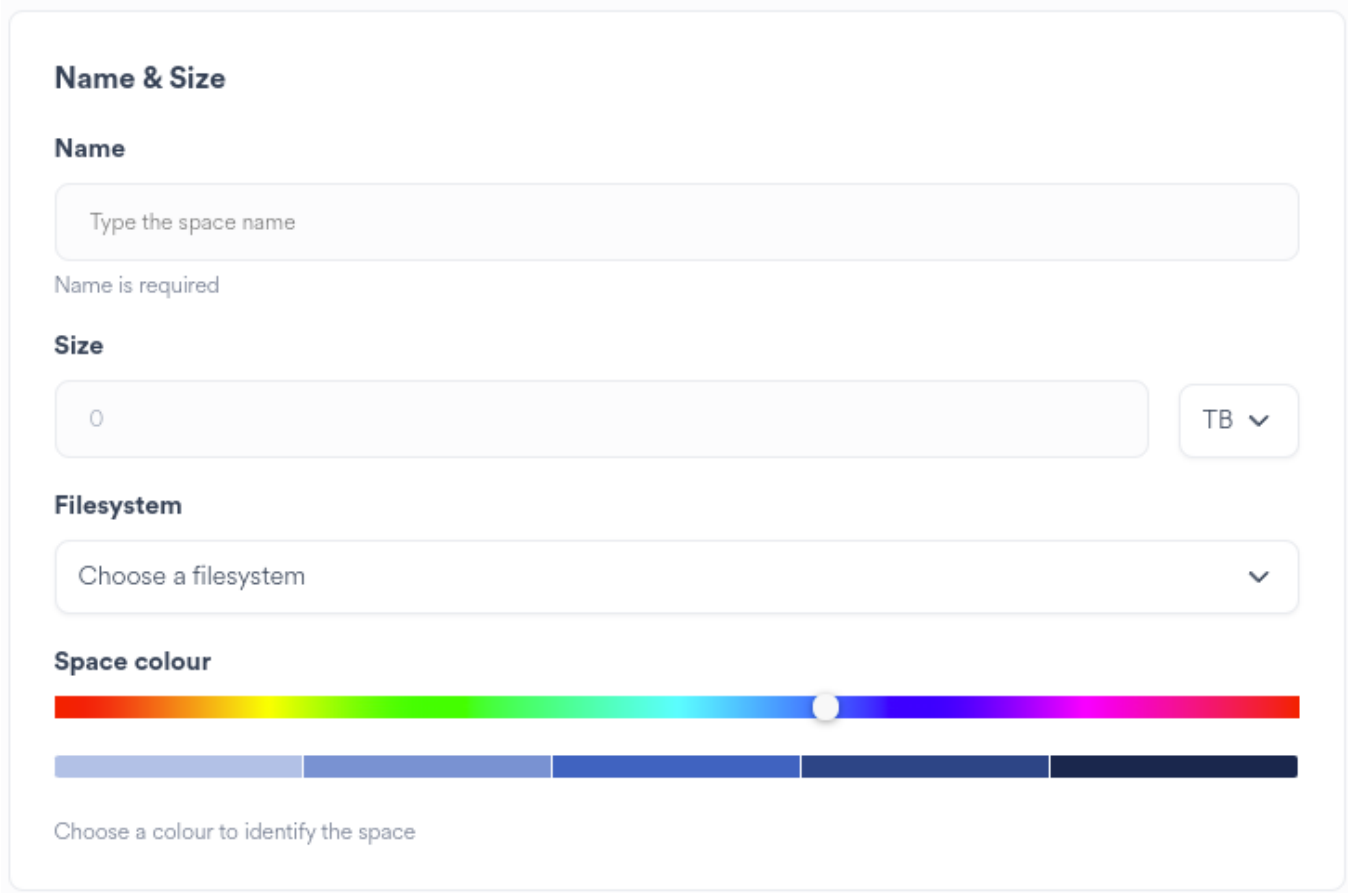
	Click the close button to exit the wizard. Changes are not saved.
	Click the Next button to advance to the next page of the wizard. The Next button is disabled until all required page elements are completed.
	Click the Go Back button to return to the previous wizard page.
	Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Basic Space Options

Spaces are created on pixstor file systems at /mmfs1/data/[Space] or at an alternative custom location.

Spaces can be restricted to a limited amount of data.

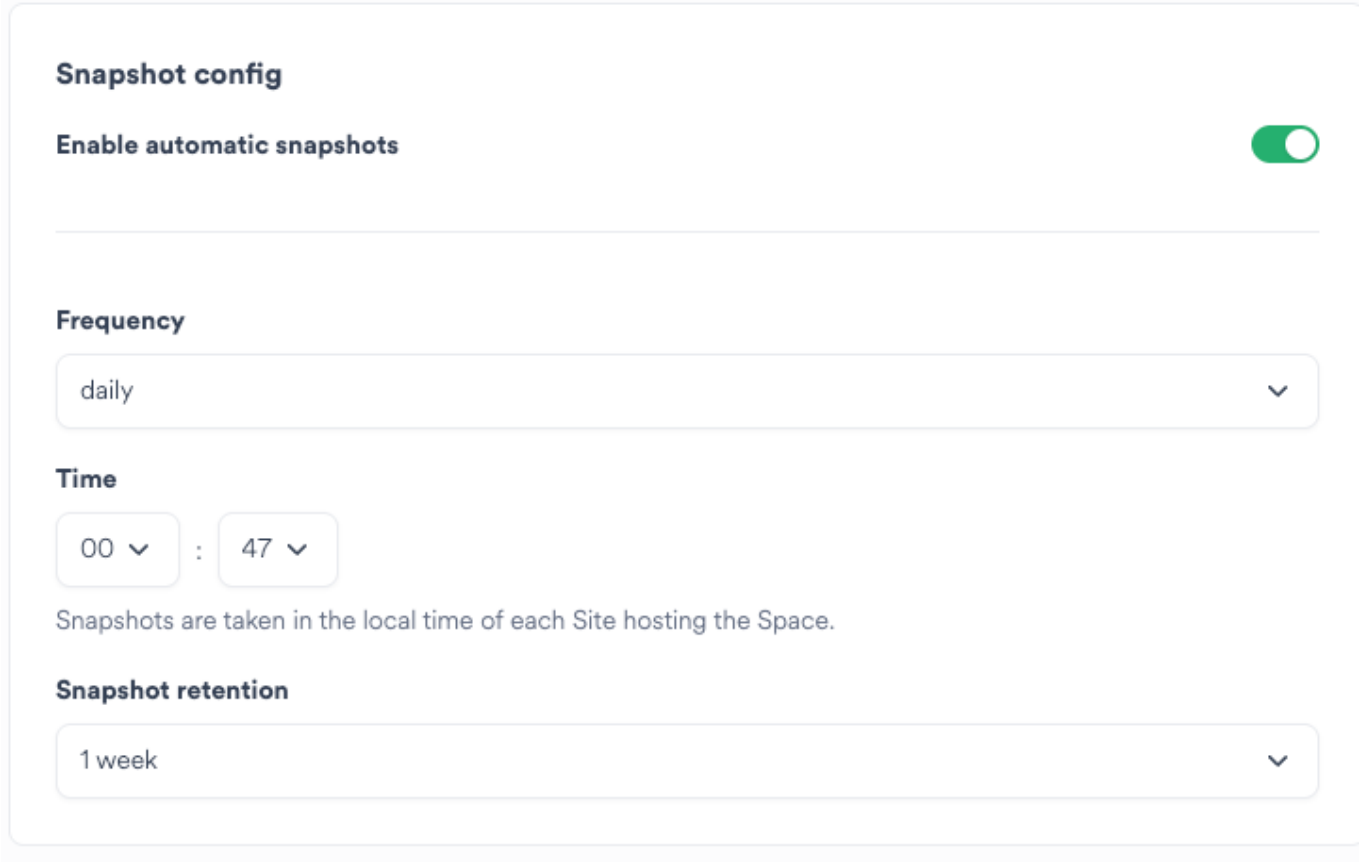
Spaces are defined to a specific file system.



- Enter a name of the Space. The name is case-sensitive.
- Choose the data limit for the Space or unrestricted (Size = 0)
- Select the file system the Space will reside on across all pixstors
- Select a colour for the Space Card

Snapshots create safety copies of data on the pixstor file system.

Snapshots are not backups and should not be used as protection against media failures



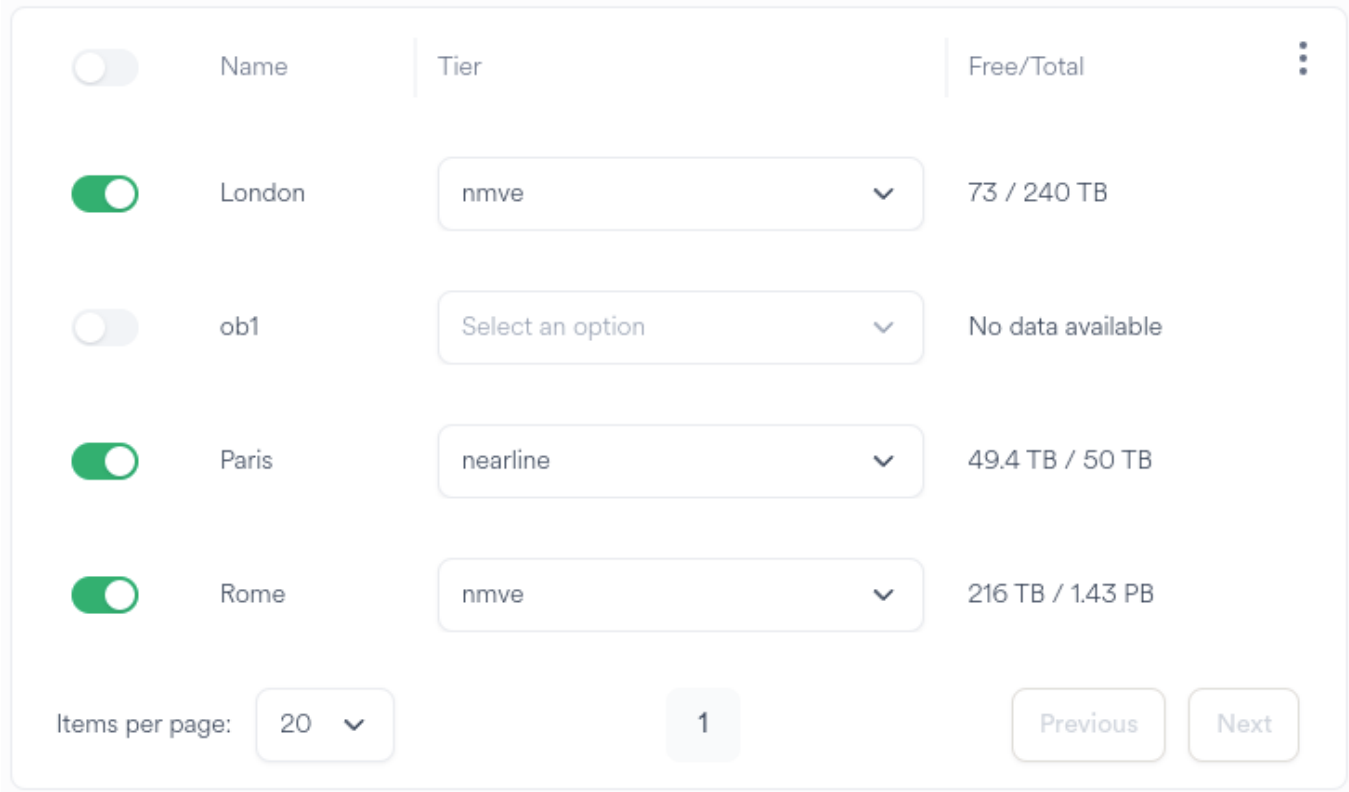
- Choose whether snapshots are required for the Space
- Choose the frequency of the snapshot
- If the frequency is daily or longer, select the time of day on the Site to snapshot the Space
- Snapshots are scheduled and taken based on the local timezone of each Site where the Space is hosted
- Choose how long to retain the snapshots

Select the Sites to create this Space on

A Space can exist across multiple Sites.

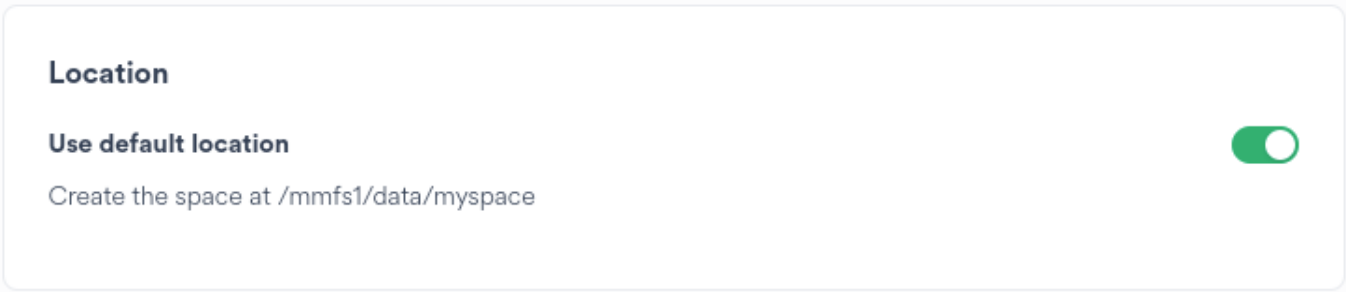
By default data does not move across multiple Sites. Scheduled workflows must be set up to enable cross-site data movement.

Hub provides the ability to synchronise a Space with two Sites - a bi-directional sync.

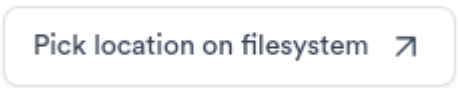


- Select the Sites on which to create the Space
- Select the pixstor file system tier on which to host the files for the Space. The tier is limited to those available for the file system selected in the prior wizard screen.

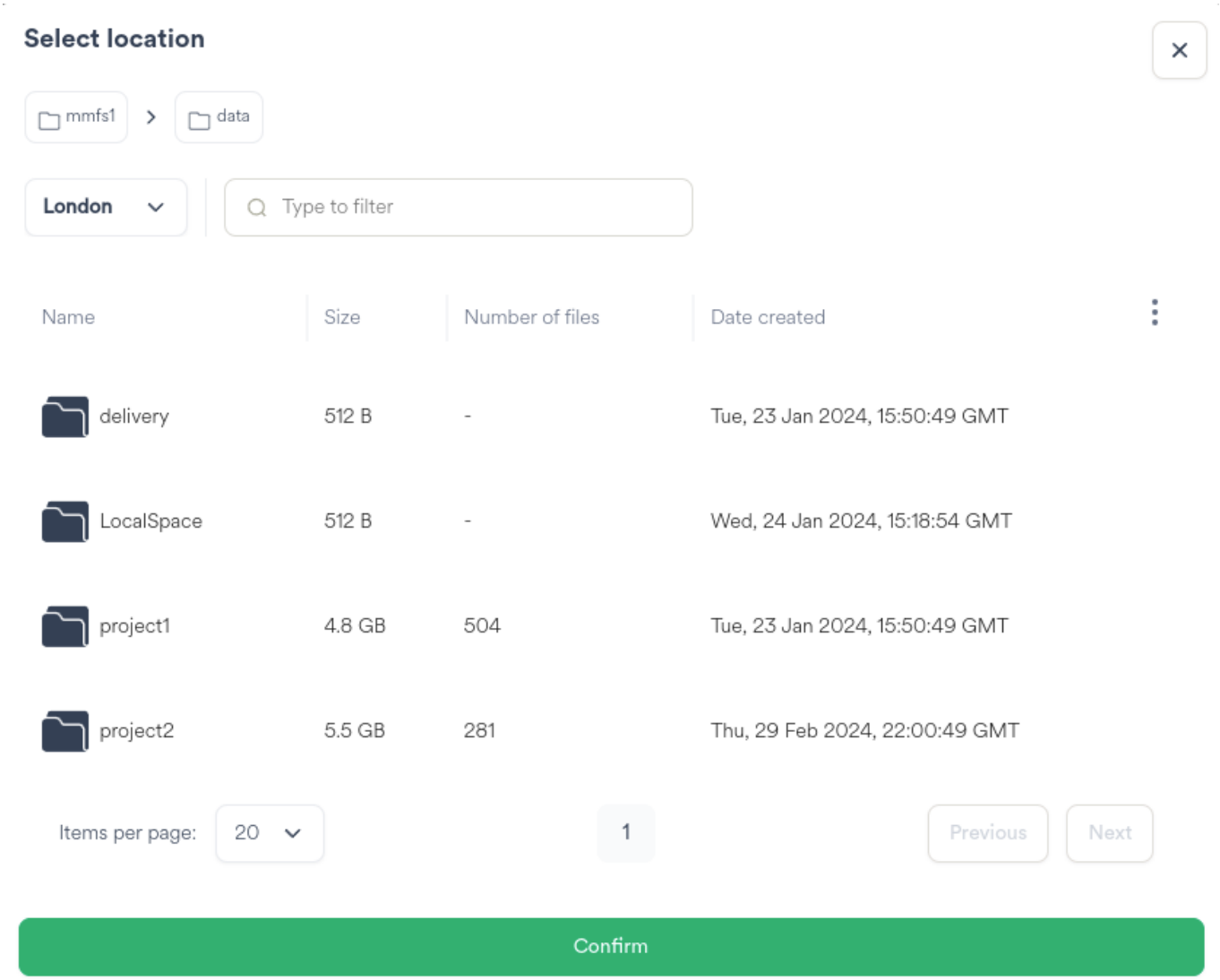
If two Sites are selected the Wizard will prompt to setup a bi-directional sync.



- To select an alternative custom location, deactivate the Use default location slider



Click the Pick location on filesystem button to open the location selector



- Select a Site to browse. Parent directories not present on another Site will be added when a Space is created.
- Navigate to the directory under which to create the Space
- Click the Confirm button to return to the Space creation wizard

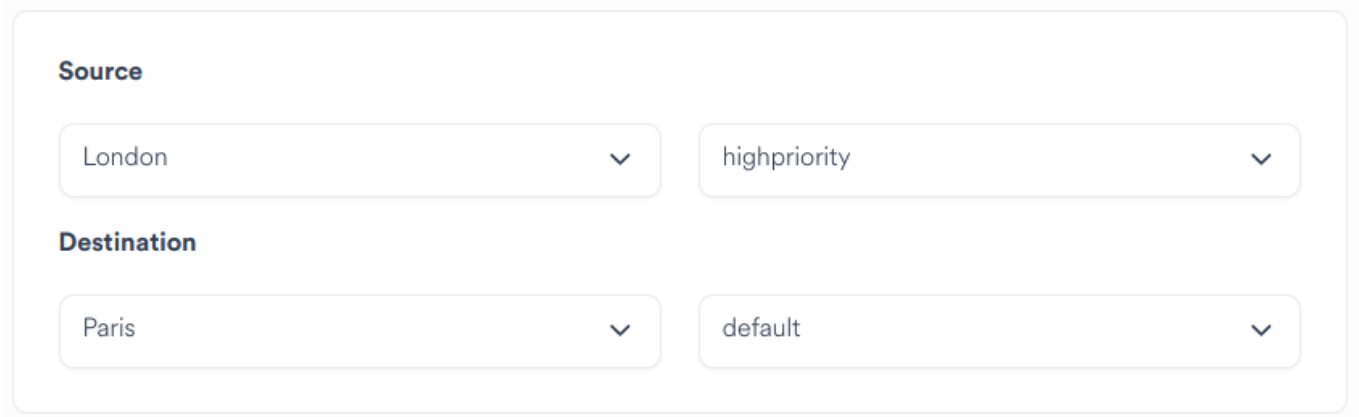
The chosen location is displayed and will be used for all Sites on which the Space is created.

Set up bi-directional sync

If two Sites are selected on the prior wizard page, Hub prompts to setup a bi-directional sync.

Only one bi-directional sync can run at a time.

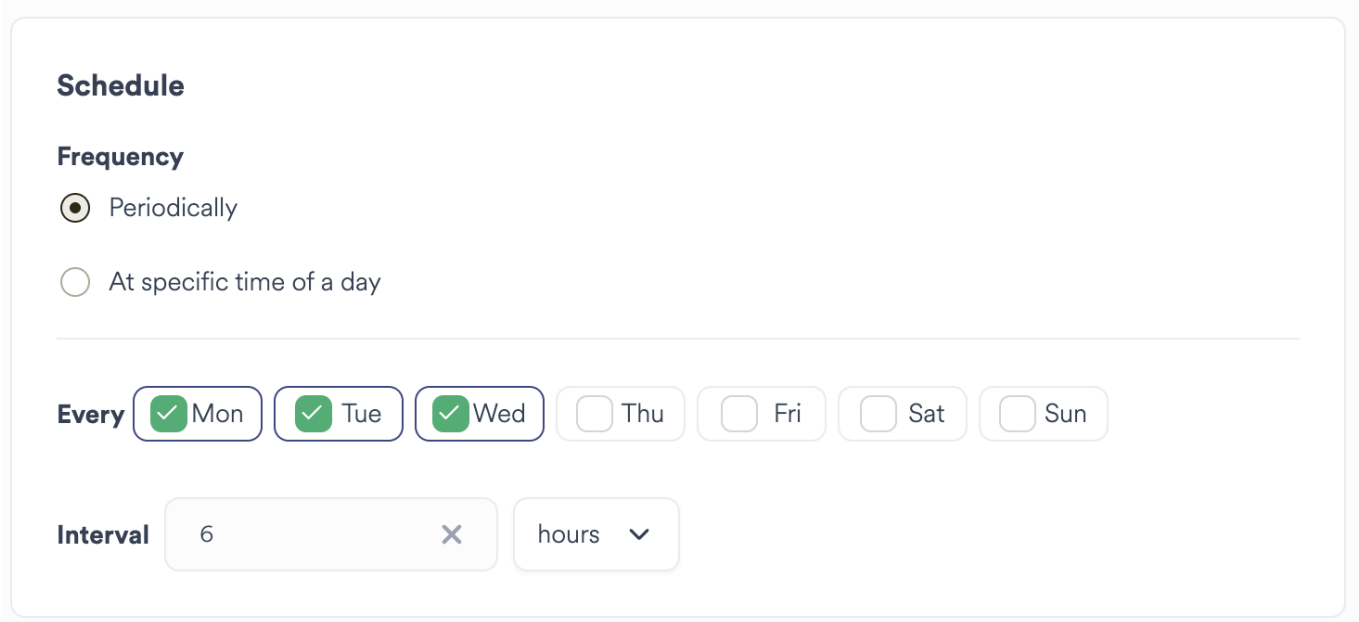
This page can be skipped if no synchronisation is required.



- Select the source Site to launch the first sync from
- Select the other Site as the destination site
- Select the Queue(s) to assign bi-directional sync to

Schedule

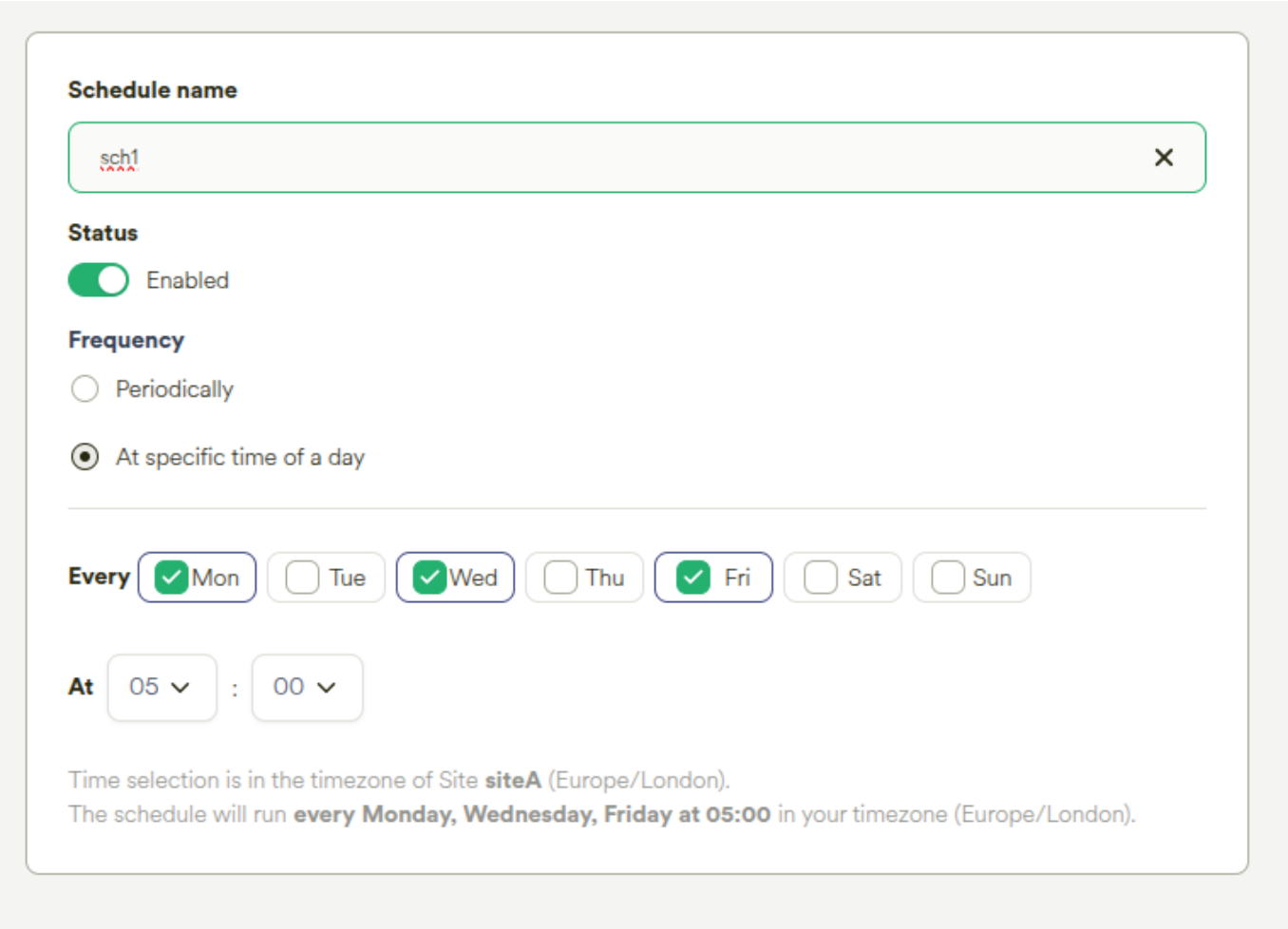
Determine the required frequency of the synchronisation.



Choosing Periodically will ensure that the schedule will run on the next interval set.

E.G:

- 1 hour: The synchronisation will run on the next hour (12.00, 13.00)
- 15 mins: The synchronisation will run on the next 15 minute interval past the hour (15, 30, 45, 00)

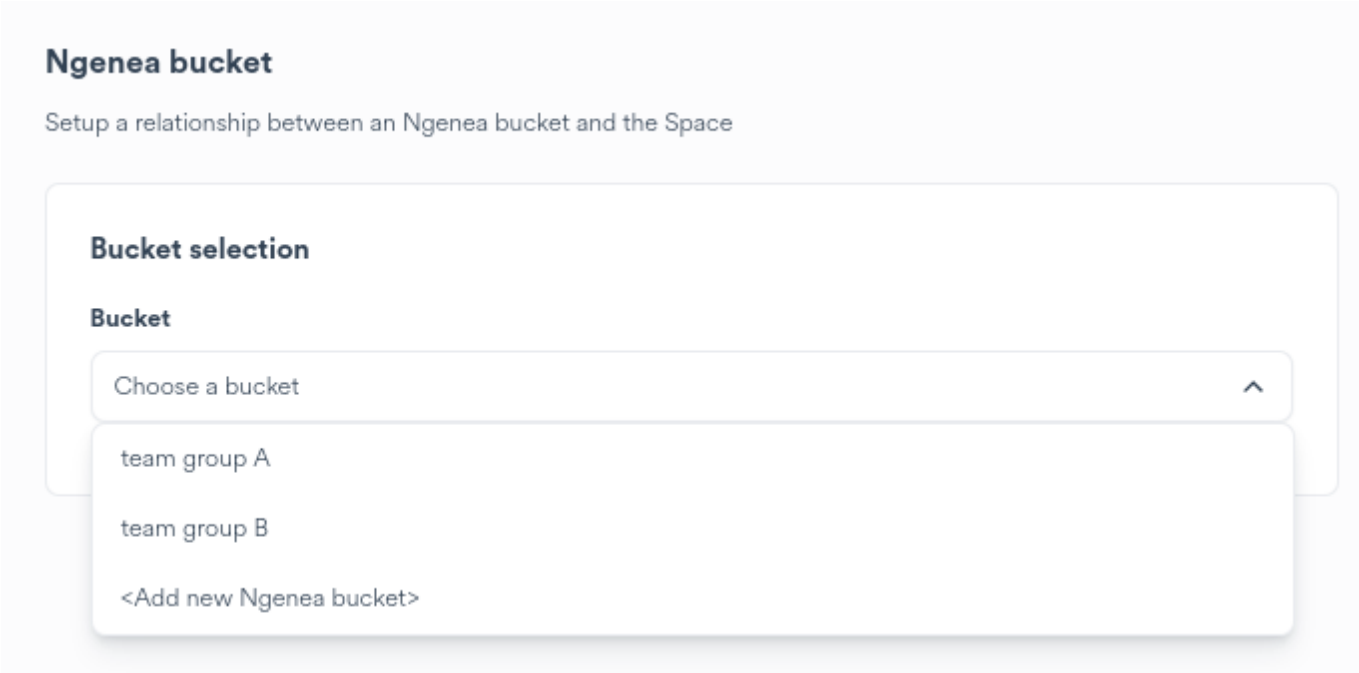


Choosing At Specific Time of A Day allows the Policy to be scheduled once per chosen day at a specific time of day.

Hint: The schedule time is set in the Site's local timezone, but will be stored in the UTC timezone.

Ngenea bucket

Select a pre-defined Bucket or choose to *Add a new Bucket* to associate with the Space.

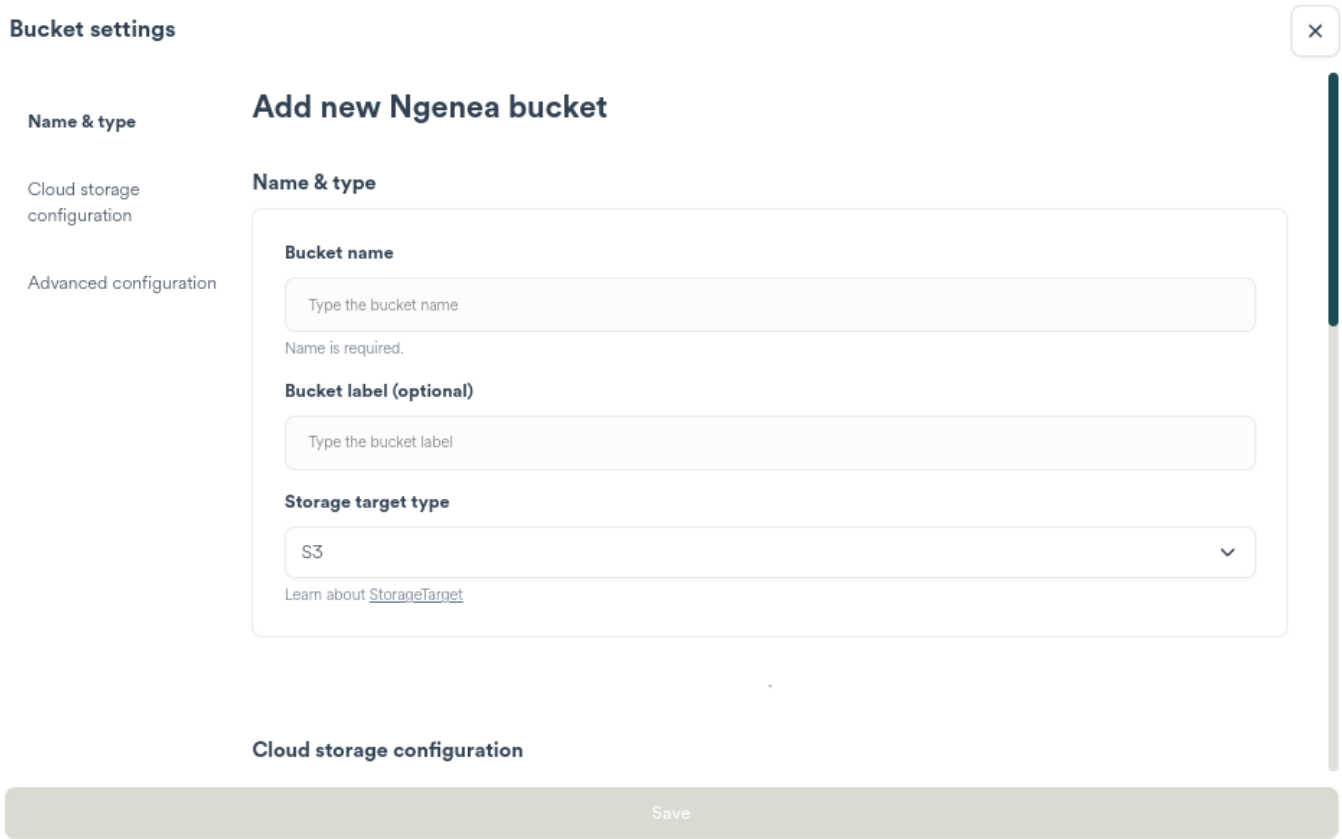


- Select a pre-defined bucket for this Space
- Choose to add a new bucket for the Space

Adding a New Bucket

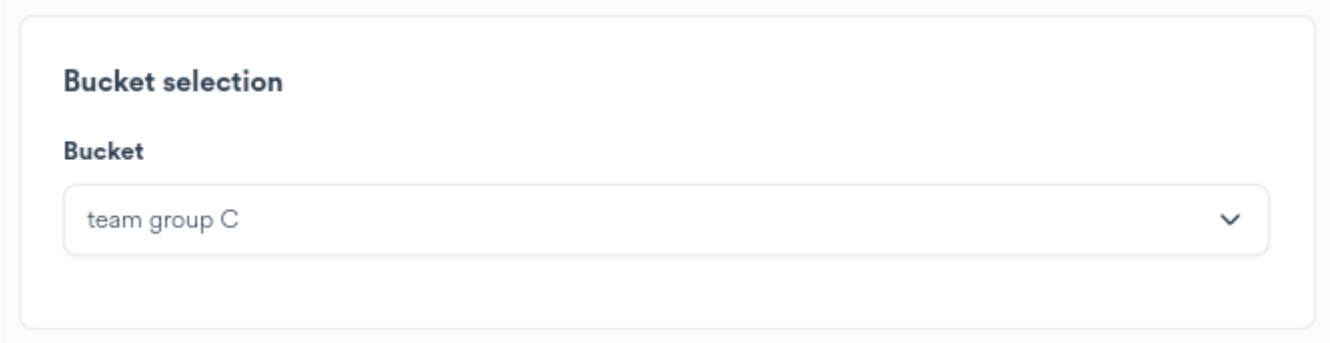
Adding a new Bucket raises the *Add new Ngenea Bucket* dialog.

See Ngenea for more details.



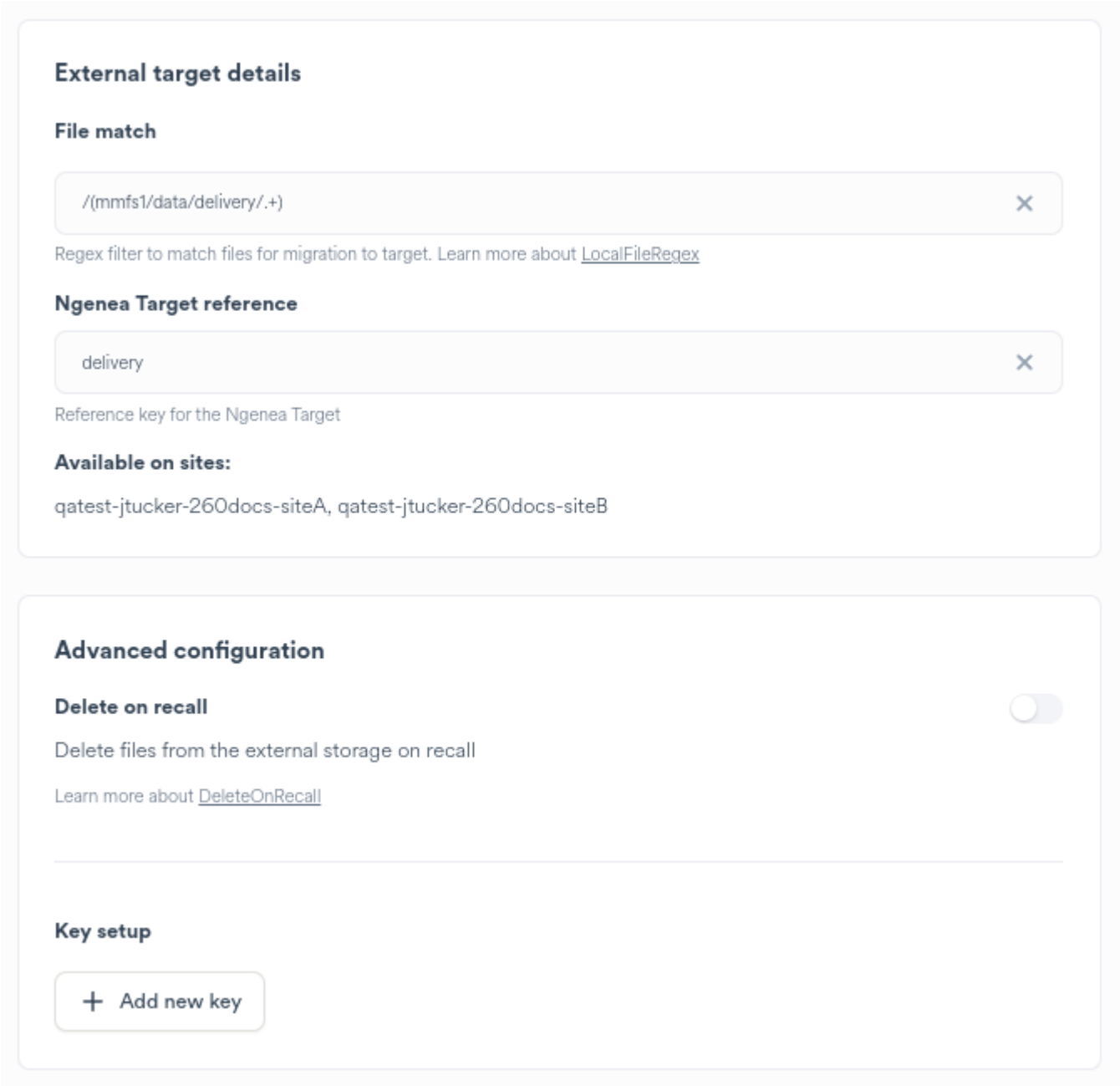
Selecting a Pre-defined Bucket

Selecting a pre-defined Bucket displays the chosen Bucket in the Bucket drop down.



External target details

After either adding a new Bucket, or selecting a pre-defined Bucket the External target details are generated for the Space to review and configure as required.



- Review the external target details and configure as required

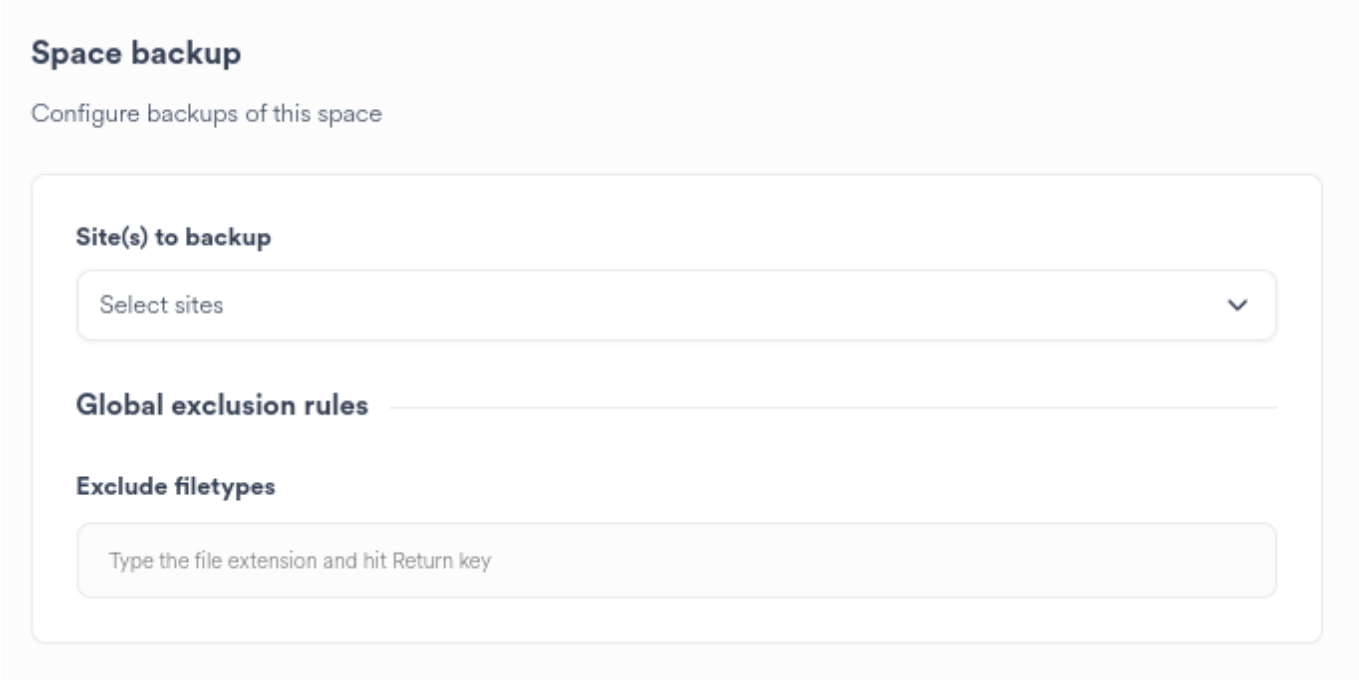
See Ngenea for more details

Shares

Prior to Space creation Shares can be assigned.

See Shares for more details

Spaces backup



- Select the Sites which will perform backups for this space
- Define file extensions which must be excluded from backups across all Sites. E.G. `*.tmp`

Choosing Site(s) to backup presents further backup options.

For more information refer to Backups.

Important: Multiple Sites are not permitted to backup to the same Ngenea Target.

Updating a Space

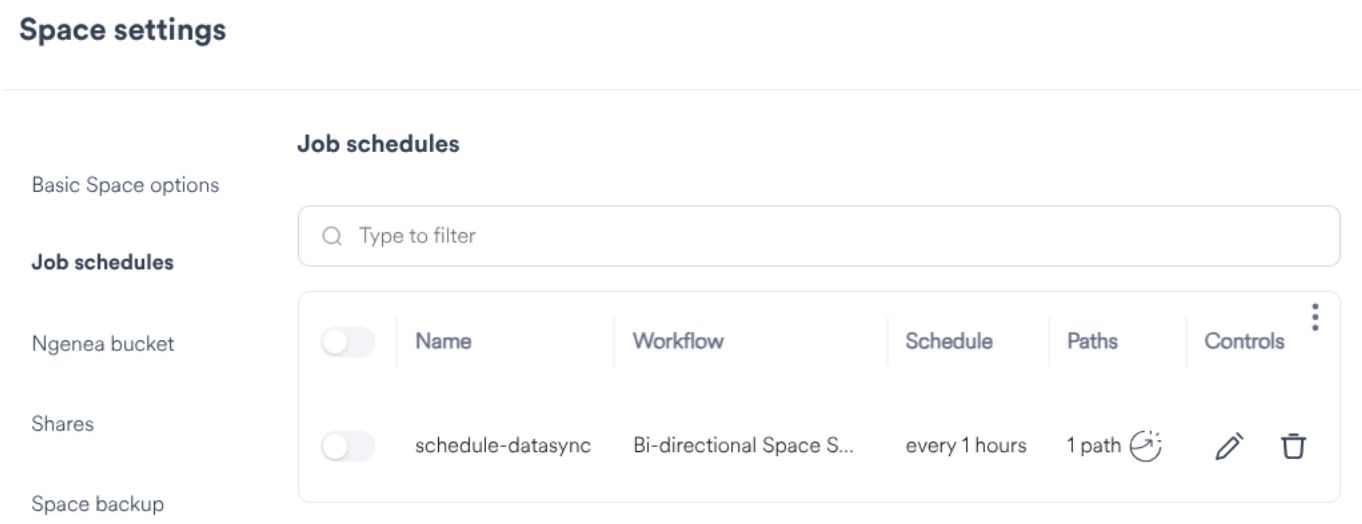
Settings can be modified as required.

	Clicking the Settings button on the Space card raises the Settings dialog for the Space.

Important: Adding or Configuring a Space can only be performed by a Hub Administrator or User with Space Administration rights granted through group management.

Schedules

Click the `Job Schedules` menu item on the Space settings dialog.

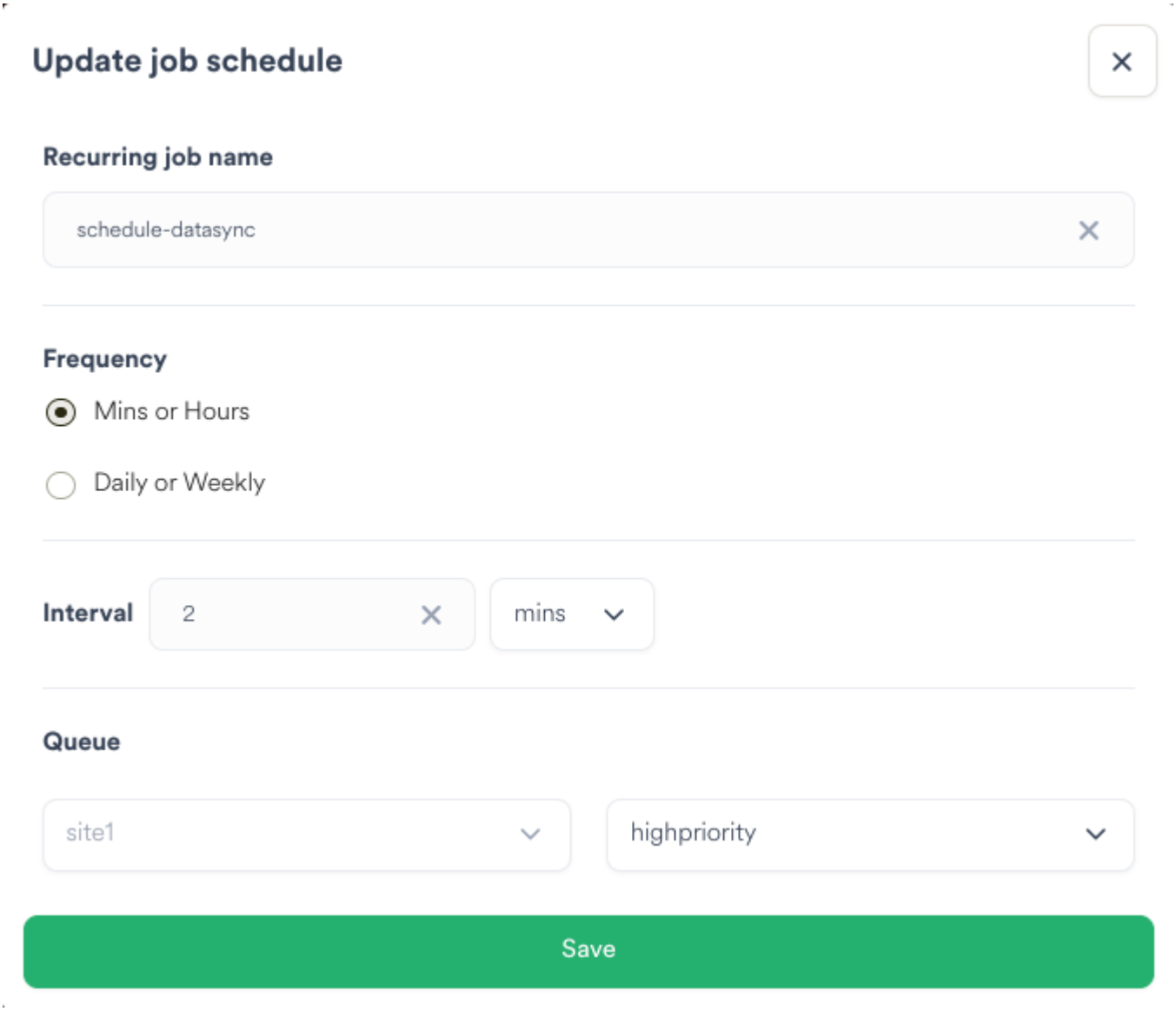


The Schedules for the Space are viewable.

Editing a Schedule

Edit	Click the Edit button for the Schedule to be modified

Clicking the Edit button raises the Update job schedule dialog.



- Modify the desired Schedule settings
- Click the Save button to store the modified Schedule settings

Clicking the Save button at the bottom of the Space settings dialog saves any changes made.

Deleting a Schedule

Delete	Click the Delete button for the Schedule to be deleted

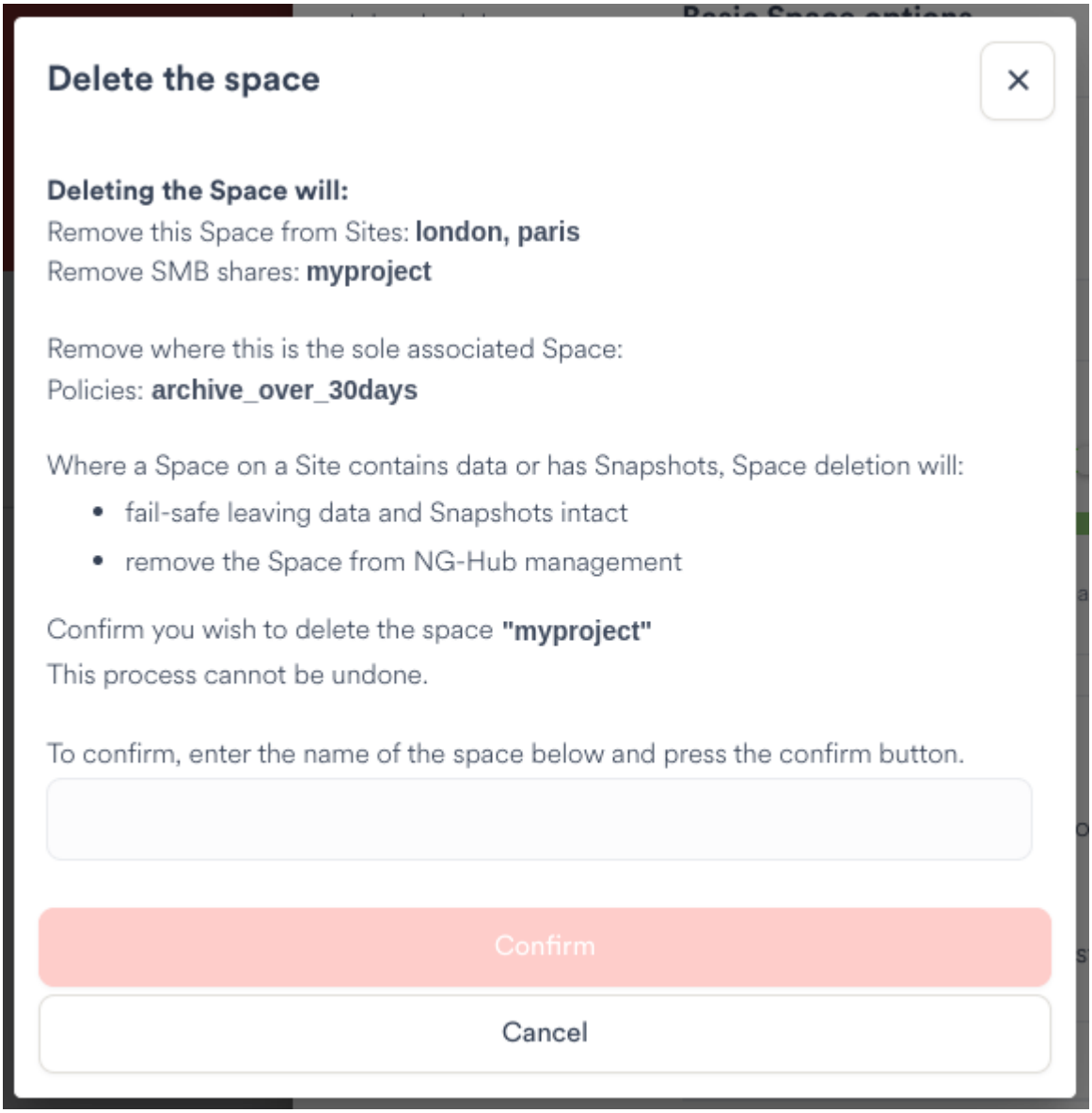
Tip: Deleted Schedules are non-recoverable. If a Schedule has been inadvertently removed, do not press the Save button, instead click off the Space settings dialog to the main area of the screen.

Clicking the Save button at the bottom of the Space settings dialog saves any changes made.

Deleting a Space

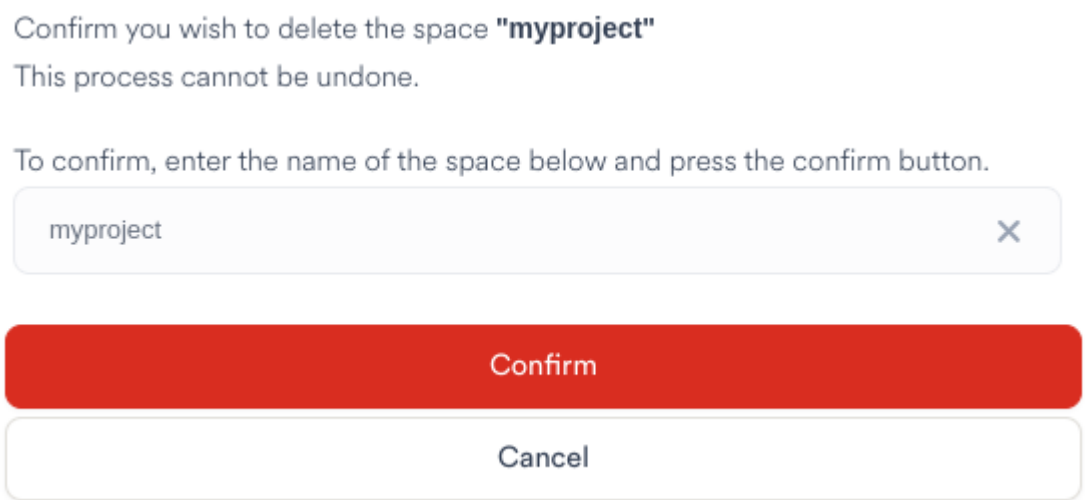
Clicking the Settings button on the Space card raises the Settings dialog for the Space.

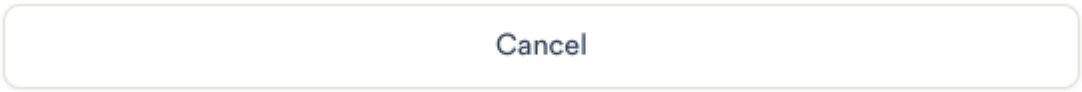
Clicking the Delete Space button raises a confirmation dialog



The delete confirmation dialog describes what will happen when the space is deleted, including what associated objects will also be removed.

Type the name of the space into the text input to activate the Confirm button.



	Click the Cancel button to close the confirmation dialog. The space will not be deleted.
	Click the Confirm button to delete the space.

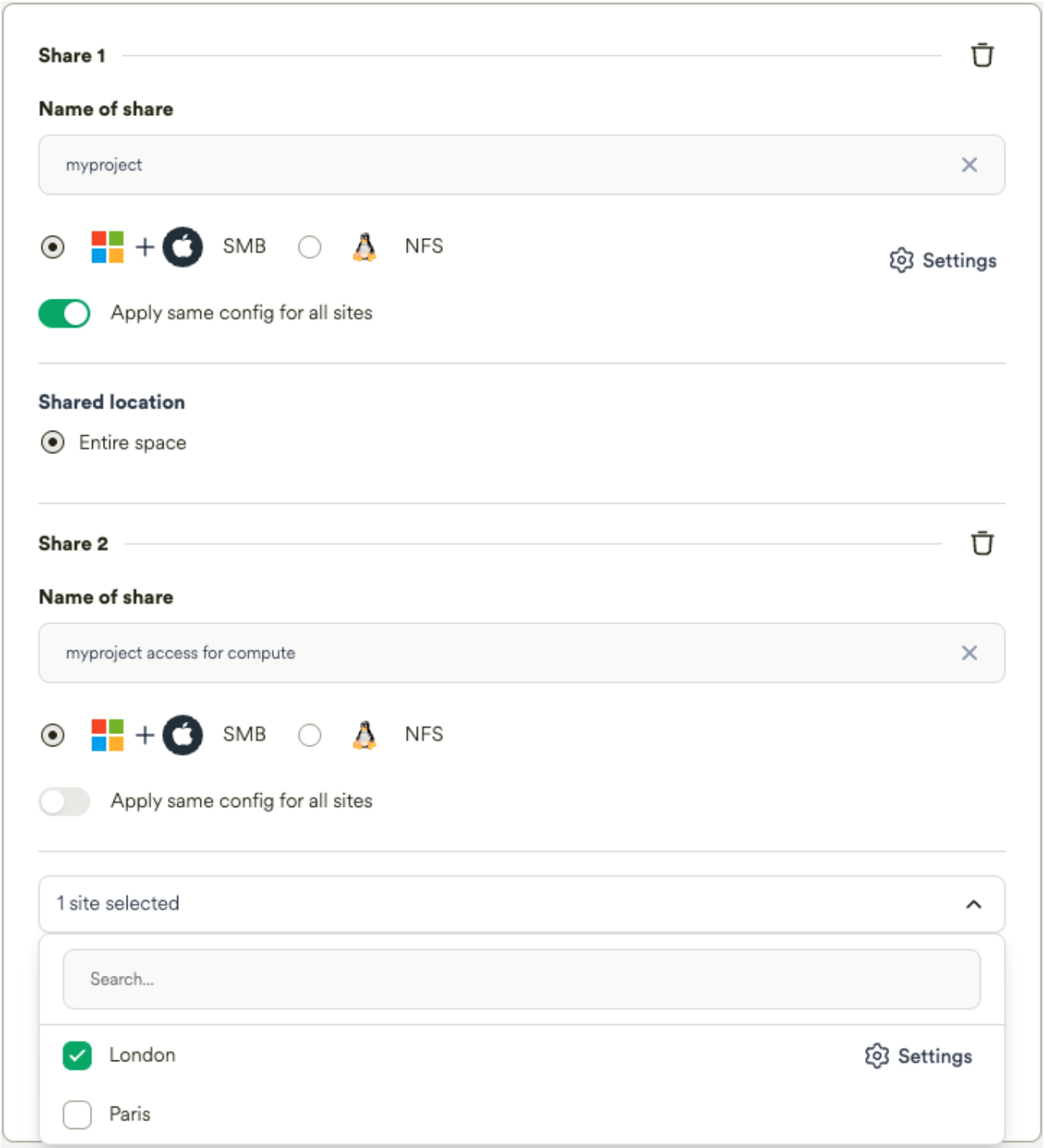
Shares

Prior to Space creation Shares can be assigned.

Shares can be created on one or more Sites where the Space is provisioned.

As the Space is not yet created any Share created is assigned to the whole Space. To create Shares within the Space, edit the Space after creation and add the required Shares after creating any subdirectories to Share.

	Shares for subdirectories within a Space can be created via the Edit Space button after successful Space creation. By default, Shares are created identically across all Sites participating in the Space.



- Select either protocol type to create a Share

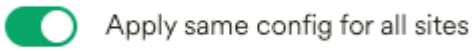
SMB protocol creates a Share with the name provided.

NFS protocol creates a Share at the Space location on the pixstor file system. Set a suitable description of the Share.

Tip: By default, a Share is created with the same configuration on all Sites participating in the Space.

Advanced Settings are accessed via the context aware Settings button.

- Choose whether to apply the Share settings to all Sites, or whether to set Shares settings per-Site



- Toggle on to define the Share configuration identically across all Sites participating in the Space

  Optionally use the Settings button to apply identical Advanced Settings across all Sites participating in the Space

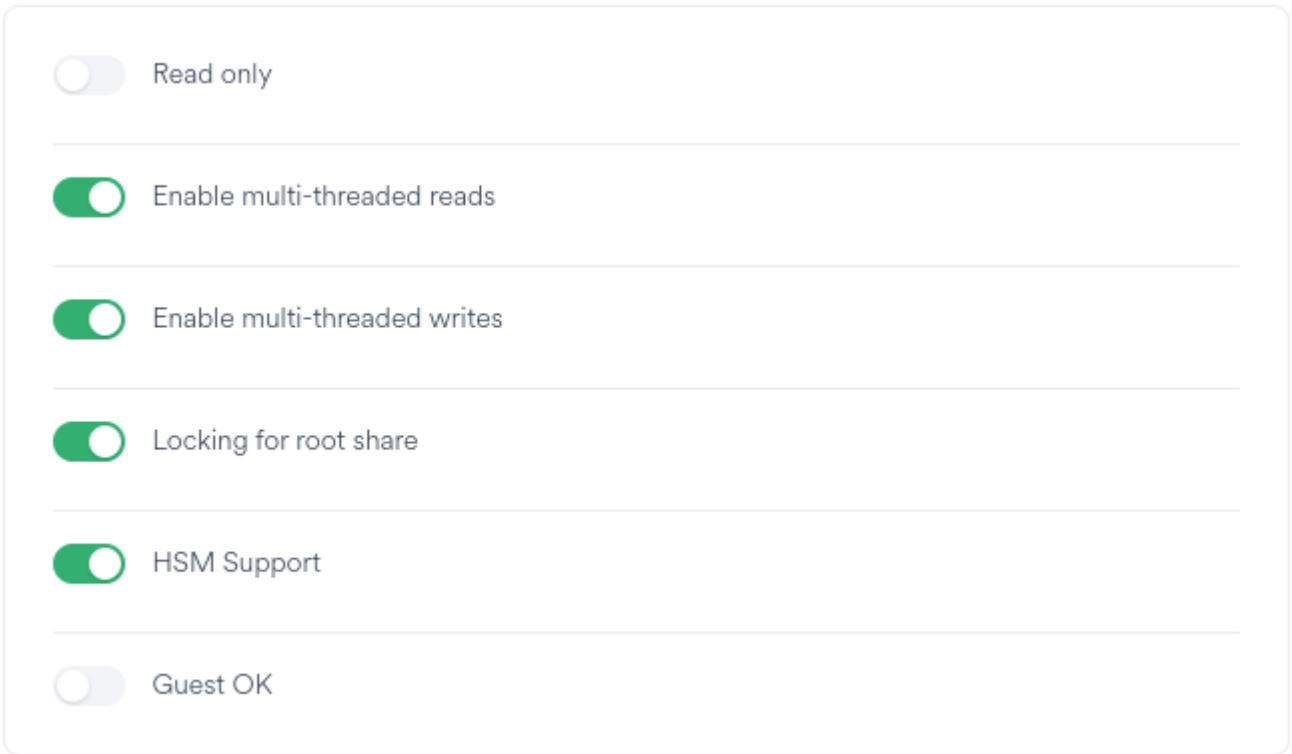
- Toggle off to define different Share configurations for specific Sites

  Optionally for each Site selected, use the Settings button to apply different Advanced Settings to each Site participating in the Space

SMB Advanced

Extended SMB options

Modifying the extended SMB options controls the Share capabilities:



Select the available extended SMB options as appropriate:

Option	Description
Read only	If enabled then users of the Share may not create or modify files in the Space.
Enable multi-threaded writes	Enable asynchronous reads and writes
Locking for root share	Enforce file locking
HSM support	Support Ngenea operations to data within the Space
Guest OK	Access will be permitted as the default guest user

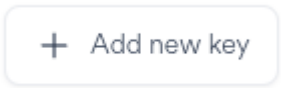
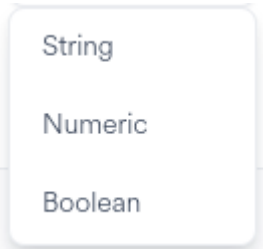
SMB Custom Options

On occasion Administrators may find it necessary to further tailor SMB Shares in order to provide additional capabilities E.G. compatibility or performance tunings.

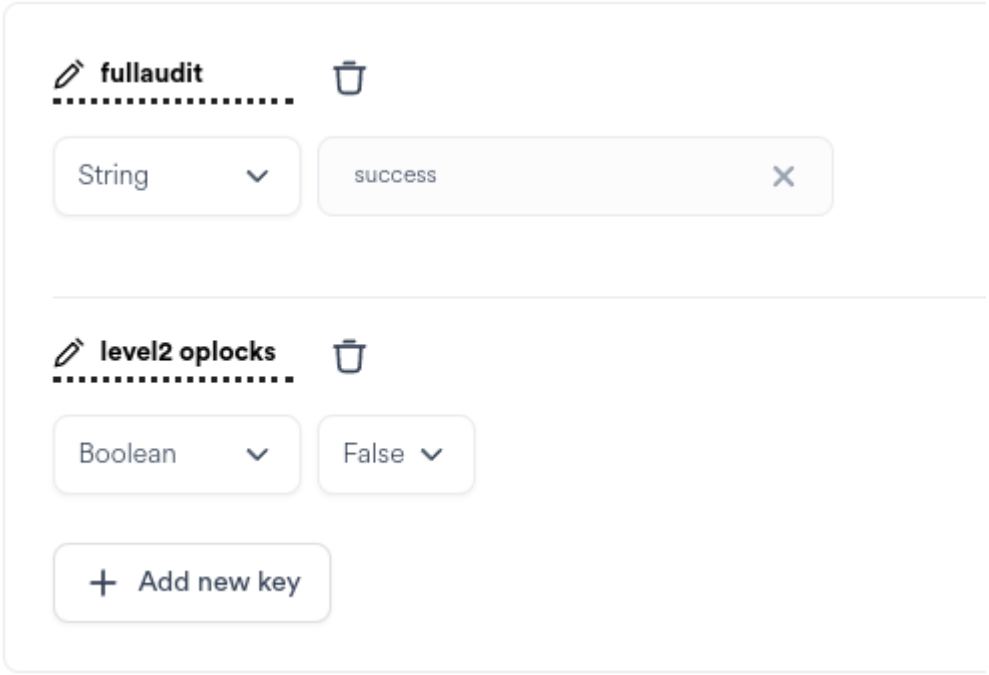
Pixstor supports additional SAMBA Custom Options per-share. Custom Options defined for a Space's Share are applied across all Sites where the Space is provisioned.

Important: SMB Custom Options are applied verbatium to the underlying SAMBA configuration. Incorrect entries can result in service outage.

Add a new Custom Option as follows:

	Click the Add new key to define a new custom option
	Enter the name of the custom option setting in the Keyword field
	Select the type of custom option from the drop down menu. Choose or enter the value for the custom option.

Example of added Custom Options:



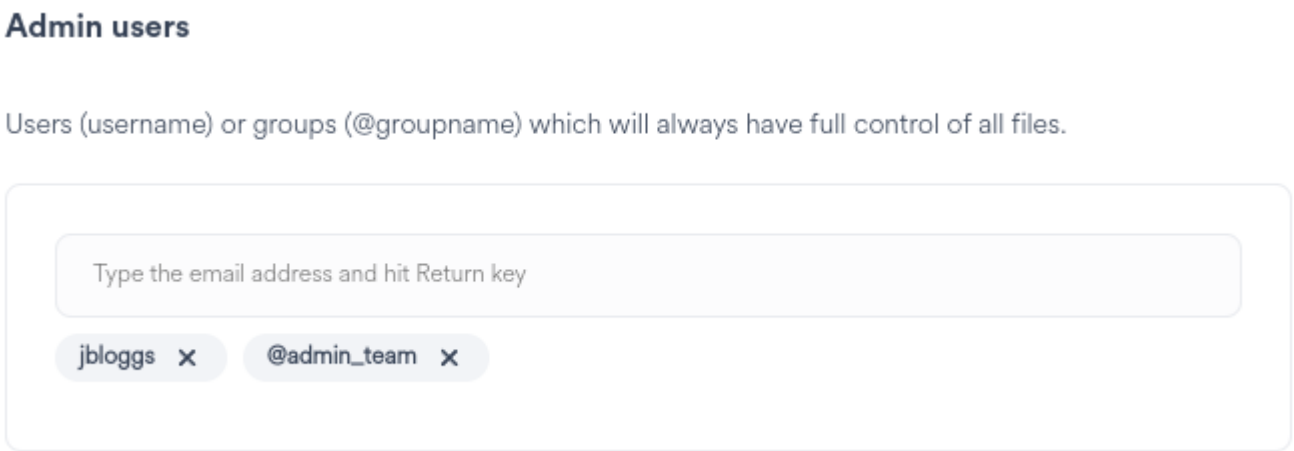
To modify an existing custom option change the keyword, type or values as appropriate.

To delete a custom option select the bin icon next to the item.

	Click the delete button to remove a custom option

Admin Users

Granted full permission to data in the Space to a specific set of users or groups:

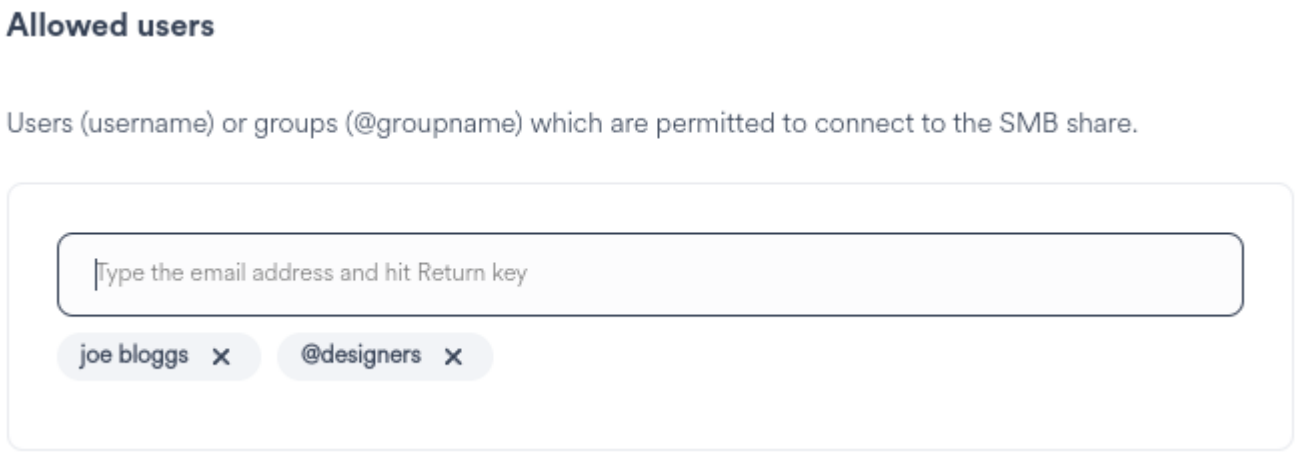


- Add users by username
- Add groups by prefixing the group name with an @ character

Hint: For users or groups containing spaces or symbols, etc. use quotes. E.G. @”Domain Admins”

Allowed Users

Restrict the access to the share to a specific set of users or groups:

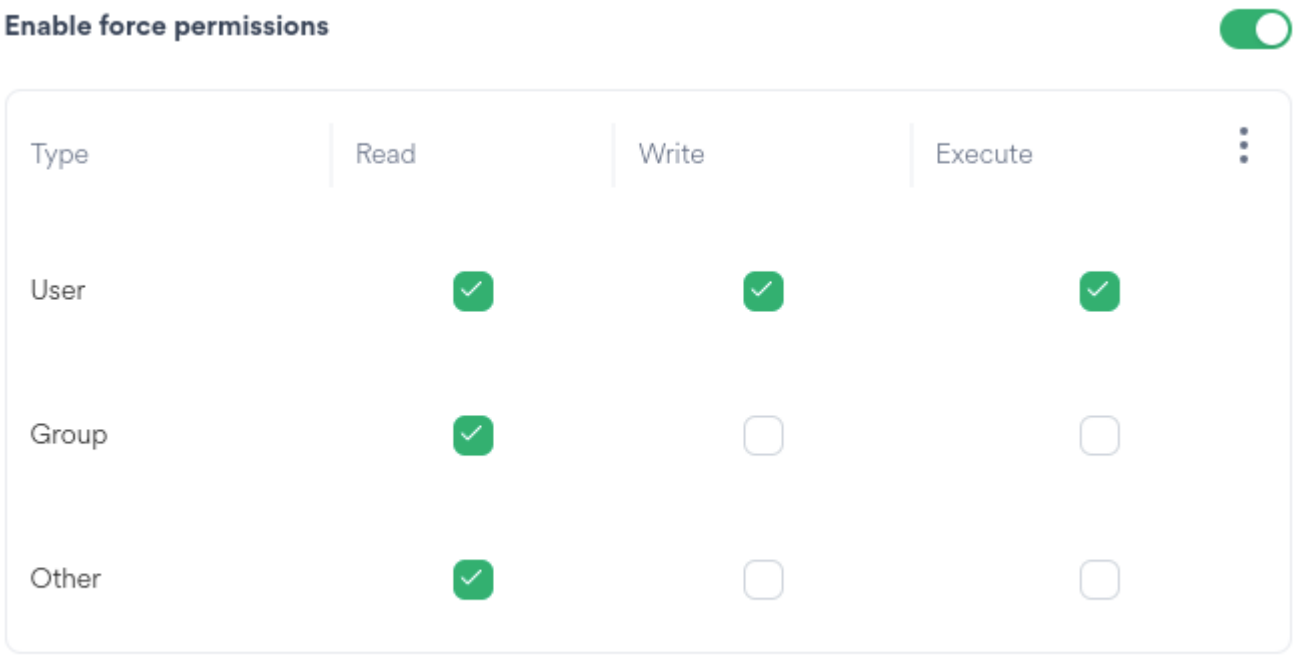


- Add users by username
- Add groups by prefixing the group name with an @ character

Hint: For users or groups containing spaces or symbols, etc. use quotes. E.G. @”Domain Admins”

Force permissions

In some scenarios it may be desirable to force the file and directory permissions to specific values in order to create a consistent known permission model. Typically this is observed where systems are not connected to Identity Management services such as Active Directory, LDAP or similar.



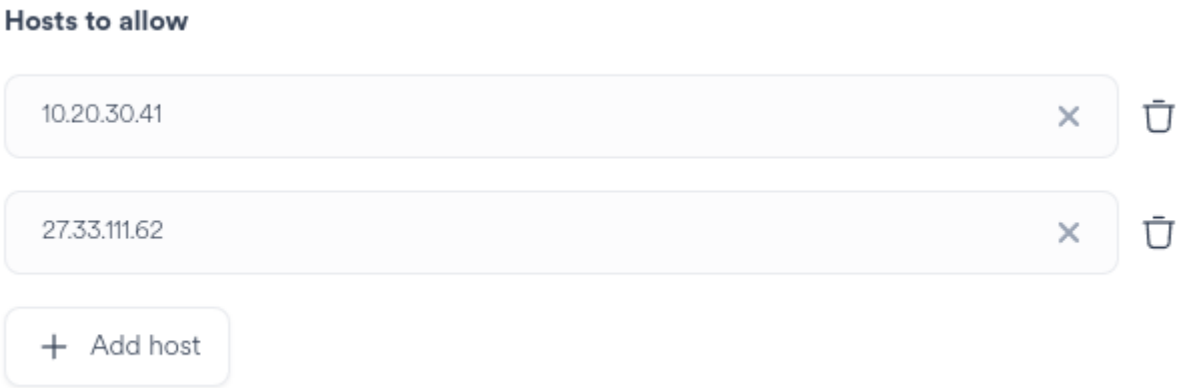
- Set the permission overrides as required

Host access control

pixstor provides the ability to limit connectivity to specific network clients.

1. If no allow or deny options are defined pixstor will allow connections from any system.
2. If only a hosts allow option is defined for a share, only the network clients listed will be allowed to use the share. All others will be denied.
3. If only a hosts deny option is defined for a share, any network client which is not listed will be able to use the share.
4. If both a hosts allow and hosts deny option are defined, a network client must appear in the hosts allow and not appear in the hosts deny to access the share.

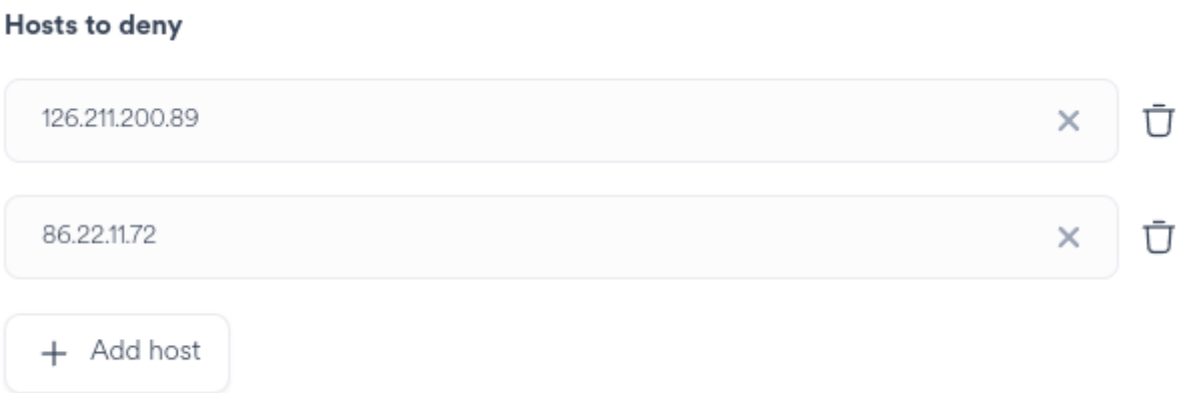
Hosts allow



- Add the required hosts by specifying the IP address

Add host	Click the Add host button to add additional hosts
	Click the delete button to remove a host

Hosts deny



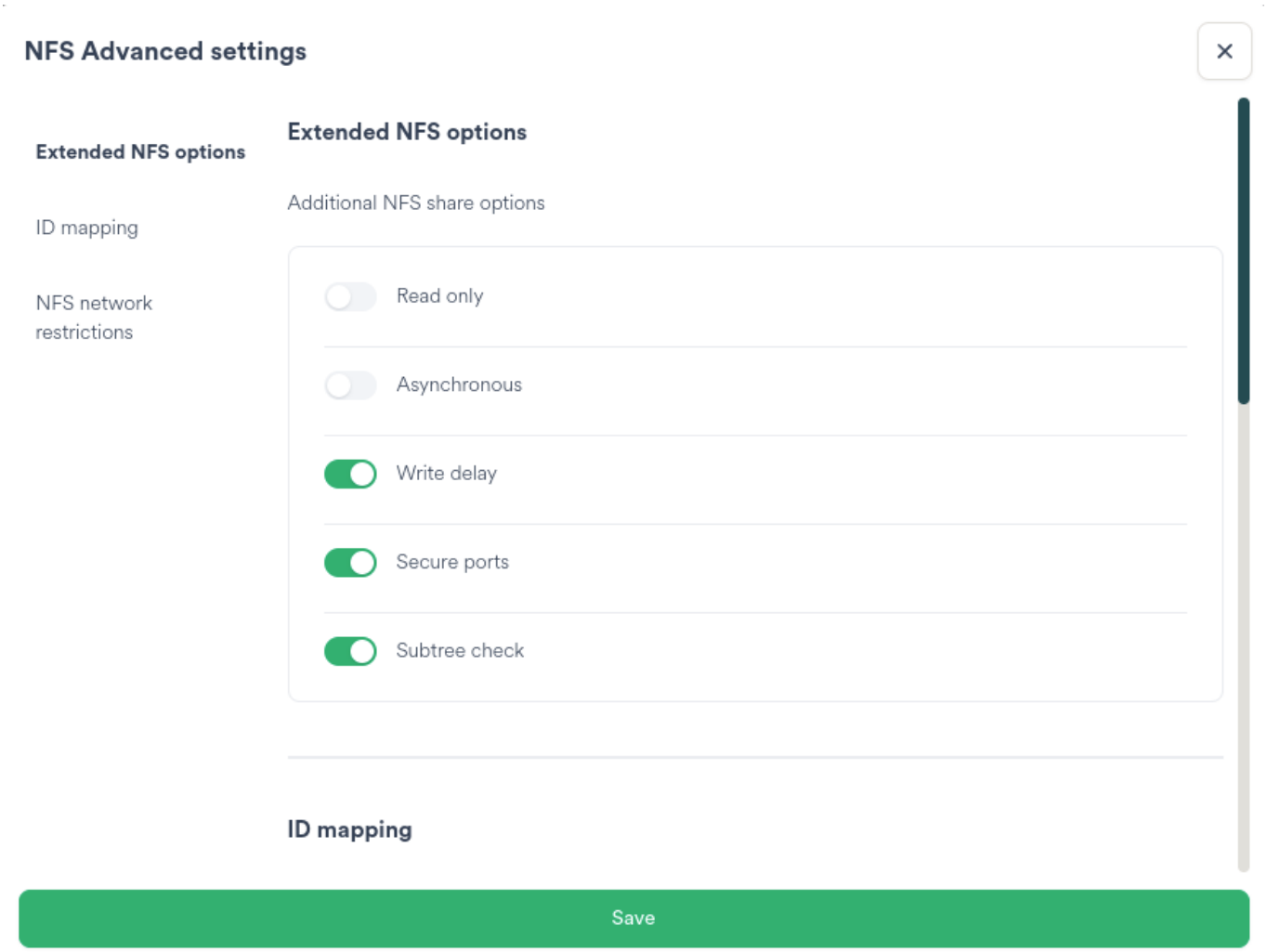
- Add the required hosts by specifying the IP address

+ Add host	Click the Add host button to add additional hosts
	Click the delete button to remove a host

NFS Advanced

Extended NFS options

Modifying the extended NFS options controls the Share capabilities:



Select the available extended SMB options as appropriate:

Option	Description
Read only	If enabled then users of the Share may not create or modify files in the Space.
Asynchronous	Enable asynchronous reads and writes
Write delay	Reply to I/O requests only after the changes have been committed to stable storage at a cost of performance reduction.

Option	Description
Secure ports	Requires that NFS requests originate from a TCP/IP port from 1-1024
Subtree check	Check the accessed file is in the appropriate filesystem and also within the Share

ID Mapping

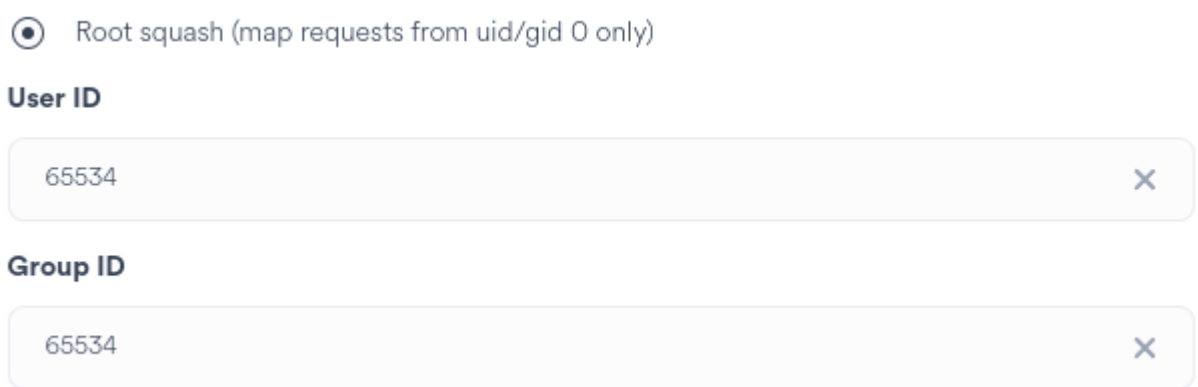
All Squash

All Squash maps all User IDs (UIDs) and group IDs (GIDs) to the anonymous user. This is useful for NFS-exported public FTP directories, news spool directories

Root Squash

Root squash allows the root user on the client to both access and create files on the NFS server as root. This is conceptually equivalent to the Adminstrator in Windows.

Root Squash is needed if you are hosting root filesystems on the NFS server (E.G. for diskless clients). You should not use no_root_squash unless you are aware of the underlying implications.



No Squash

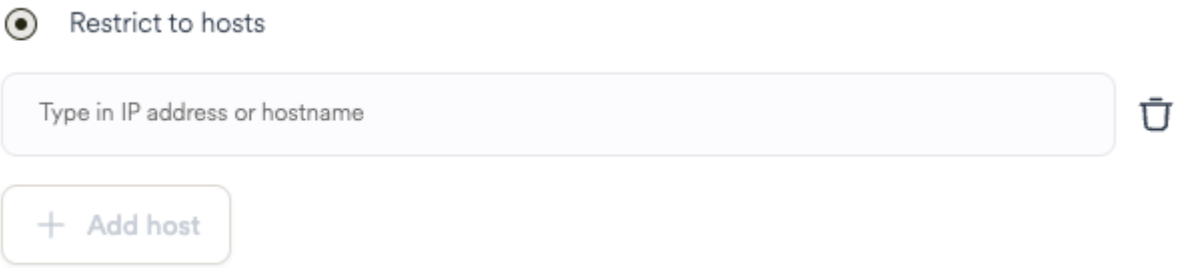
No Squash allows the root user on the NFS client host to access the NFS-mounted directory with the same rights and privileges that the superuser would normally have.

NFS network restrictions

pixstor provides the ability to limit connectivity to specific network clients.

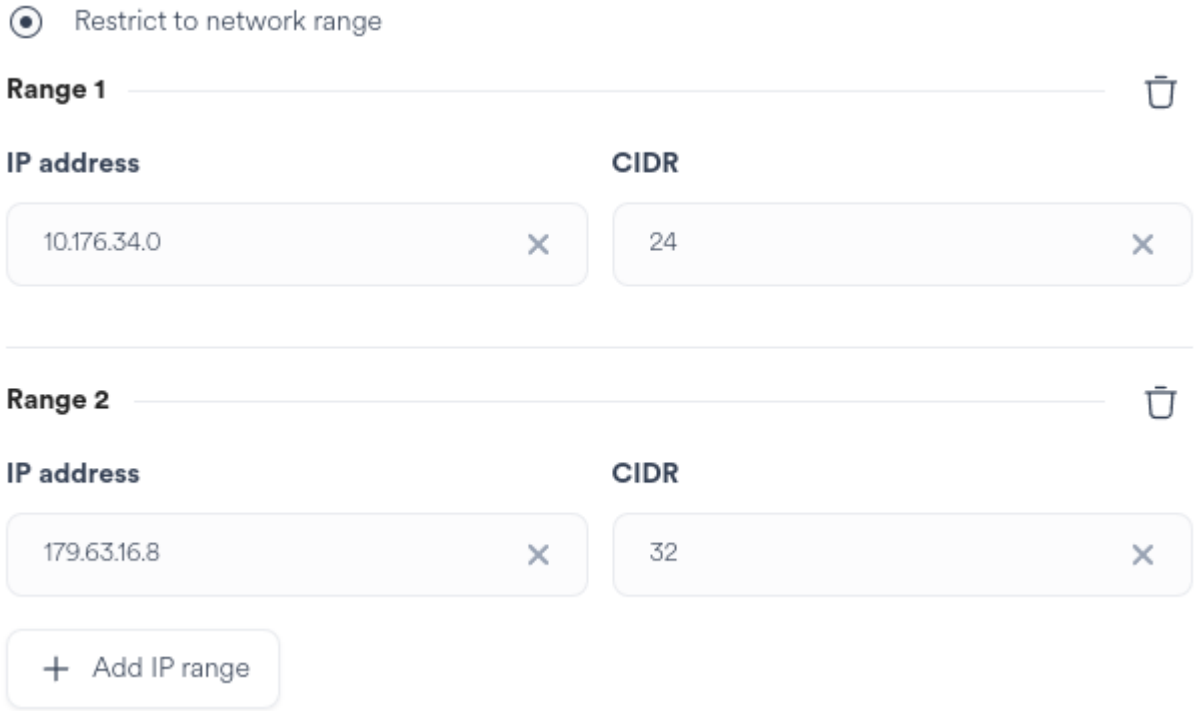
- Select the Available to all clients option for no restrictions

Restrict to hosts



- Add required hosts to restrict by specifying the IP address or the FQDN hostname

Restrict to network range



- Add the required IP address or network range and specify a valid CIDR mask to apply the restriction

Add IP range	Click the Add IP range button to add additional restrictions
	Click the delete button to remove a host

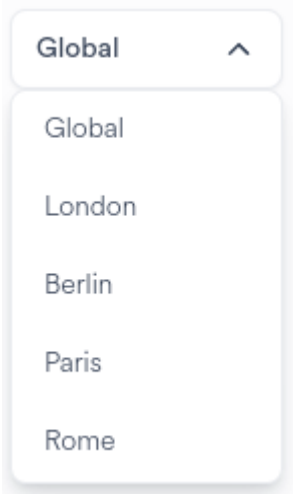
Jobs

A job comprises one or more tasks, each of which perform an action.

Tasks can be data orientated (E.G. hydrate, dehydrate, SendToSite) or can be management or configuration tasks of pixstor sites and/or services.

Viewing Jobs

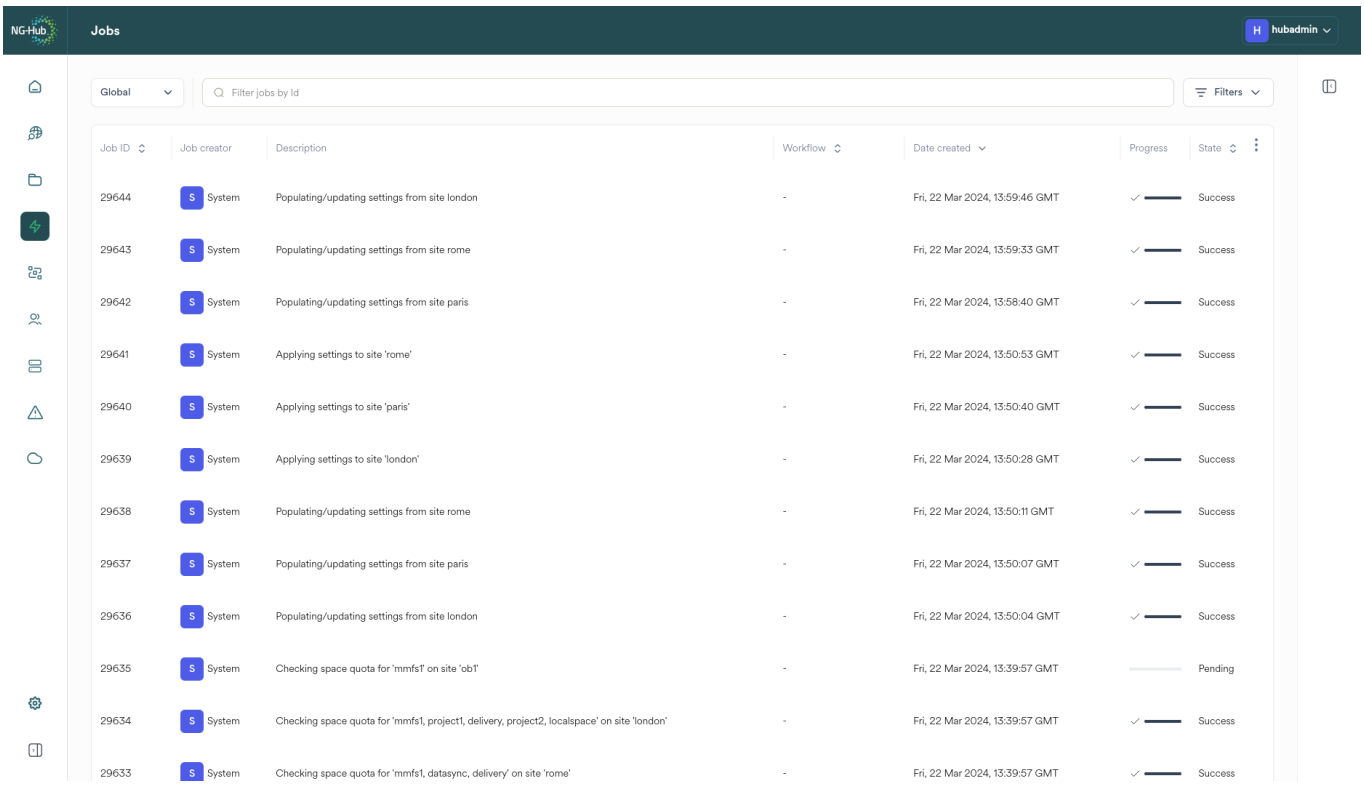
To view Jobs executed on a specific Site, select the Site from the Sites drop down menu or choose Global to display all Jobs from all Sites.



Click and select the site to show the jobs from the selected site.

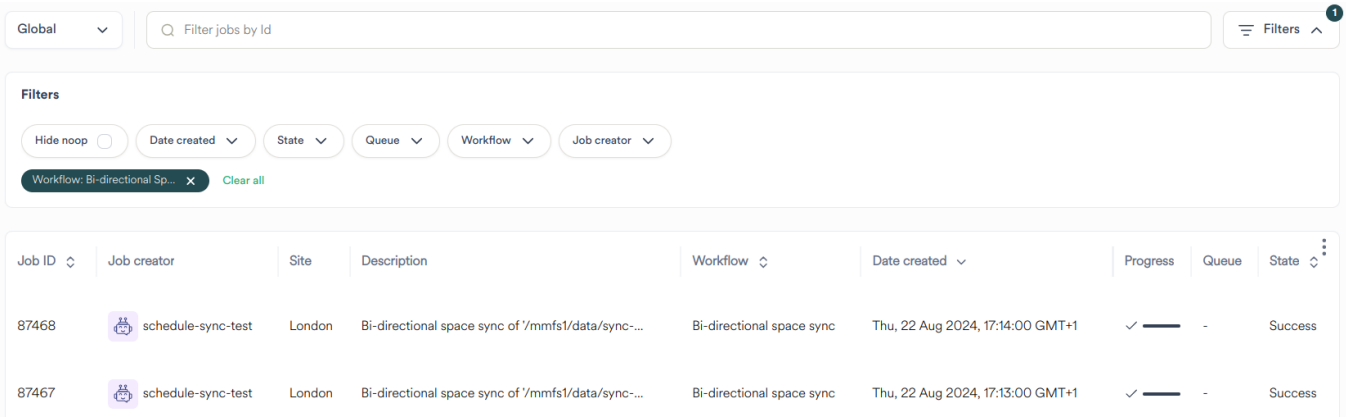
After selection the Jobs from the selected Site are displayed including high level information for each Job.

Click a Job ID to view in depth information for the job.

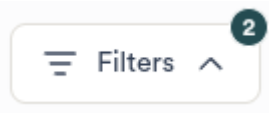


Filtering the Jobs View

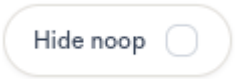
To display a specific Job ID, enter the ID in the filter bar.



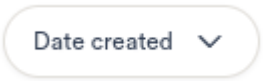
Jobs for a site can be additionally filtered through selection of various criteria.



Click the filter button to display the available filters. The number of applied filters is displayed on the Jobs filter button.

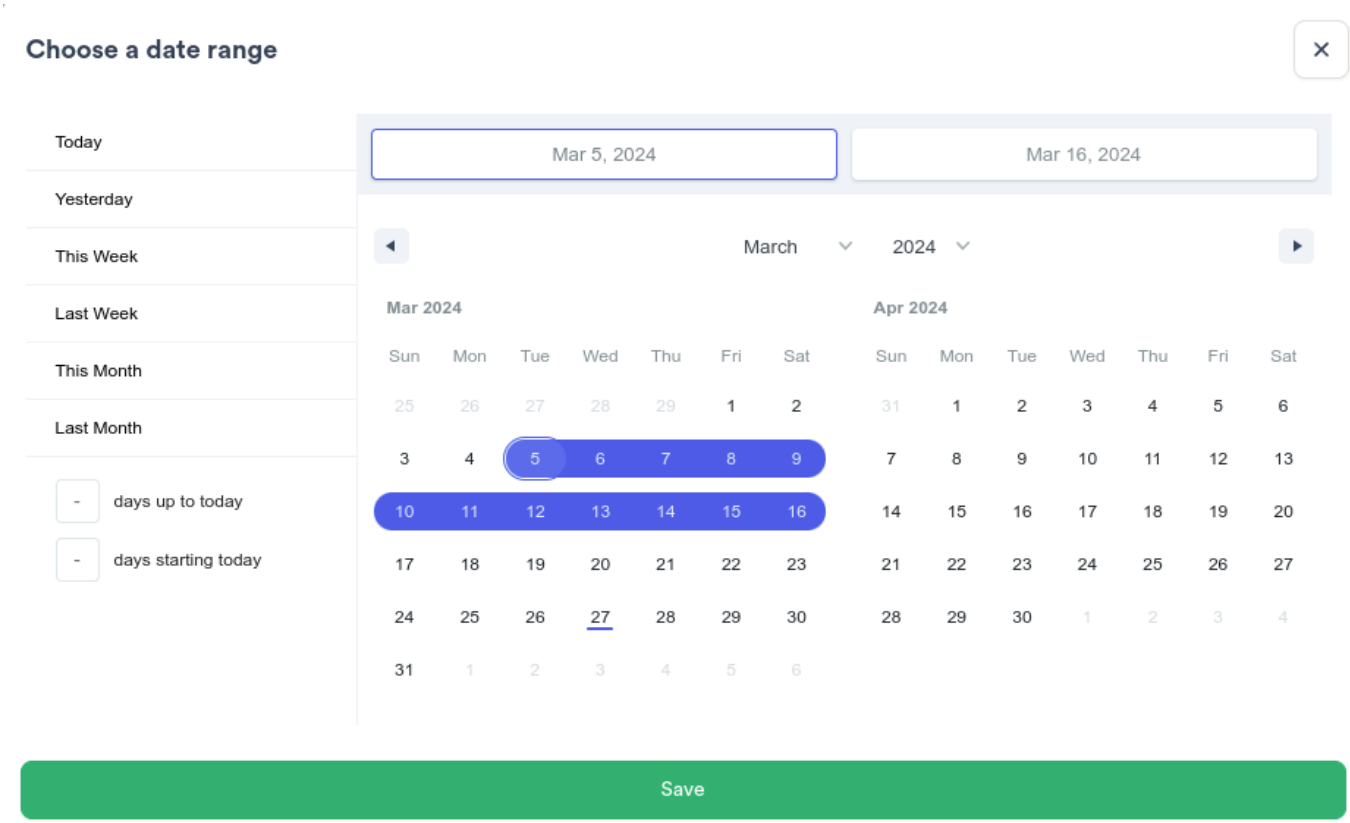


Select the Hide noop filter option to hide any job which did not perform any action as the job determined that no actions were required.



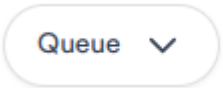
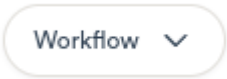
Click the Date Created button to filter for any job which was created between two date ranges.

Clicking the Date Created button raises the date selection dialog. Choose the criteria required and click the Save button to apply the date range.

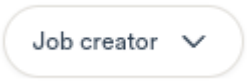


	Click the Job State button to display jobs matching the status. Available statuses are:

Status	Description
New	A job has been created
Pending	The job is waiting to run
Started	The job is running
Succeeded	The job finished successfully
Error	The job finished with one or more error conditions
Failure	The job finished with one or more failure conditions
Skipped	The job was skipped as the work assigned to the job was not required to be undertaken - no change would have occurred if the job had run.
Cancelled	The job was cancelled
Dequeuing	The job is in the process of removing tasks from the queue - Dequeuing is the process which occurs before the job is in state Cancelled.
Pausing	The job is in the process of Pausing
Unknown	The job experienced a result which could not be matched to a Status

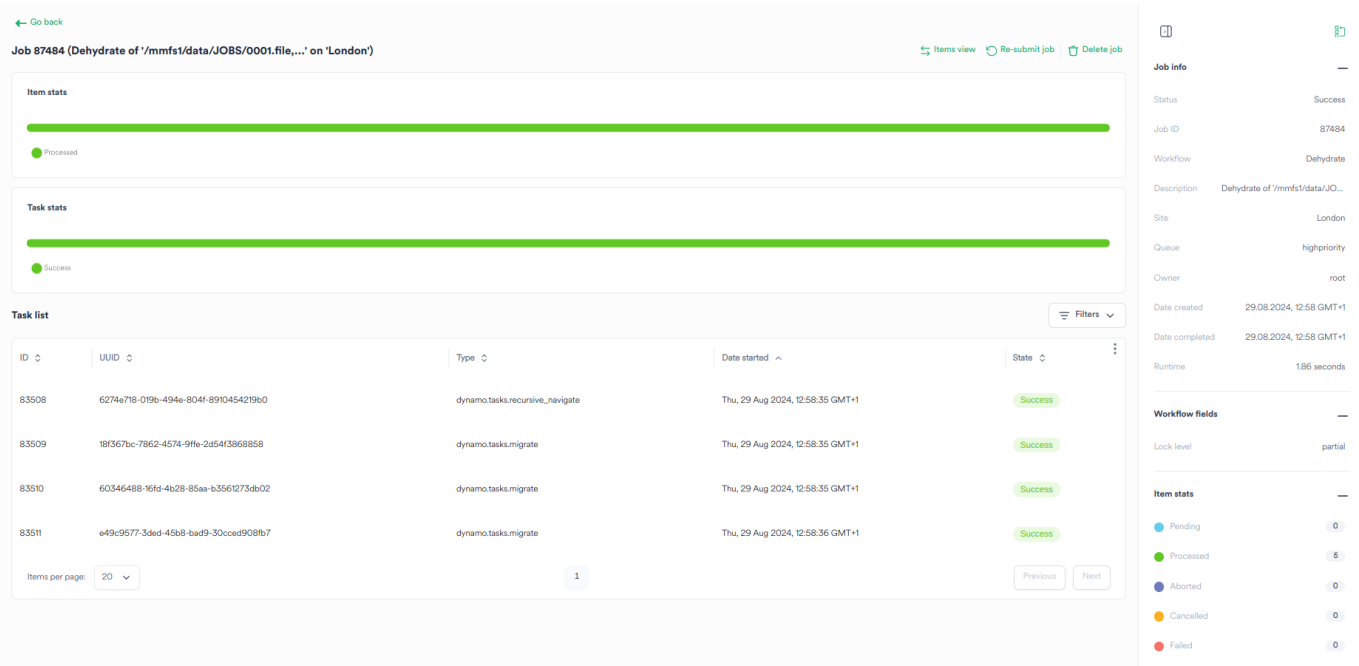
	Click the Queue button to filter for jobs matching the Queue.
	Click the Workflow button to filter for jobs matching the Workflow. Available workflows are:

Type	Description
Bi-Directional Sync	Data synchronisation between two sites
Delete File	Deletion of a designated data
Dehydrate	Dehydration of data
Pre-Stage	Staging of data to an Ngenea target. Subsequent dehydrations save time as there is no requirement to stage data prior to dehydration.
Hydrate	Hydration of data
Reverse Stub	Creation of dehydrated files not prior existing which reference data in an Ngenea target
Send	Deliver data from a source to a destination site

Type	Description
Send to Site (hydrated)	Deliver data from a source to destination site as hydrated files
Send to Site (dehydrated)	Deliver data from a source to destination site asdehydrated files
Sync Space to Site	Synchronising data from a source to a destination site
Transparent Recall	User or application initiated hydration of data on reading
	Click the Job creator button and select the name of one or more creator(s) to view jobs by the selected creator(s)

Viewing a Job

Clicking a Job ID in the main Job screen displays the in depth information for the job.



The Job View comprises:

Task stats

The total count of types operation result per task is represented by the horizontal bar segments.

Hovering over the bar provides the count of each task status for the tasks processed.

File/Item stats

The total count of types operation result per item is represented by the horizontal bar segments.

Items may be files, directories or Objects.

Hovering over the bar provides the count of each type processed.

Bi-Directional Sync and Site Sync workflows only provide File stats.

The Job Side bar

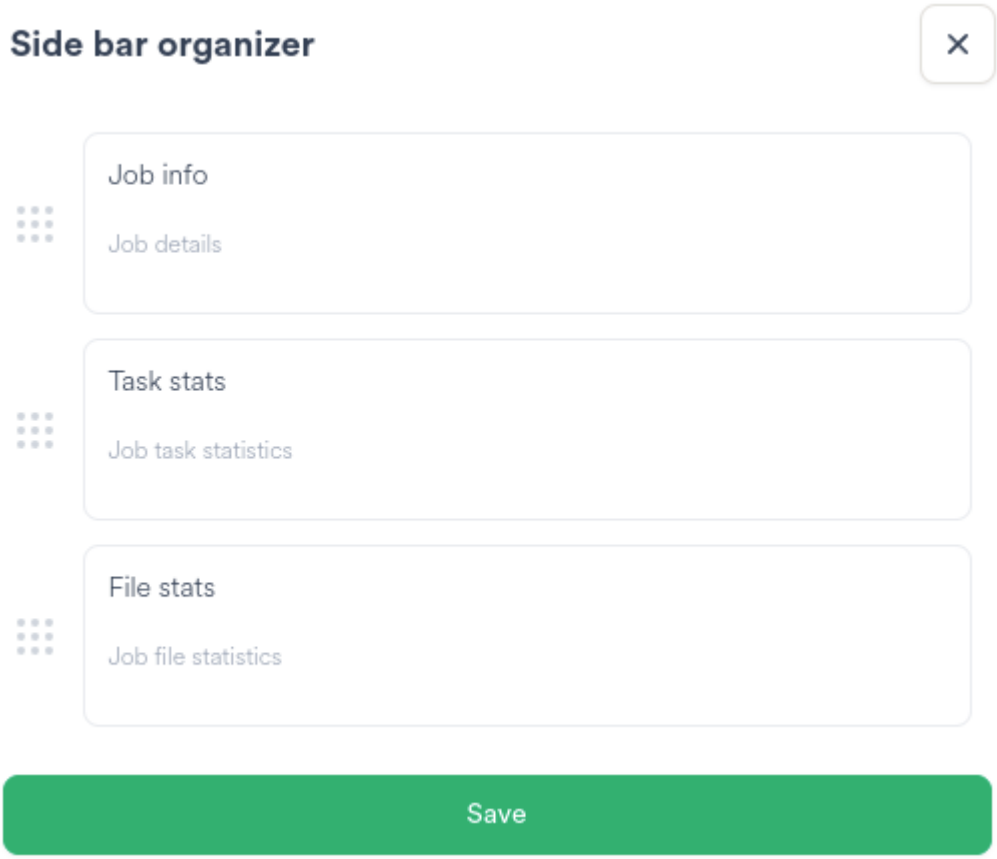
Enlarge/Reduce

	Clicking the toggle displays the Job Side bar to show: • Job info • Task stats • File Stats (Item stats)

Clicking the toggle button again hides the Job Side bar.

Reordering

Re-arrange items in top to bottom order by dragging vertically up or down.



Job Controls

Jobs provide the following controls where supported by the Workflow:

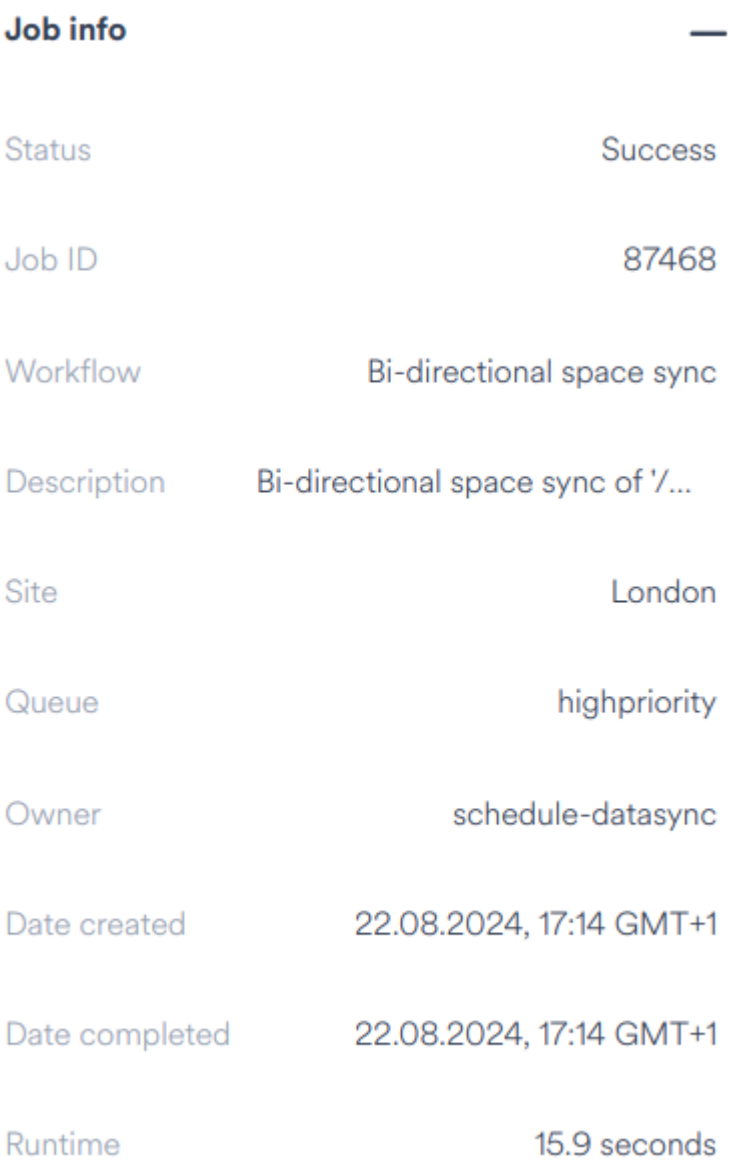
Pause job	If supported by the Workflow, click the Pause job button to pause the Job. Tasks which are not running will be paused.
Resume job	If supported by the Workflow, click the Resume job button to resume a paused Job.
Cancel job	Click the Cancel Job button to dequeue any upcoming tasks and no longer proceed with the job.
Re-submit job	Click the Re-submit job button to launch an identical Job on the Job queue
Delete job	Click the Delete Job button to remove the records of the Job and associated tasks from the Hub database

When job cancellation is initiated, the job is not terminated instantly. Instead, a de-queuing process begins to gracefully shut down the tasks. The job's state is immediately updated to Dequeuing, signifying that the system has received the request. All future tasks associated with that job are then removed from the queue. Any tasks that are currently running will be allowed to complete their execution, which means the final cancellation will take time. This design prevents a cancelled

job from blocking other jobs and ensures the system’s overall efficiency is maintained.

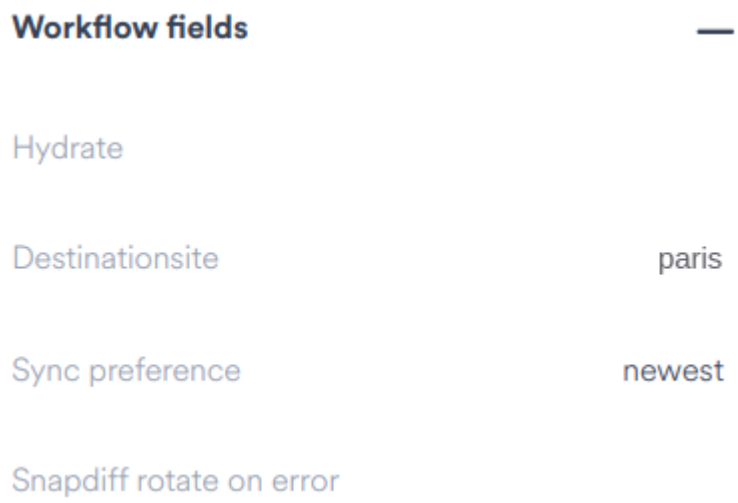
Job Info Summary

Provides an overview of high level information of the Job.



Job Workflow Fields

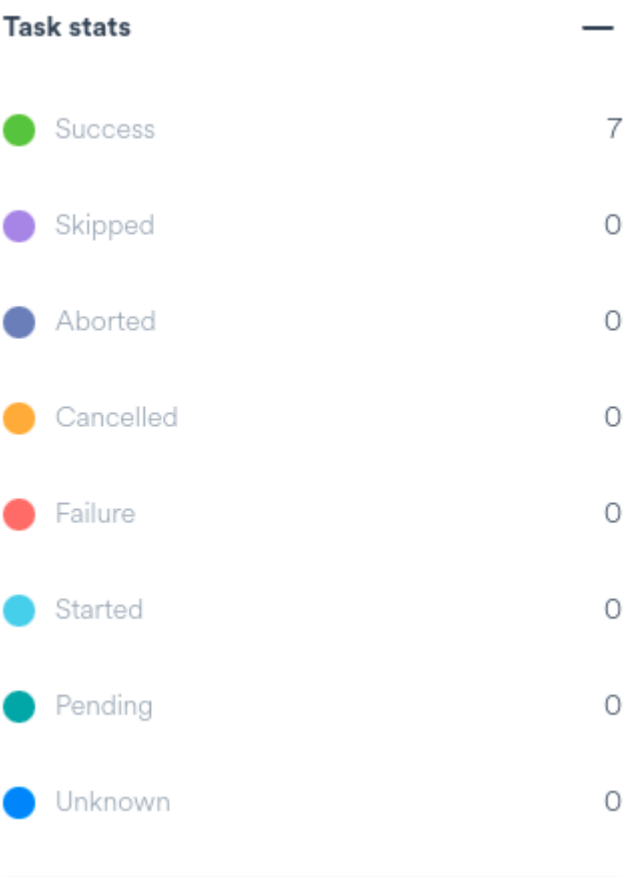
Provides a listing of any workflow fields the job was executed with.



Job Info Task Stats

Jobs undertake tasks to process one or more actions, files, directories or objects as defined by the Workflow.

The Task stats displays the total count of types operation result per task.

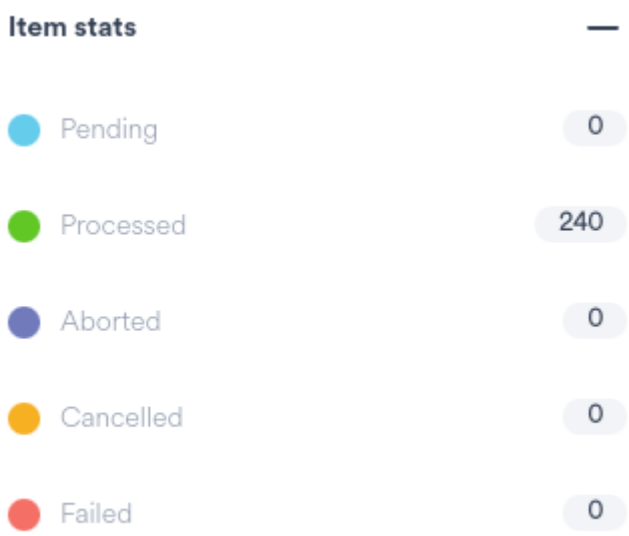


Item stats

The total count of types operation result per item is represented by the horizontal bar segments.

Items may be files, directories or Objects.

Hovering over the bar provides the count of each operation type for the files processed.



File stats

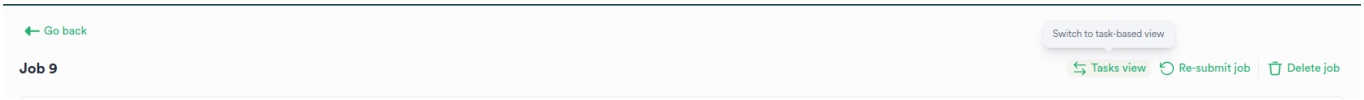
Bi-Directional Sync and Site Sync workflows only provide File stats.

The File stats displays the total count of types operation result per file or directory.

The Job Task List

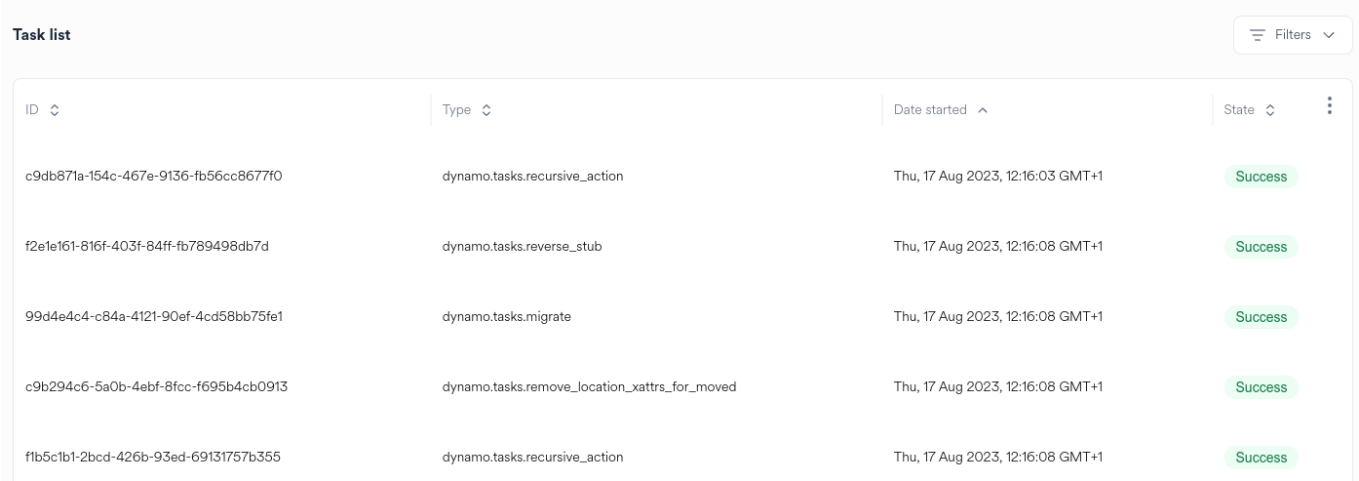
The Job details page is viewed with a `task-based view` or with a `item-based view`.

Switching between views can be done by clicking `Tasks view` or `Items view` on the top right corner of Job Tasks page.



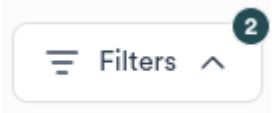
Tasks View

The Task view displays all tasks comprising a job, their ID, type, start time and status.

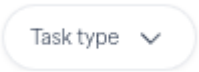


Filtering the Job Task List

Tasks for a job can be additionally filtered through selection of various criteria.

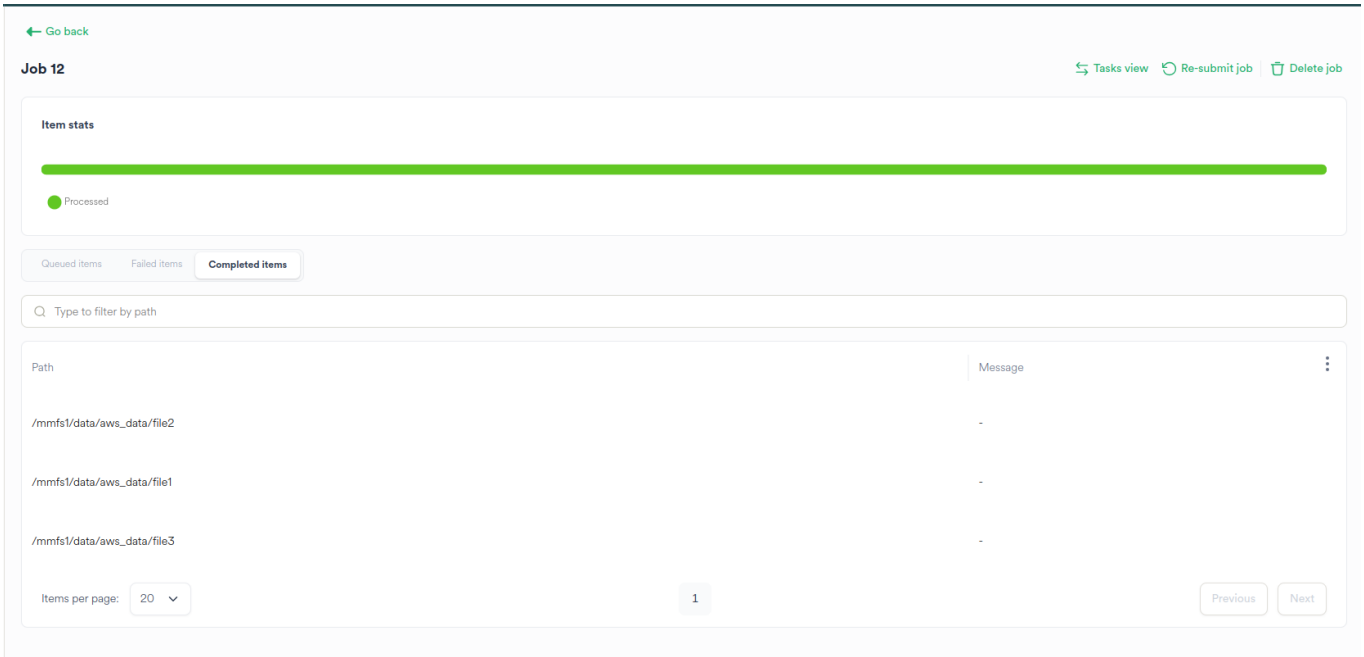
	Click the filter button to display the available filters. The number of applied filters is displayed on the Task list filter button.
	Click the State button to display jobs matching the status. Available statuses are:

Status	Description
New	A task has been created
Pending	The task is waiting to run
Started	The task is running
Succeeded	The task finished successfully
Error	The task finished with one or more error conditions
Failure	The task finished with one or more failure conditions
Skipped	The task was skipped as the work assigned to the job was not required to be undertaken - no change would have occurred if the job had run.

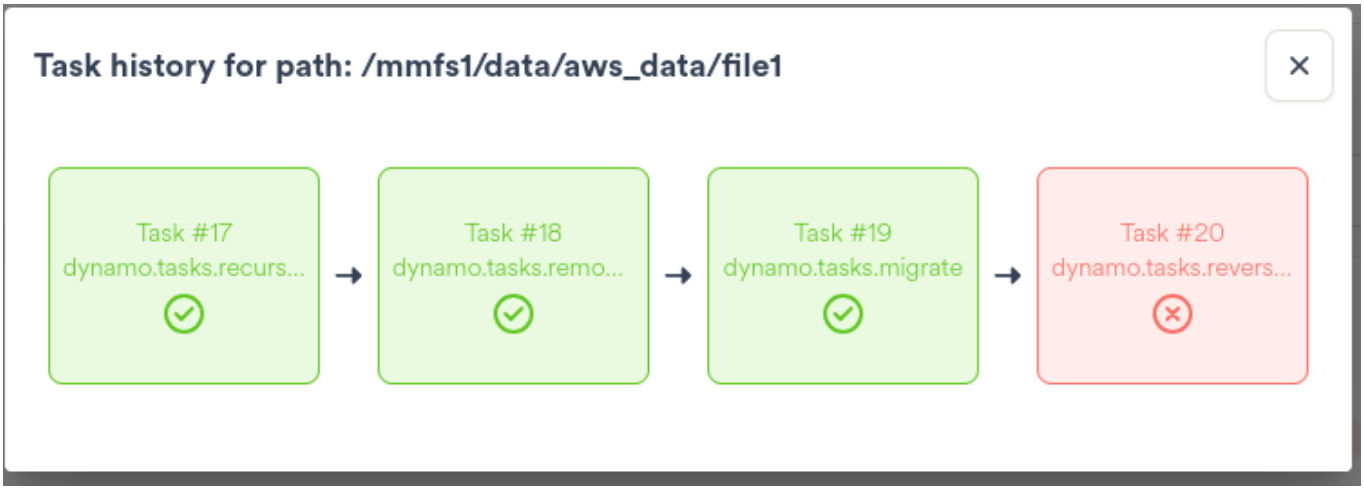
Status	Description
Cancelled	The task was cancelled
Dequeuing	The job has been manually cancelled via request and is in the process of removing tasks from the queue
Aborted	Whilst running, the task was unable to proceed correctly and aborted the in progress action
Paused	The task is paused
Pausing	The task is in the process of pausing
Unknown	The task experienced a result which could not be matched to a Status
	Click the Task type button to filter for a specific task type which comprises the Job. The task types are dynamic therefore the displayed types may differ per job type.

Items View

The Item view displays the object paths for all tasks across the job. When the `Failed items` tab is selected, the error message is displayed for failed items.



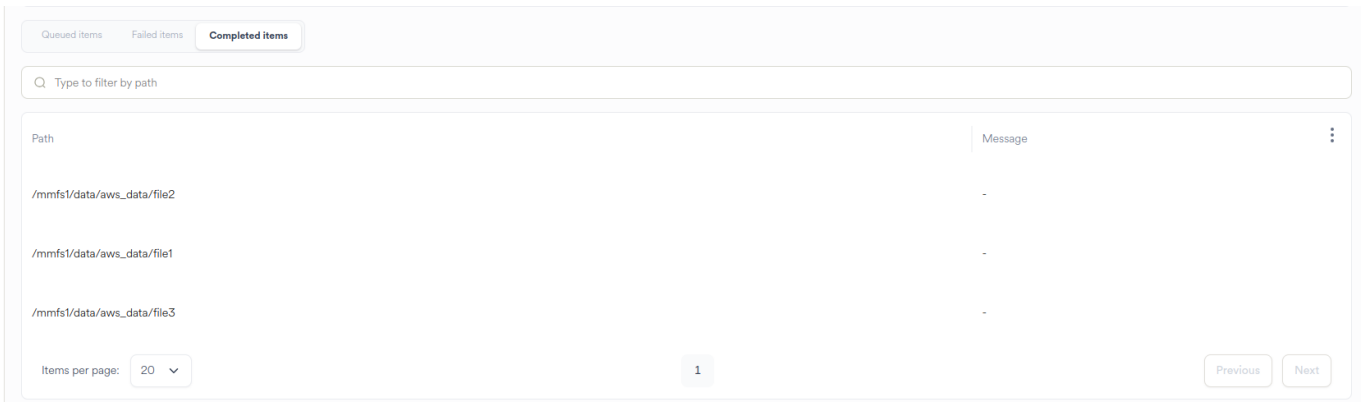
Clicking on a path displays the task history across the job.



Clicking on each task in task history displays the task details for the chosen task ID.

Filtering the Items List

To search for a specified item, enter the path in the filter bar. It will only search paths based on the state of the active tab out of the `Queued/Failed/Completed` options.



Task information Dialog

Clicking a task ID in the Job Task List displays the information for the chosen task.

Dependent on the number of operations and the quantity of inputs to the task, the information displayed can range from short to extensive.

Optionally select an action button to more easily view the Task Information.

	Click the Full Screen button to display the Task Information in a larger view
	Click the Copy button to copy the Task Information output to the clipboard.
	Click the Download button to download the Task Information locally
	Click the Close button to close the Task Information dialog

Settings Task Jobs

All settings tasks are shown as submitted by System. This includes tasks triggered automatically in the background (like for shares and spaces), where the system doesn’t have information about the user who started the action.

Schedules

Hub can regularly perform the same action to a Space via Job Schedules.

Spaces which are deployed with Bi-Directional Sync automatically have a Schedule added.

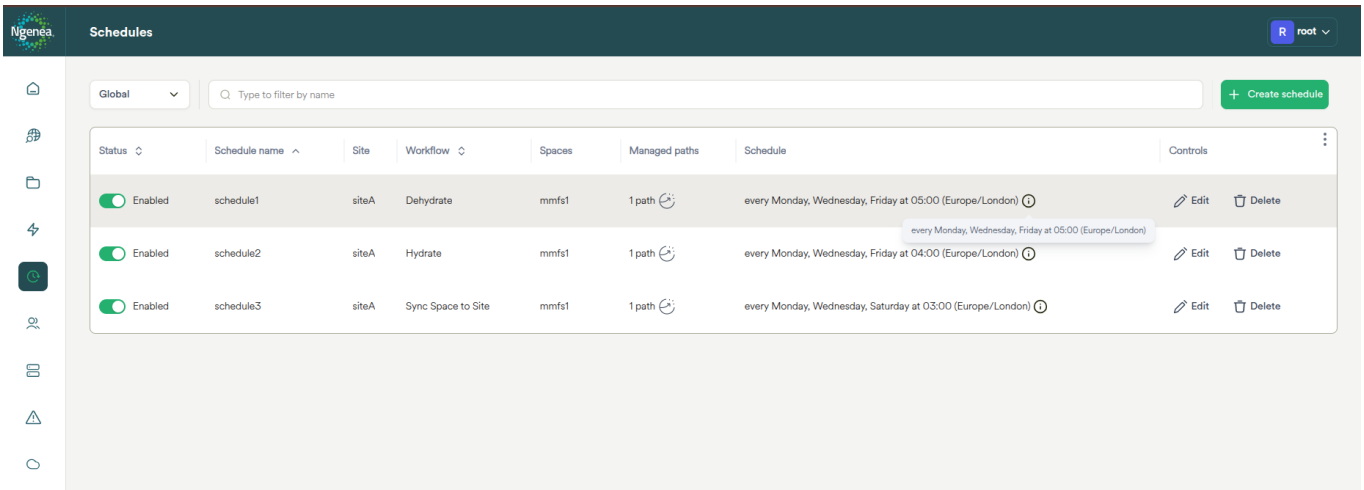
Schedules can be added to the entirety of a Space. It is also possible to schedule more granular activities within a Space for some workflow types, E.G. regular Hydration of an individual file or directory.

`Policy-based Tiering` has different configuration options to other workflow types. See the Policies page for more details.

Viewing Schedules

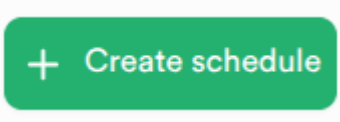
To view available Schedules, click the Schedules menu button.

Navigates to the Schedules screen



Adding a Schedule

Click the Create Schedule button to display a dialog to configure a Schedule.

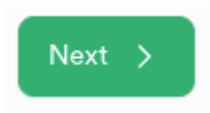
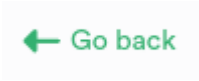
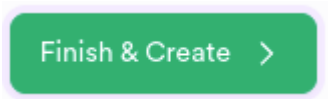


Important: This function can only be performed by a Hub Administrator

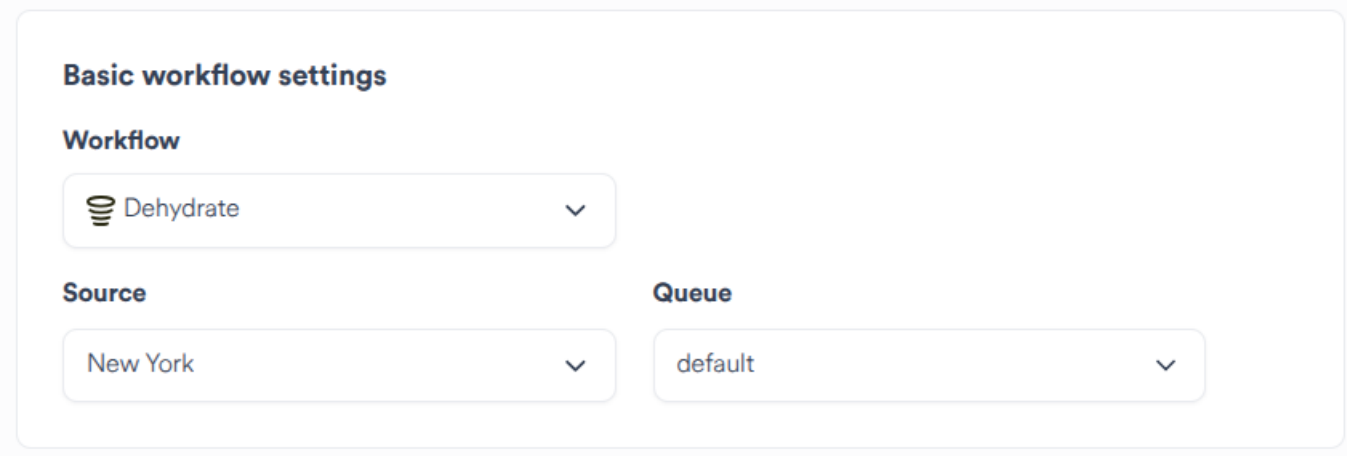
Creating a Schedule Wizard

Navigating the Wizard

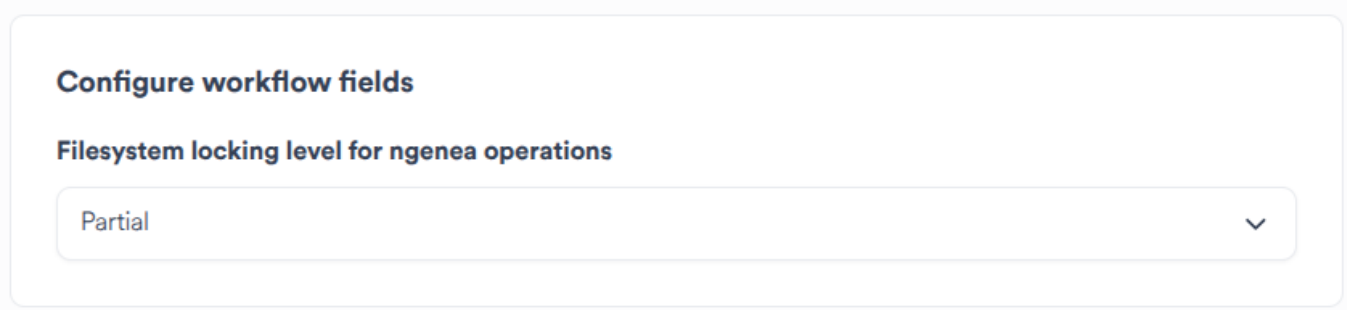
Click the close button to exit the wizard. Changes are not saved.

	Click the Next button to advance to the next page of the wizard. The Next button is disabled until all required page elements are completed.
	Click the Go Back button to return to the previous wizard page.
	Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Workflow



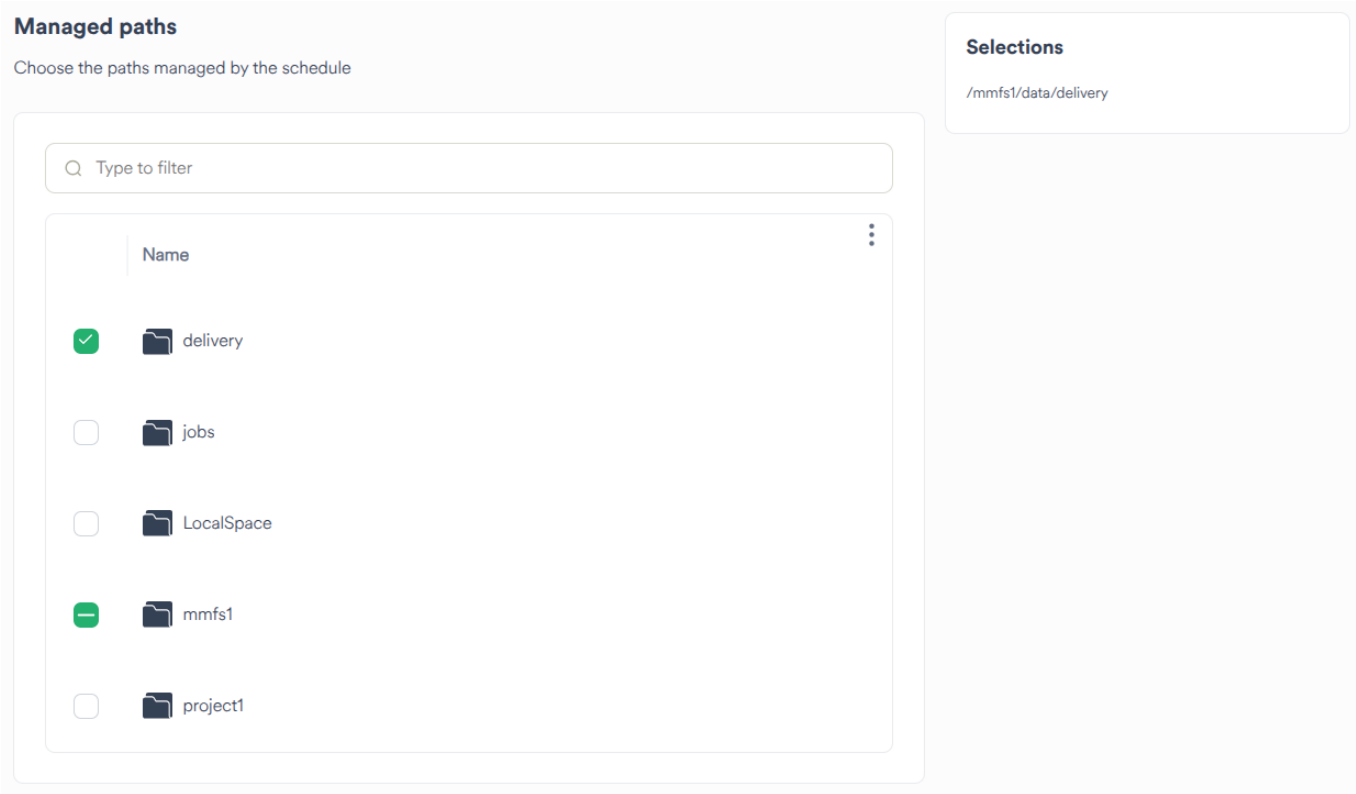
- Set the desired workflow type
- Select the Site and Queue the workflow will run on



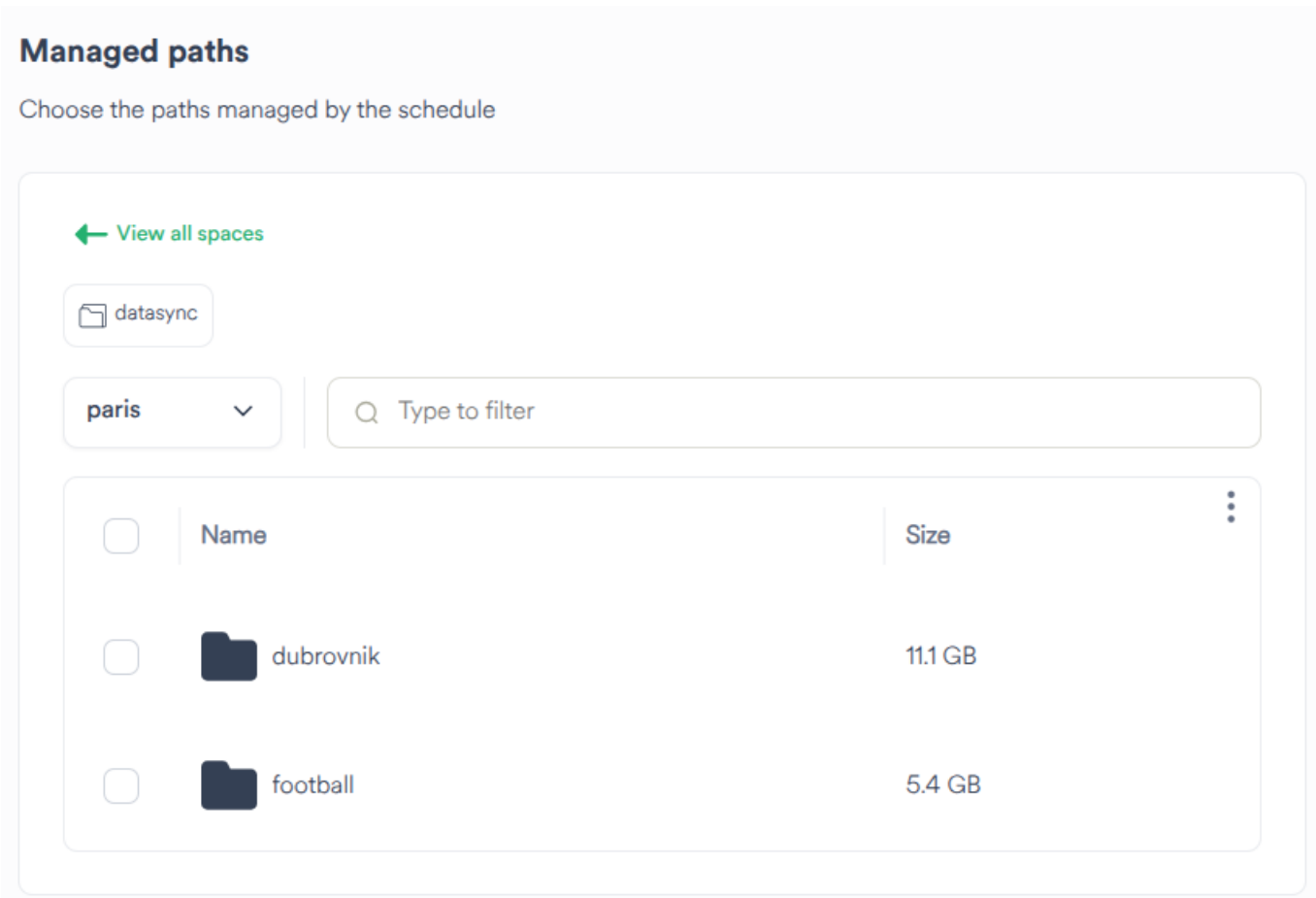
- Configure workflow-specific fields (if applicable)

Managed paths

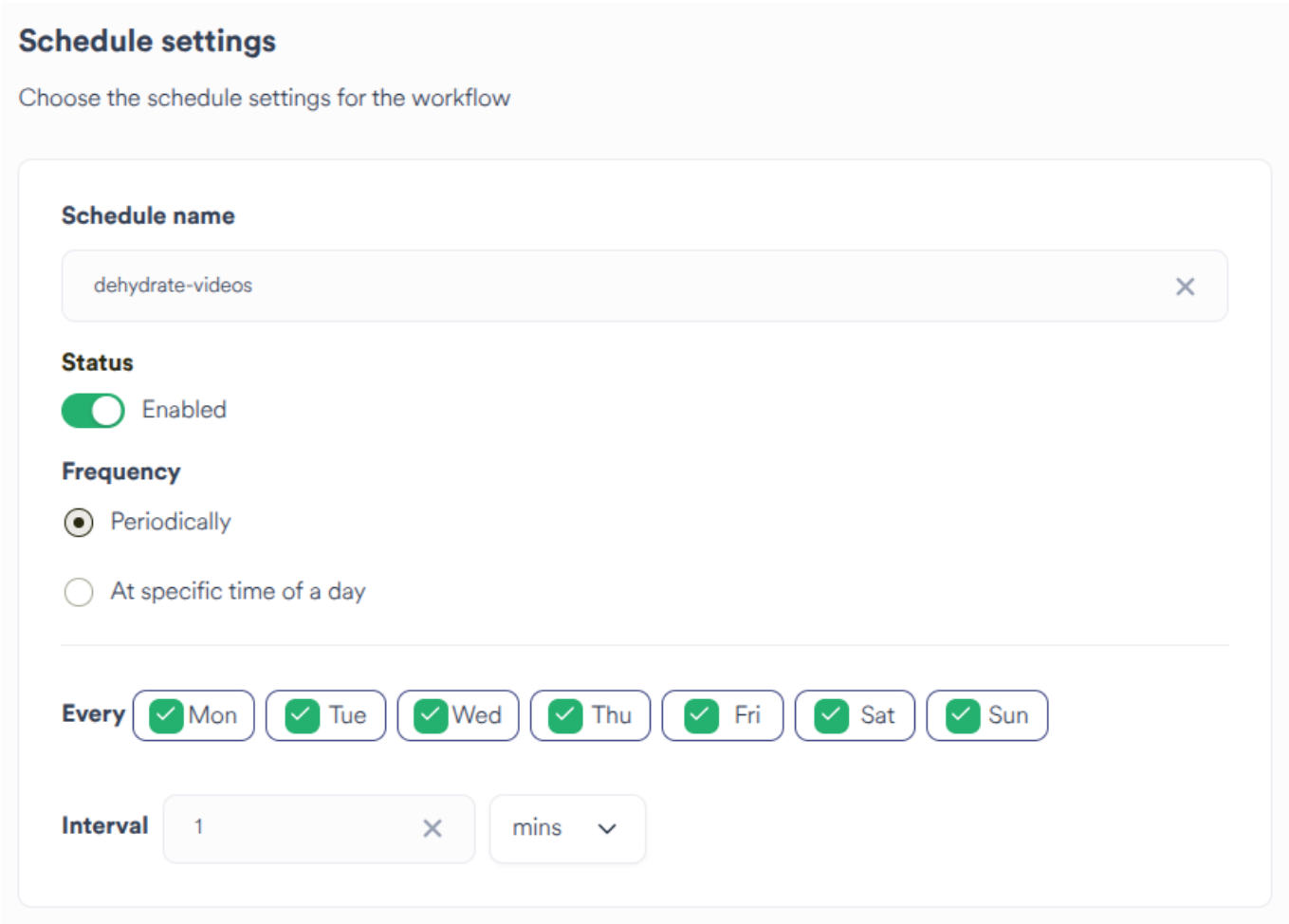
Select paths which the schedule should manage



Initially, a list of spaces is displayed.



Click a space name to view and select space contents.



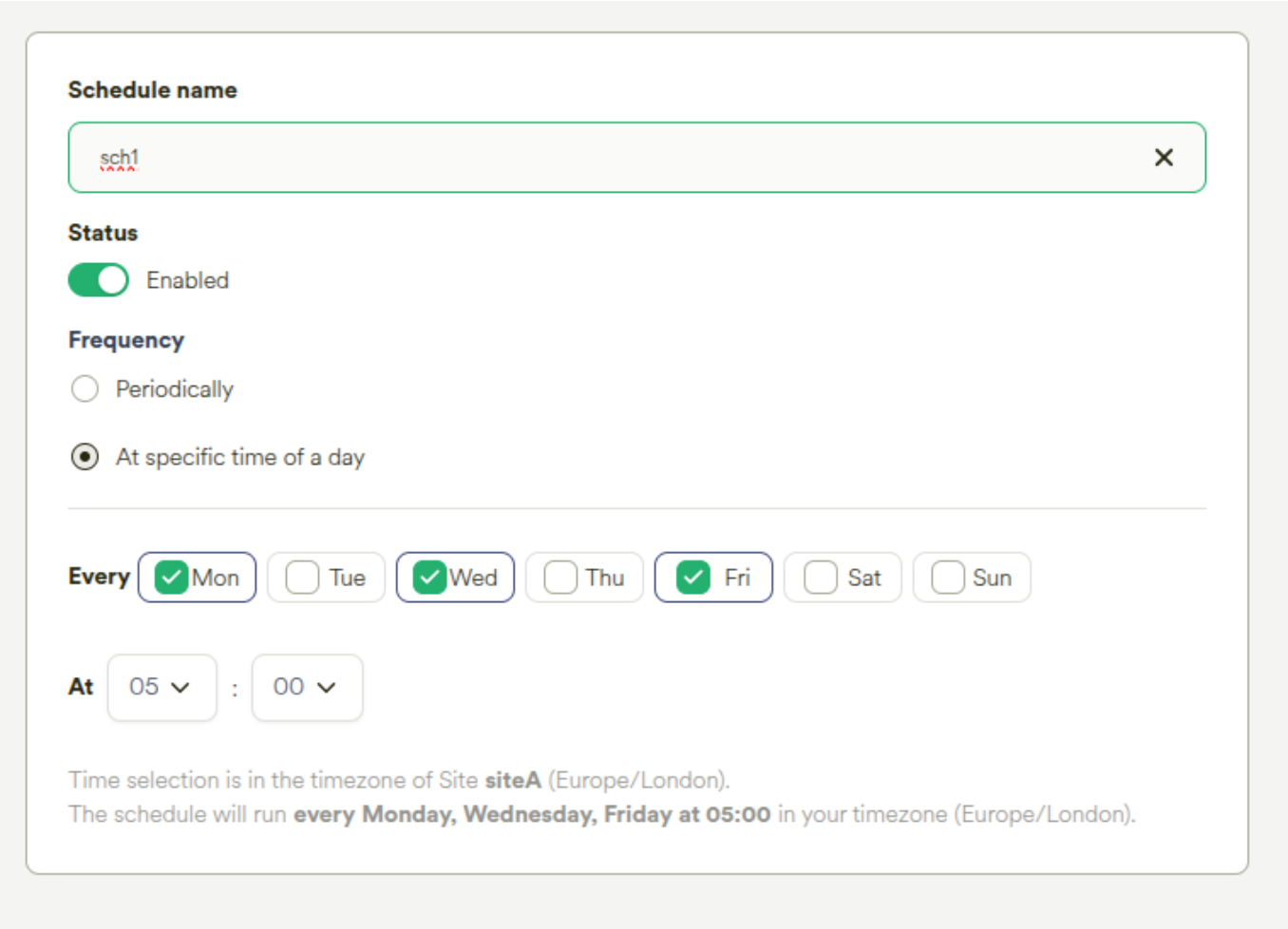
Give the Schedule a descriptive name

Determine the required frequency of the Scheduled run.

Choosing Periodically will ensure that the Schedule will run on the next interval set.

E.G:

- 1 hour: The Schedule will run on the next hour (12.00, 13.00)
- 15 mins: The Schedule will run on the next 15 minute interval past the hour (15, 30, 45, 00)



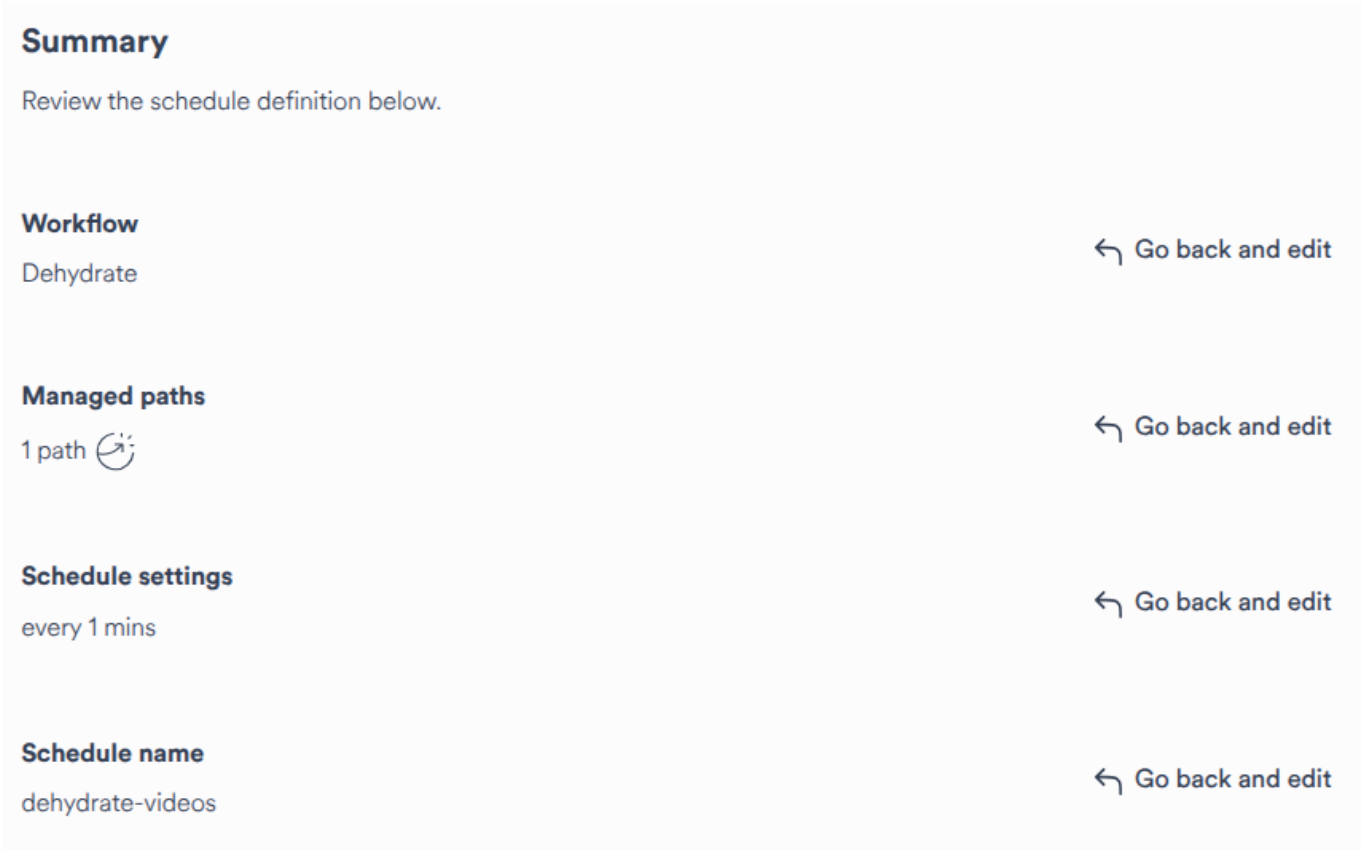
Choosing At Specific Time of A Day allows the Schedule to run once per chosen day at a specific time of day.

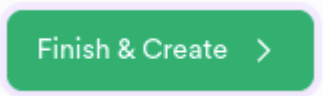
Hint: The schedule time is set in the Site's local timezone, but will be stored in the UTC timezone.

The schedule may be disabled at the point of creation by switching the `Enabled` toggle to `Disabled`.

Summary

Upon completing the wizard steps a summary is presented:





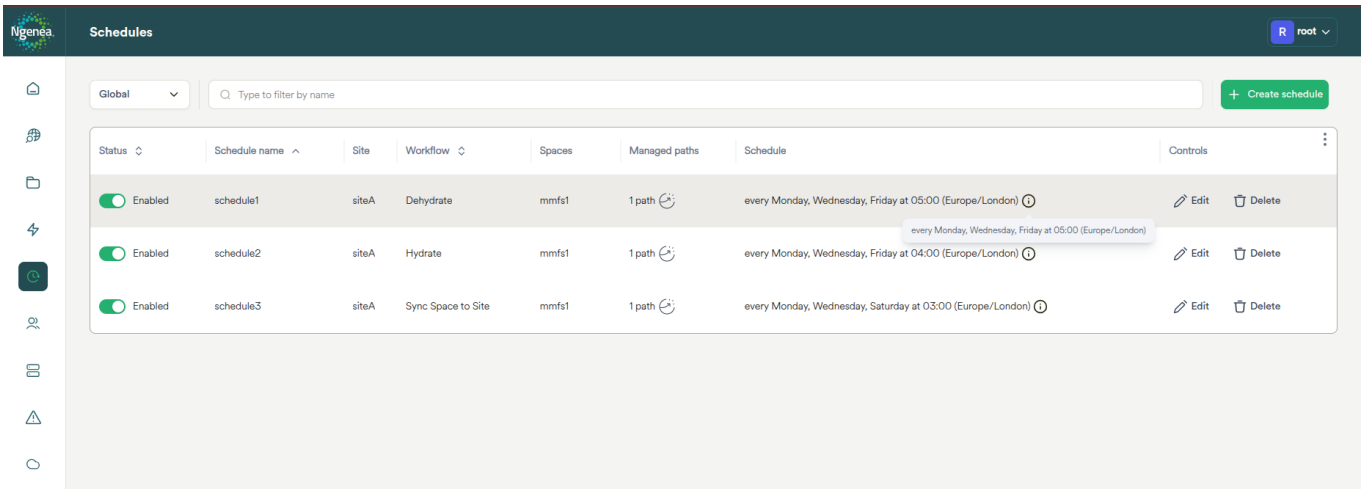
Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Alternatively `Go back` and change the proposed configuration as required or `close` the wizard to cancel the creation of the Schedule.

Editing a Schedule

Important: This function can only be performed by a Hub Administrator.

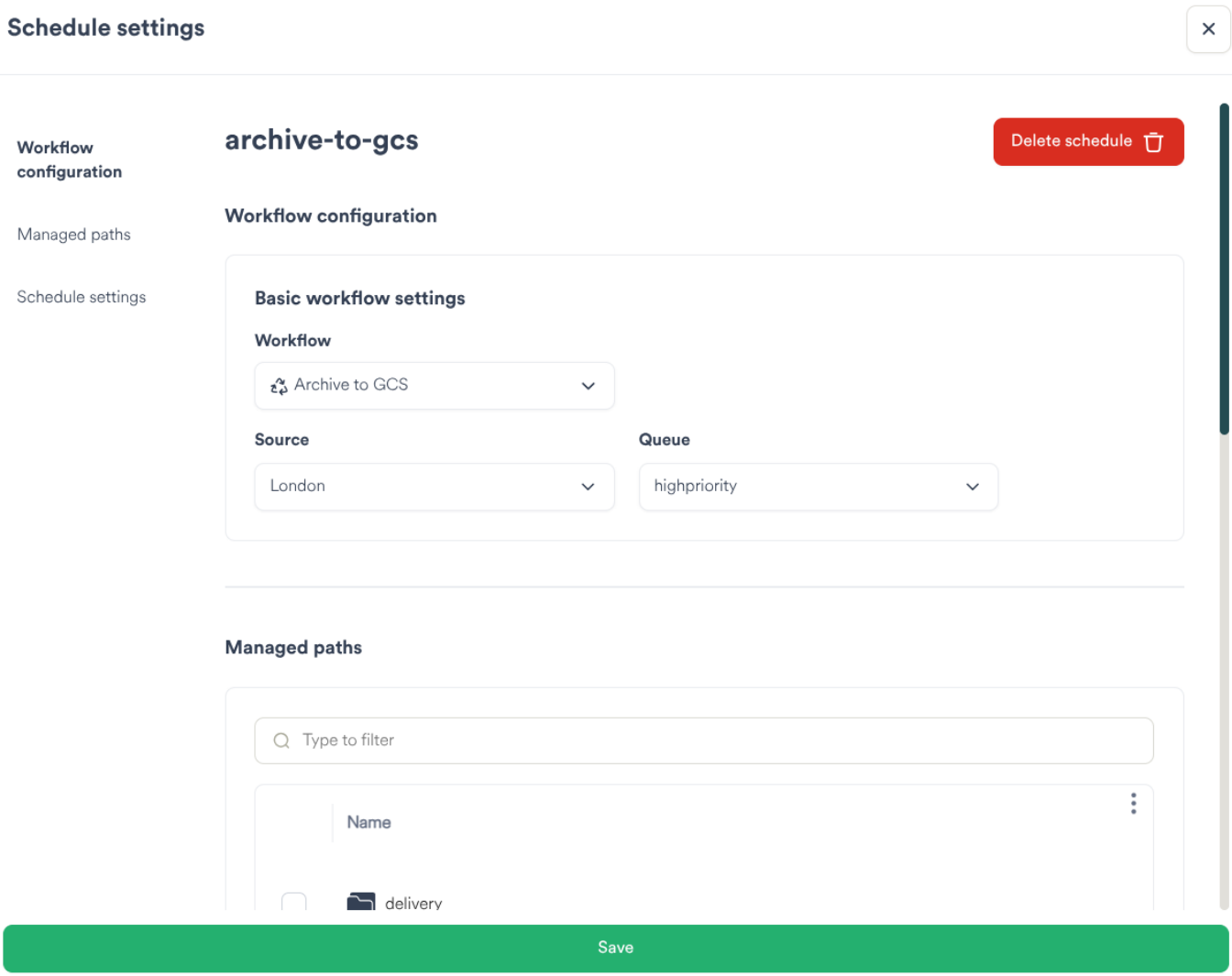
Clicking the Schedules button in the main menu bar displays the list of schedules:



Edit	Click the edit icon on the required Policy row to edit the Policy

- Modify the Schedule settings as required. Refer to Adding a Schedule for settings guidance.

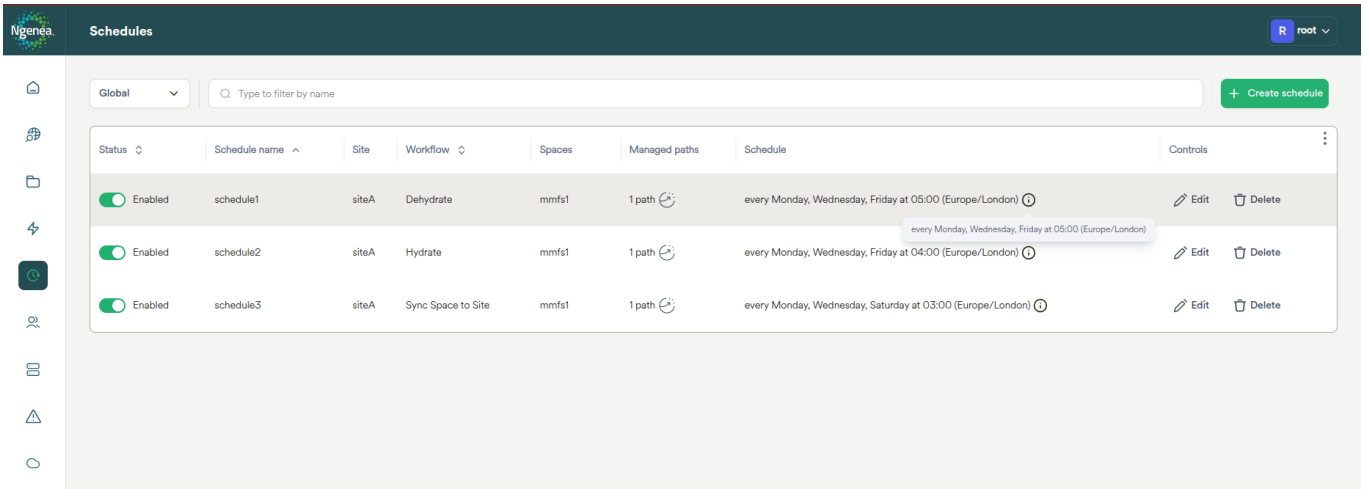
Important: In the schedule settings, hidden workflows types are also available for selection in the Workflow drop down menu. These are workflows defined with `visible:false`.



Deleting a Schedule

Important: This function can only be performed by a Hub Administrator.

Clicking the Schedules button in the main menu bar displays the list of Schedules:



Delete	Click the delete icon on the required Schedule row to delete

A confirmation dialog is raised:

- Click Yes to delete the Schedule. This action is irreversable.
- Alternately click no, or close the confirmation dialog.

Disabling a Schedule

Important: This function can only be performed by a Hub Administrator.

Enabled	Click the Enabled toggle to disable the schedule

When a schedule is disabled it will not run.

Click the toggle again to re-enable the schedule.

Policies

Hub provides the capability to perform actions to data via Policies.

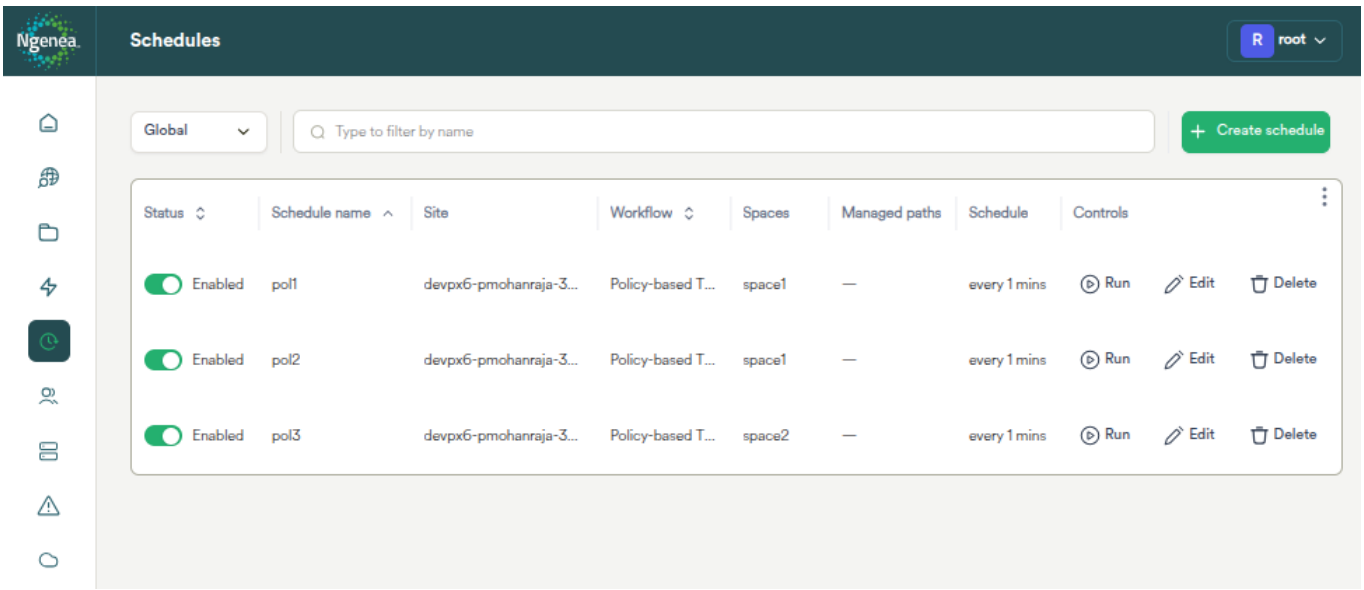
Policies comprise criteria, conditions and triggers which are applied to data within one or more Spaces residing on a Site.

Viewing Policies

To view available Policies, click the Schedules menu button.

	Navigates to the Schedules screen

Policies are displayed alongside other workflow schedules on the Schedules page.



Policy Types

There are two policy workflow types:

Policy-based Tiering

Automatically moves data between storage tiers within a site based on rules like file age, access time, or pool capacity thresholds. Use it to keep high-performance storage free by migrating cold data to cheaper tiers, with transparent recall when files are accessed again.

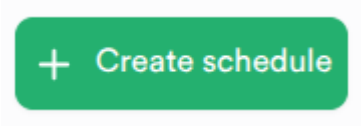
Iris Metadata Policy

Drives data movement decisions using Iris metadata (e.g. content tags, media properties) rather than filesystem attributes alone. Use it when you need to tier or archive files based on what the content is rather than when it was last accessed.

> **Note:** You must be logged in via **Iris Single Sign On (SSO)** to select the Iris Metadata Policy workflow type, otherwise the option will be greyed out.

Adding a Policy

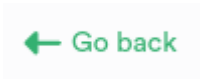
Click the Create Schedule button to display a dialog to configure a Schedule.



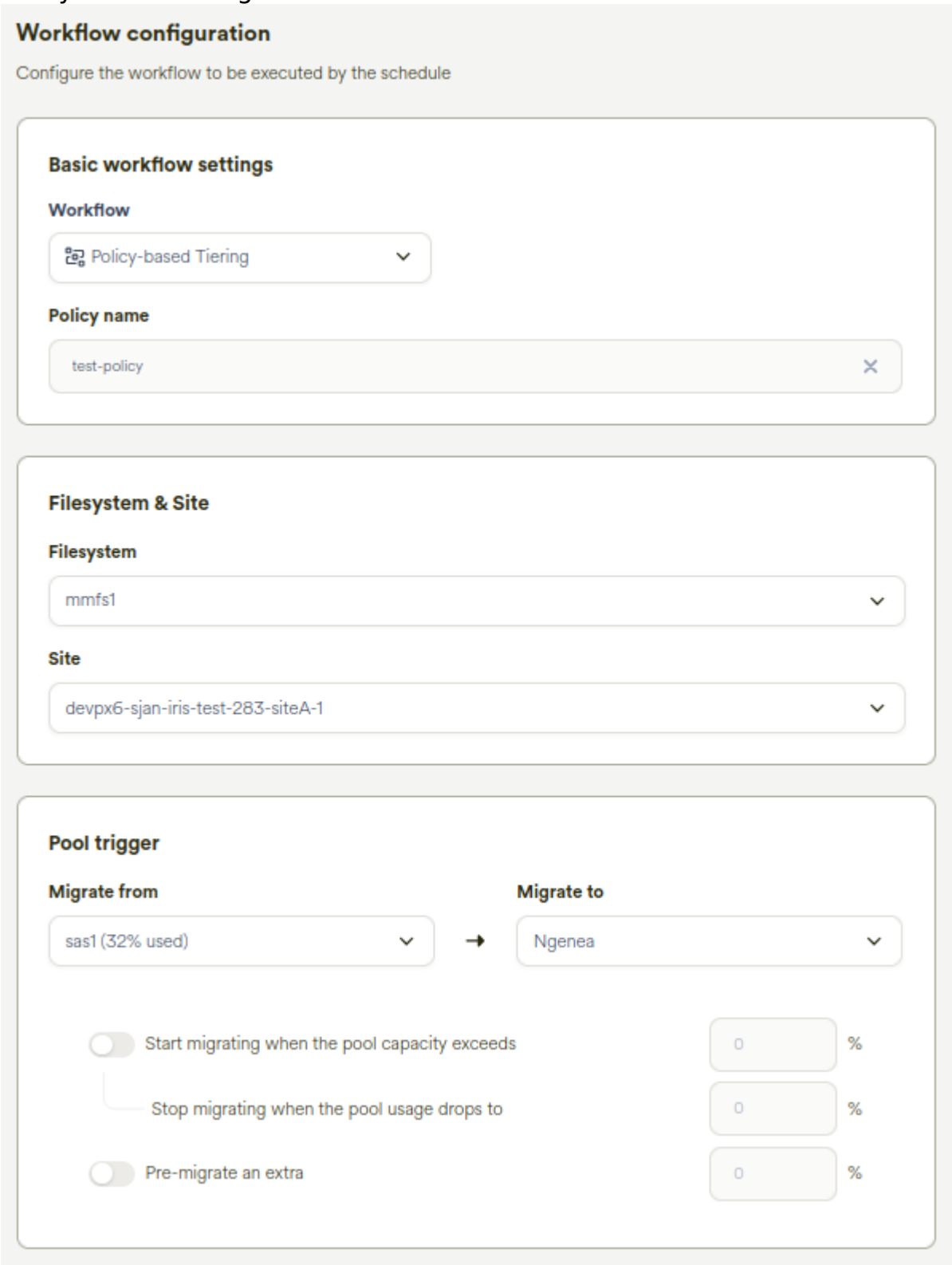
Important: This function can only be performed by a Hub Administrator

Creating a Schedule Wizard

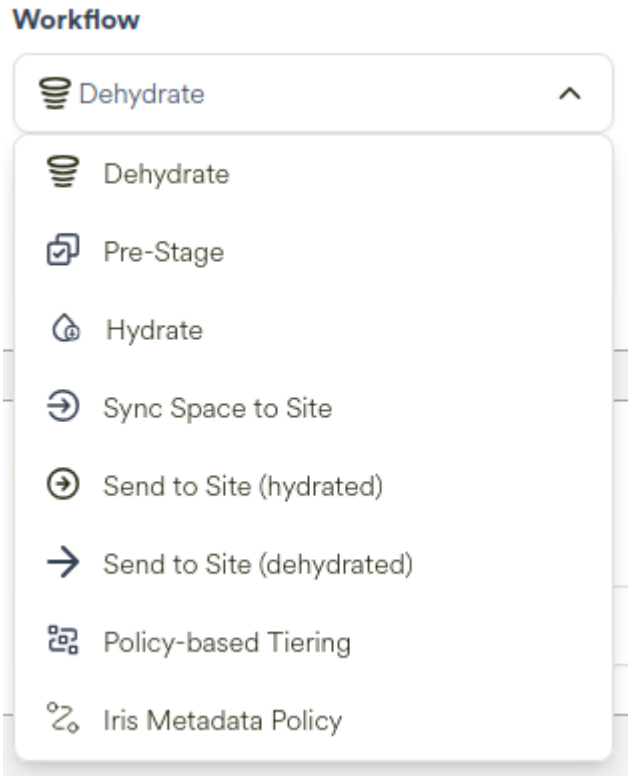
Navigating the Wizard

	Click the close button to exit the wizard. Changes are not saved.
	Click the Next button to advance to the next page of the wizard. The Next button is disabled until all required page elements are completed.
	Click the Go Back button to return to the previous wizard page.
	Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Policy-based Tiering



- Set the Workflow type to “Policy-based Tiering”
- Provide a descriptive name for the Policy



Iris Metadata Policy

Iris Metadata Policies enable the generation of candidate datasets for data movement. This is achieved by leveraging metadata extracted by **Iris**, allowing policies to be defined in a flexible and business-aware manner.

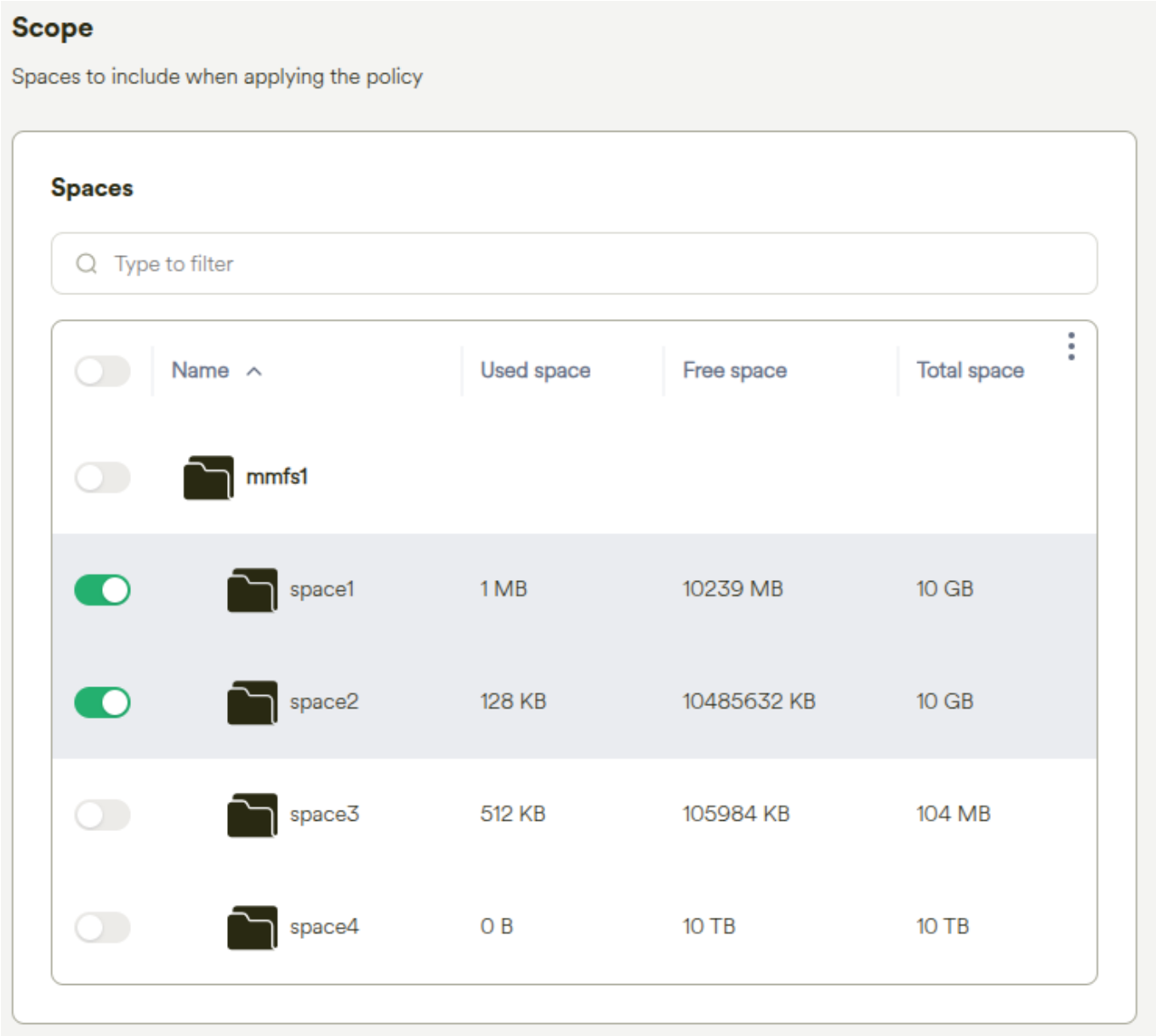
- Set the Workflow type to “Iris Metadata Policy”
- Provide a descriptive name for the Policy

Scope

Policy-based Tiering

The same Policy can run against one or more Spaces concurrently for a Site.

Policies are specific to a Site.



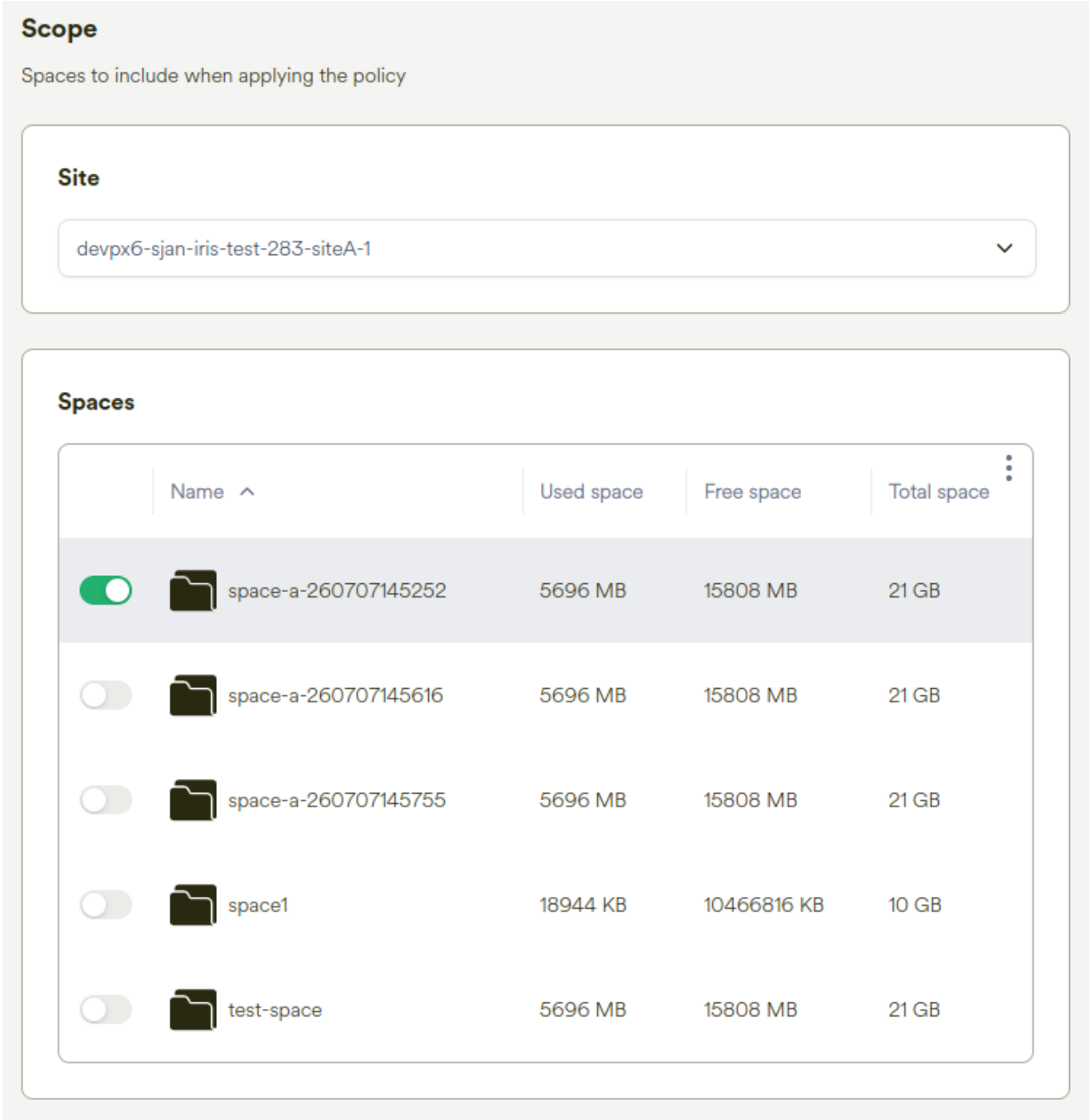
- Select the Space or Spaces to run the Policy against on the Site

Iris Metadata Policy

Iris Metadata Policies can only be run on:

- Iris enabled Sites
- Iris enabled Spaces on the selected Iris enabled Site

Only one Iris enabled space can be processed by an Iris metadata policy.

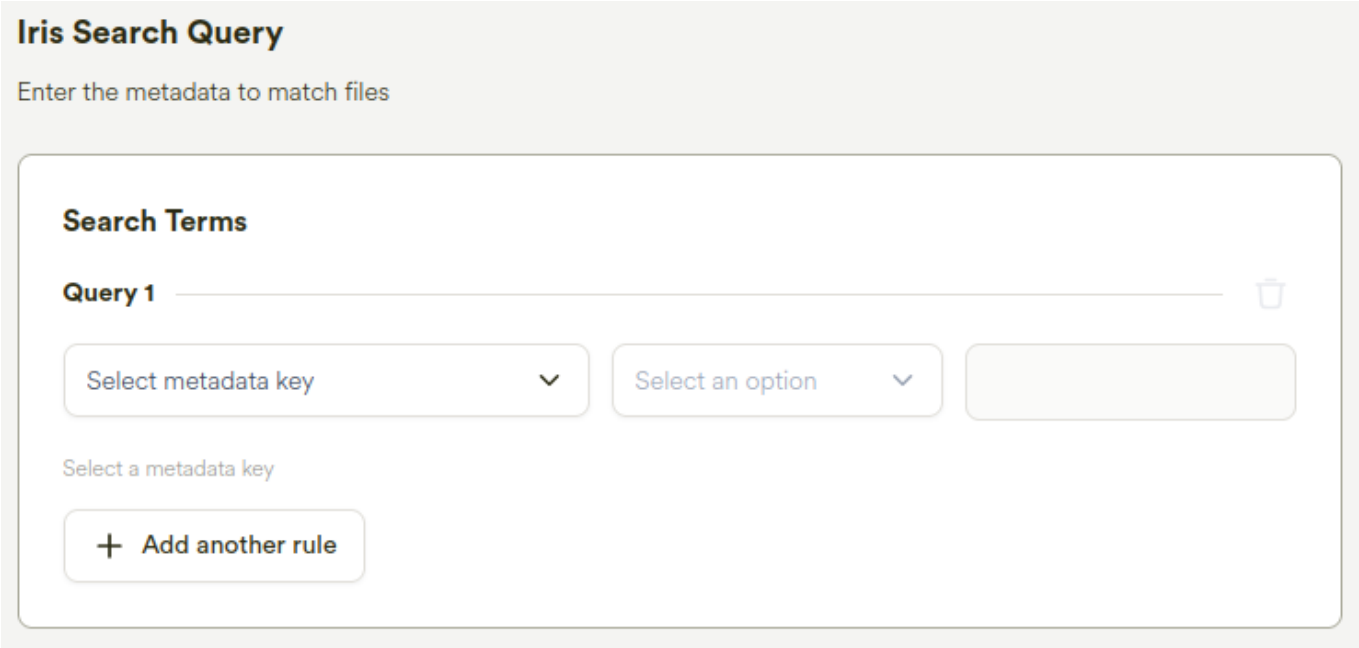


- Select the Iris enabled Space to run the Policy against on the Iris enabled Site

Iris Search Query

The Query Builder is a visual interface component designed to simplify the construction of metadata queries.

This policy wizard step is only presented for Iris Metadata Policy workflows.



It supports multiple data types, including:

- Text
- Keyword
- Boolean
- Long
- Float
- Date

This enables users to build structured and precise queries without needing to write raw query syntax.

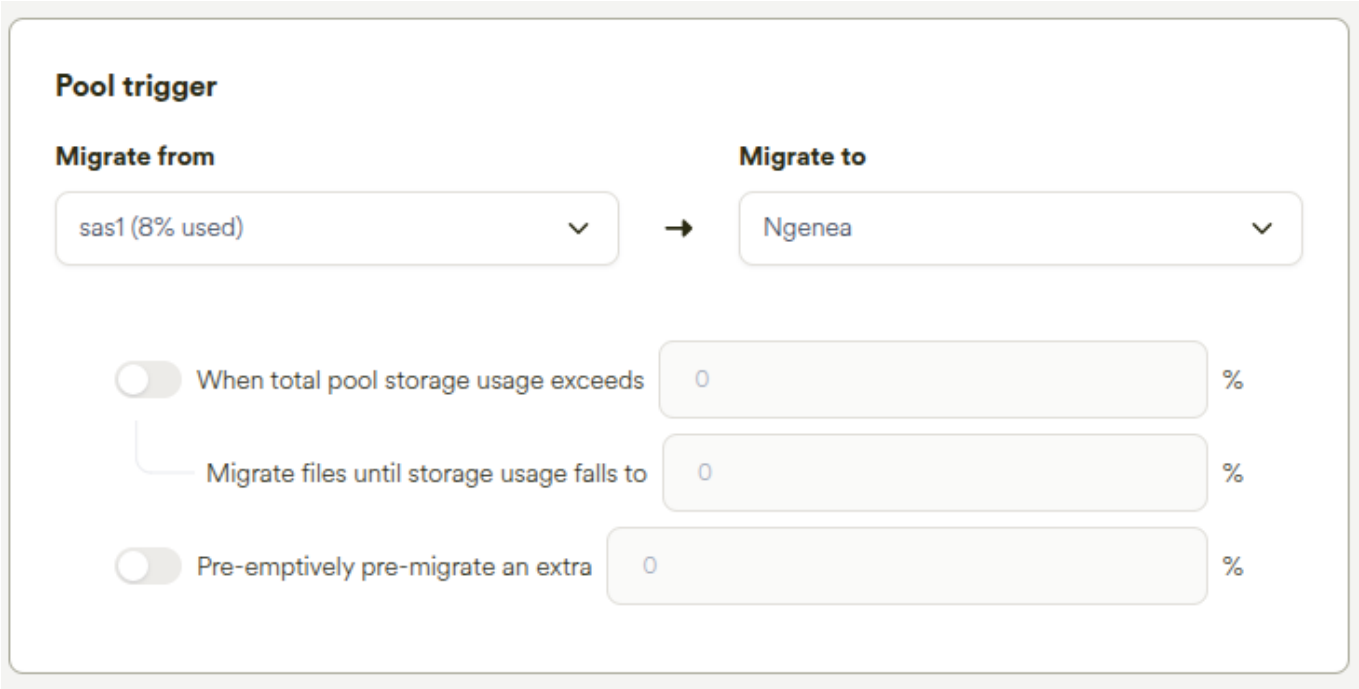
Pool triggers

Setup a pool trigger to define the flow of data.

Examples of supported data flows are:

Pool to Ngenea

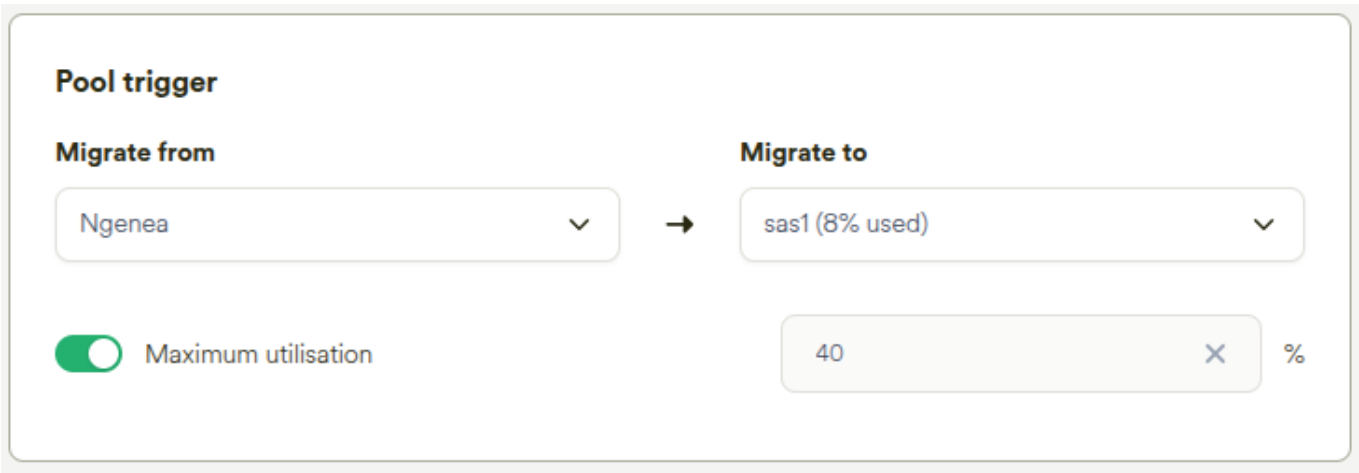
Data in pool `sas1` is dehydrated to an Ngenea Target according to the Policy conditions.



Ngenea to Pool

Data in an Ngenea Target is rehydrated to pool `sas1` according to the Policy conditions.

When `sas1` is configured as the migration target, the `sas1` pool utilisation limit determines how much data can be migrated. Data matching the policy conditions is selected for migration from Ngenea until the `sas1` pool reaches its configured maximum utilisation limit. If the utilisation target cannot be achieved, a best-fit approach is applied.



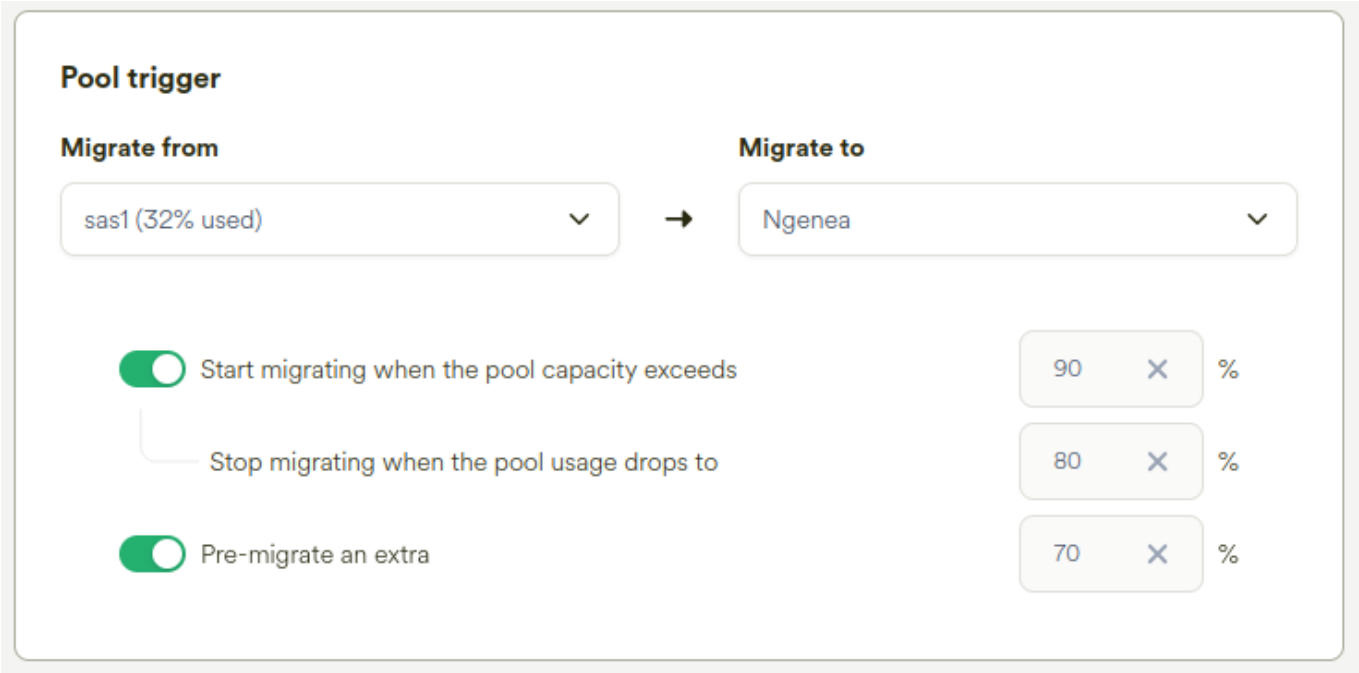
Ngenea Watermarks

Utilise Ngenea watermarks to provide finite control over data states during Policy processing.

The following example provides (assuming 93% utilisation):

- Data will only be de-hydrated when the Storage Pool 1 (E.G. `sas1` in the above example) is utilised over 90%
- 13% of data residing on the pool will be de-hydrated, reducing utilisation to 80%

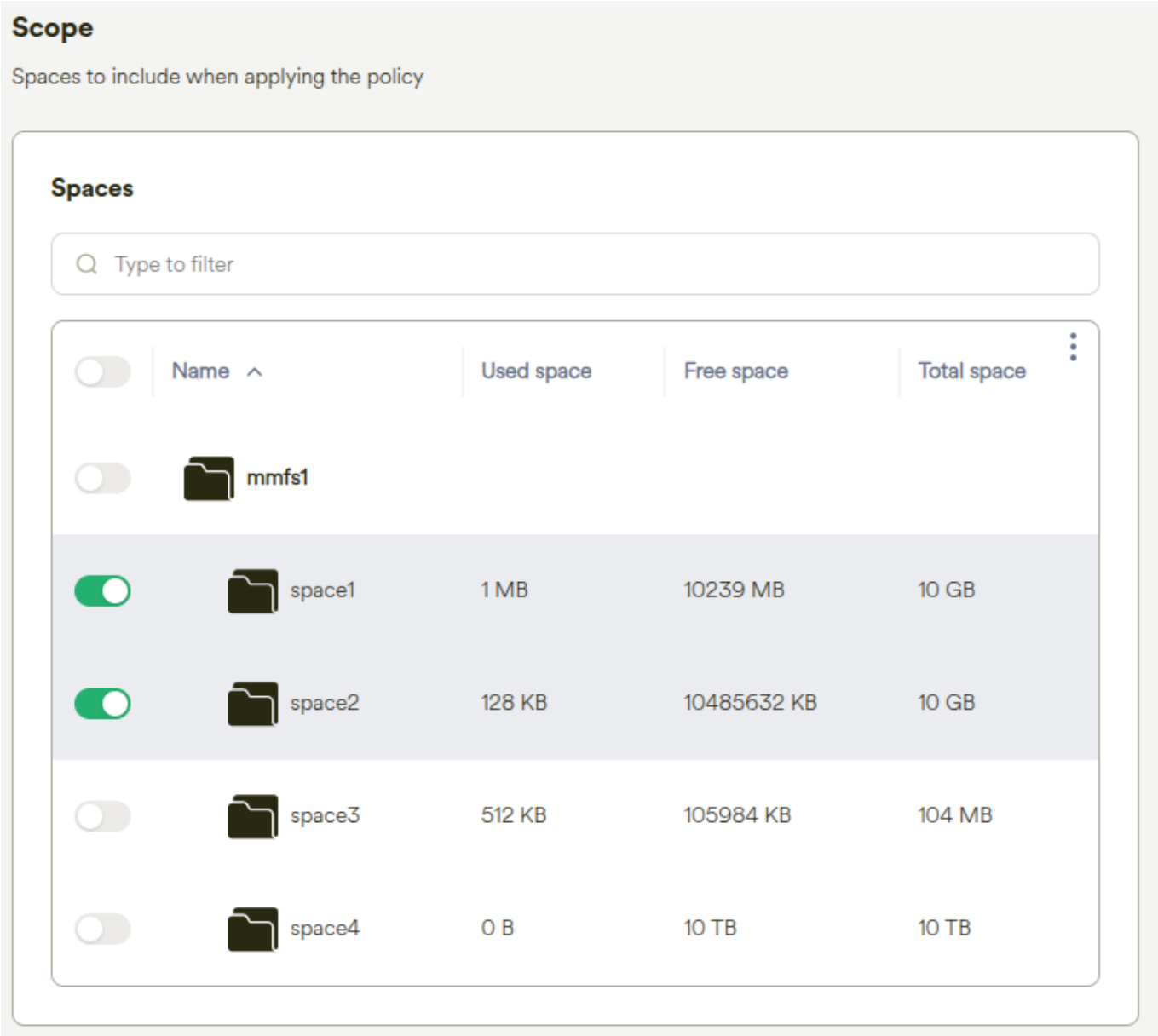
- A further 10% of data residing on the pool will be Pre-Staged, ensuring that future de-hydrations process the first 10% of data faster as the data is already present in the Ngenea Target



Scope

The same Policy can run against one or more Spaces concurrently for a Site.

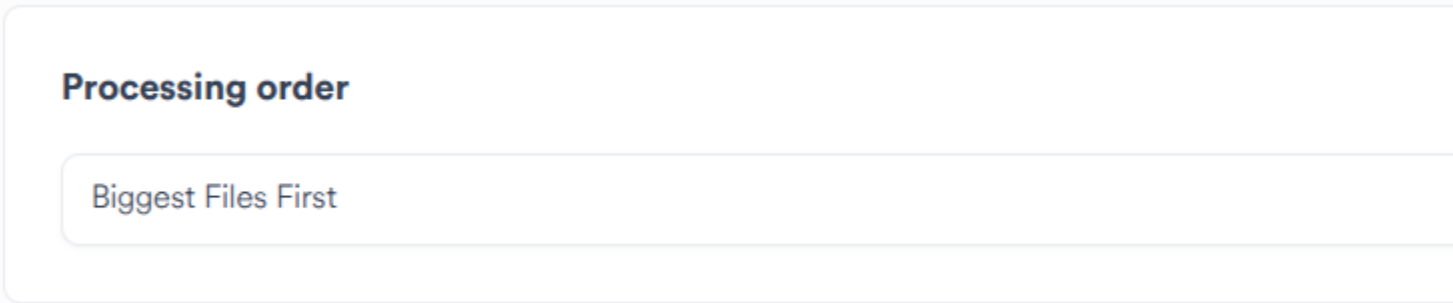
Policies are specific to a Site.



- Select the Space or Spaces to run the Policy against on the Site.

Processing order

Select the processing order of the data.

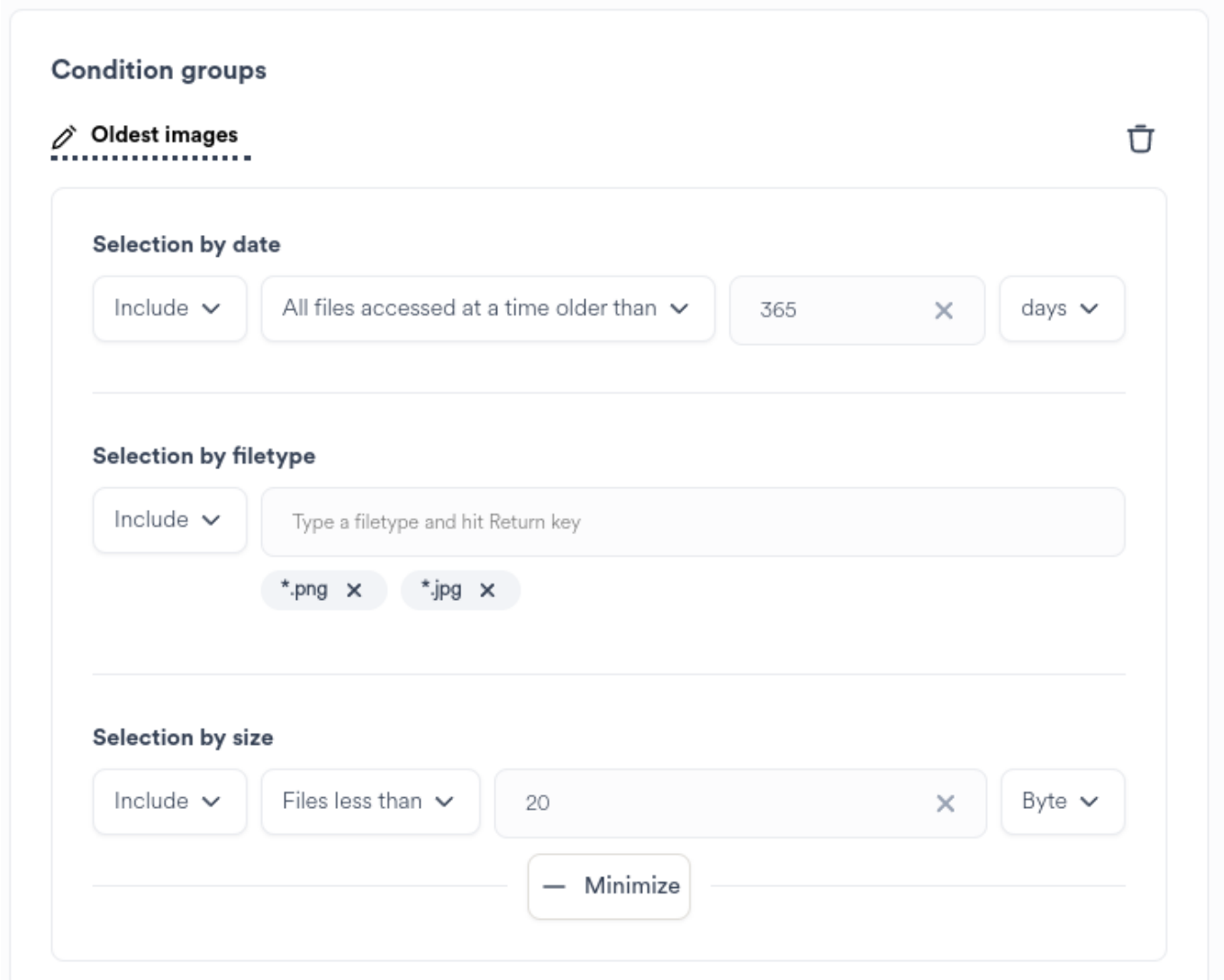


Conditions

For more finite control of the data targeted, add a Condition group.

+ Add condition group	Click the `Add condition group` button to add a Condition group

- Specify the criteria for the condition group as necessary
- It is not required to specify all selections of the Condition group



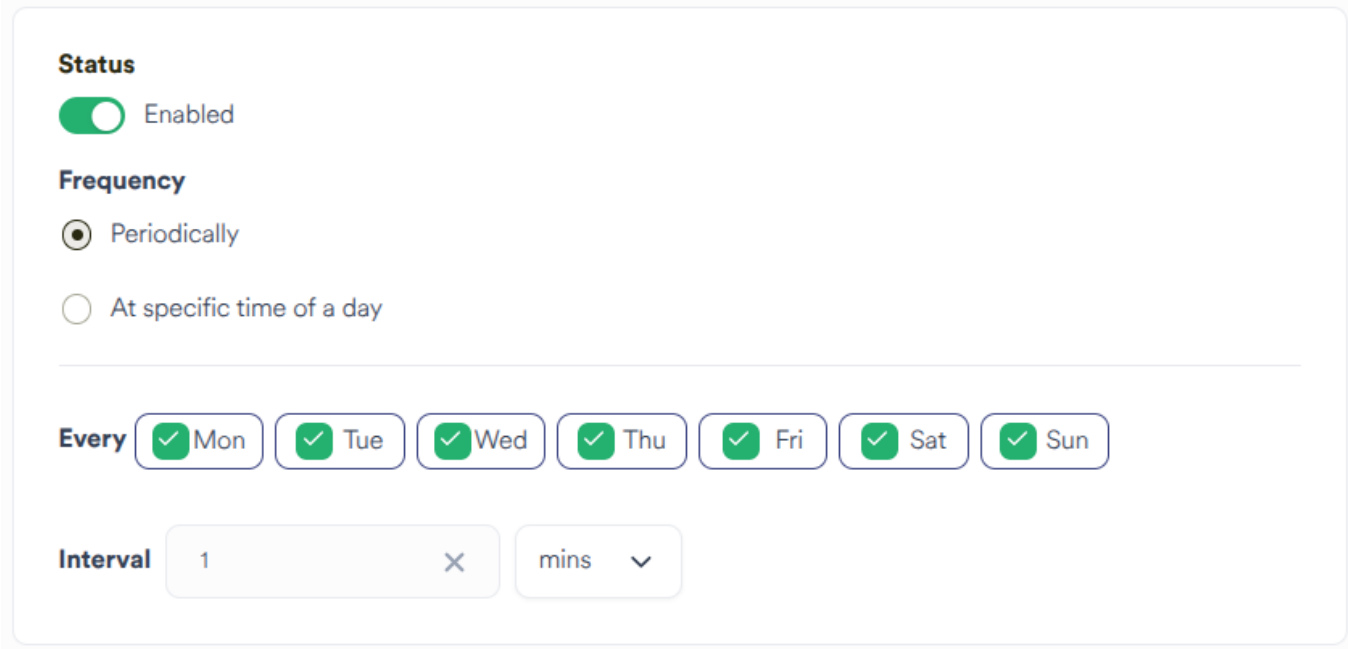
Perform optional actions:

Condition group	Click the Condition Group Edit button to name the Condition group
	Click the delete icon to remove the Condition group
+ Add condition group	Click the `Add condition group` button to add further Condition groups

Automation

To setup a regular scheduled Policy run, enable the Automate Schedule slider.

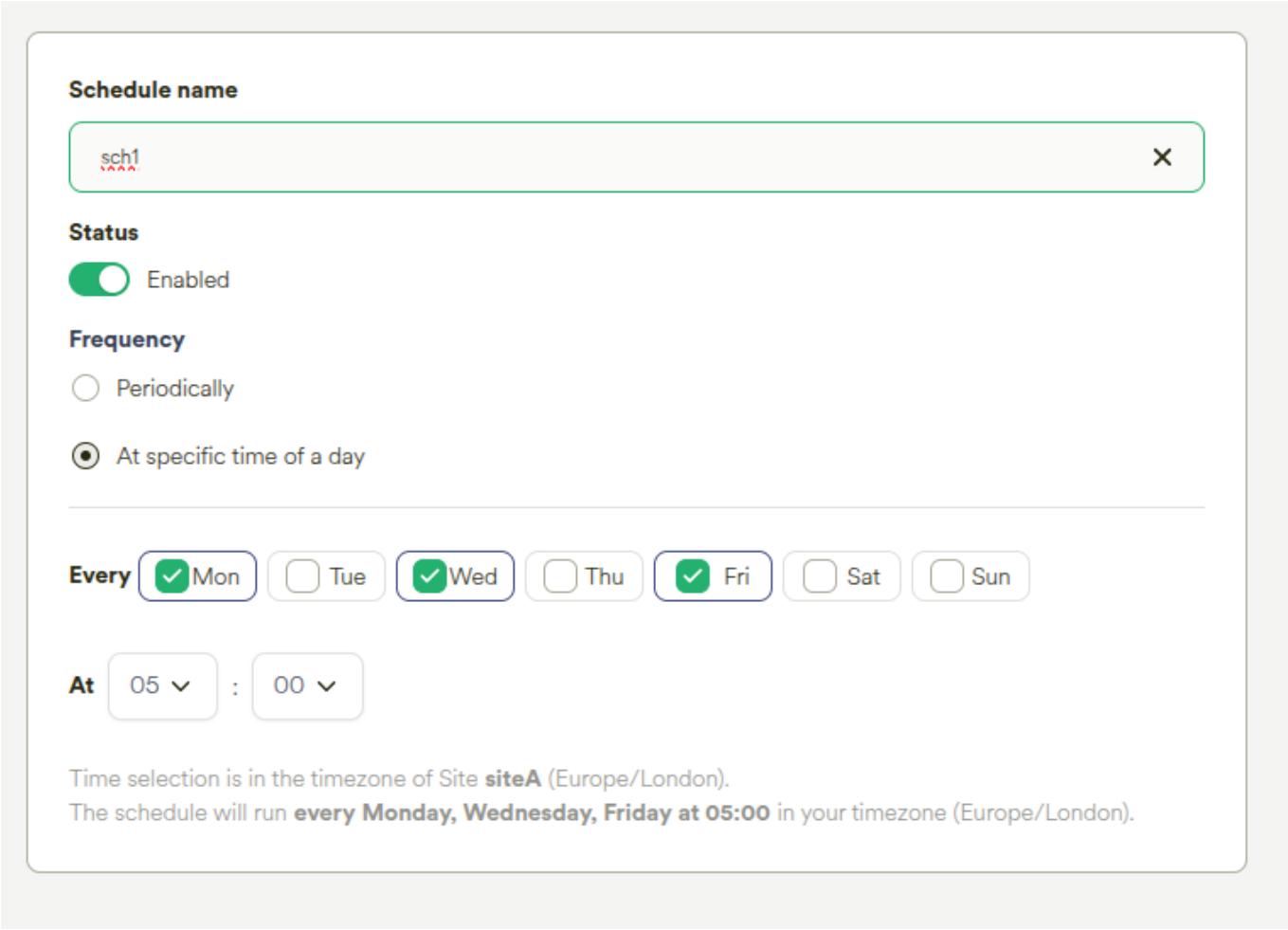
Determine the required frequency of the Policy run.



Choosing Periodically will ensure that the Policy will run on the next interval set.

E.G:

- 1 hour: The Policy will run on the next hour (12.00, 13.00)
- 15 mins: The Policy will run on the next 15 minute interval past the hour (15, 30, 45, 00)



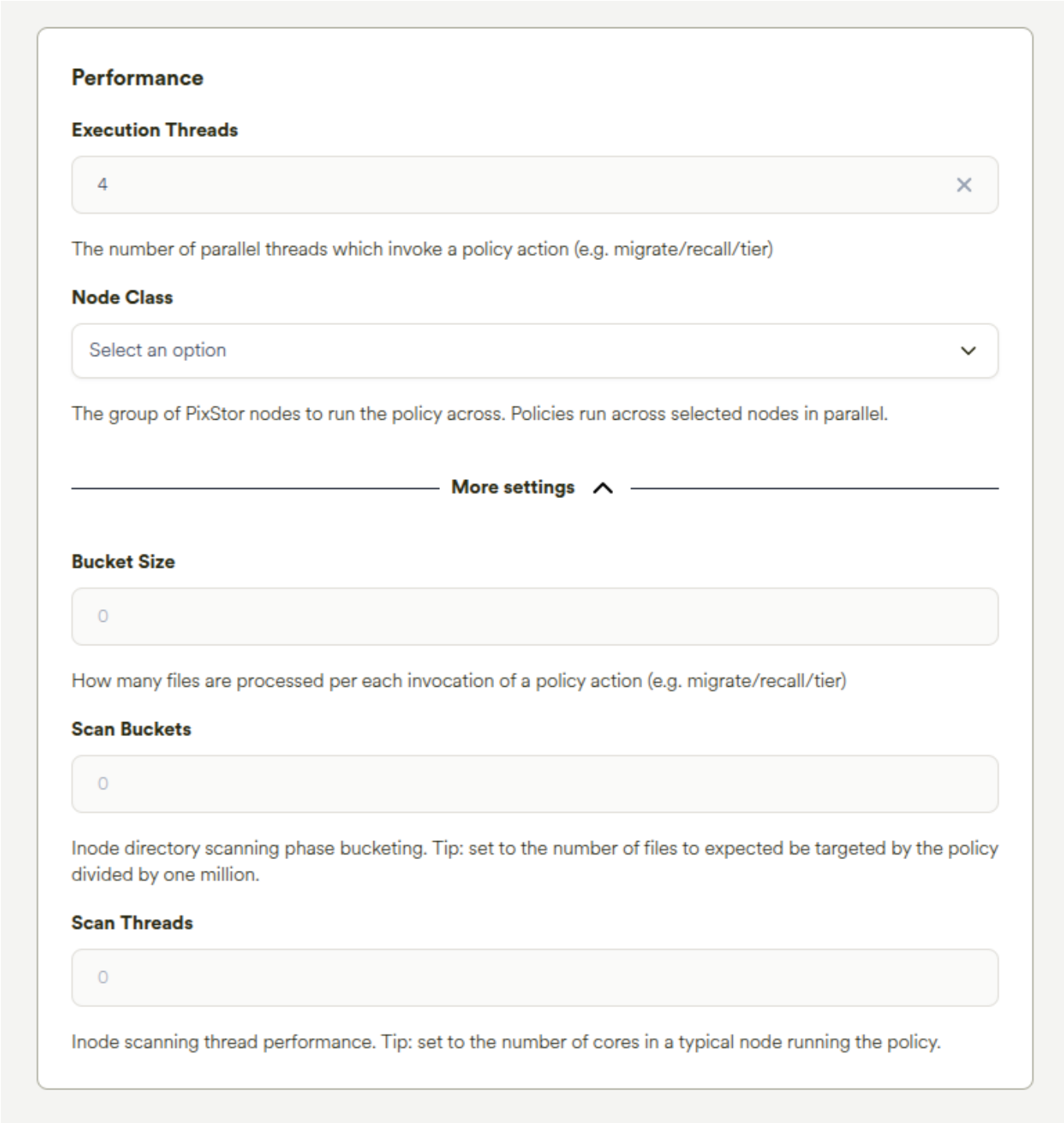
Choosing At Specific Time of A Day allows the Policy to be scheduled once per chosen day at a specific time of day.

Hint: The schedule time is set in the Site's local timezone, but will be stored in the UTC timezone.

The schedule may be disabled at the point of creation by switching the `Enabled` toggle to `Disabled`.

Performance

Choose the number of Threads to assign to the policy run.



Hint: The number of parallel threads which invoke a policy action (e.g. migrate/recall/tier)

The group of PixStor nodes to run the policy across. Policies run across selected nodes in parallel.

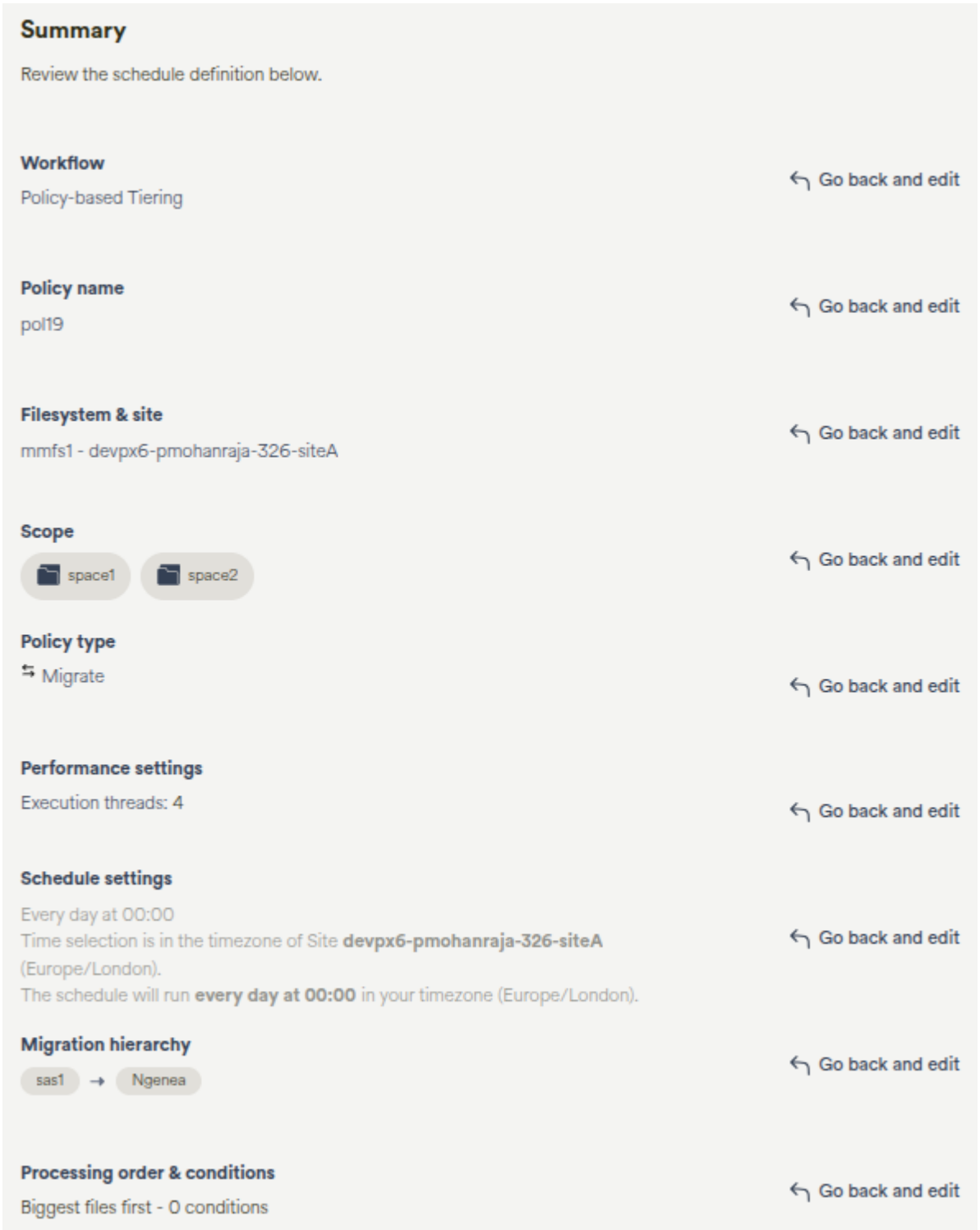
How many files are processed per each invocation of a policy action (e.g. migrate/recall/tier)

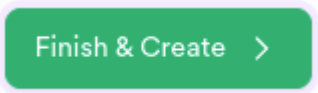
Inode directory scanning phase bucketing. Tip: set to the number of files to expected be targeted by the policy divided by one million.

Inode scanning thread performance. Tip: set to the number of cores in a typical node running the policy.

Summary

Upon completing the wizard steps a summary is presented:





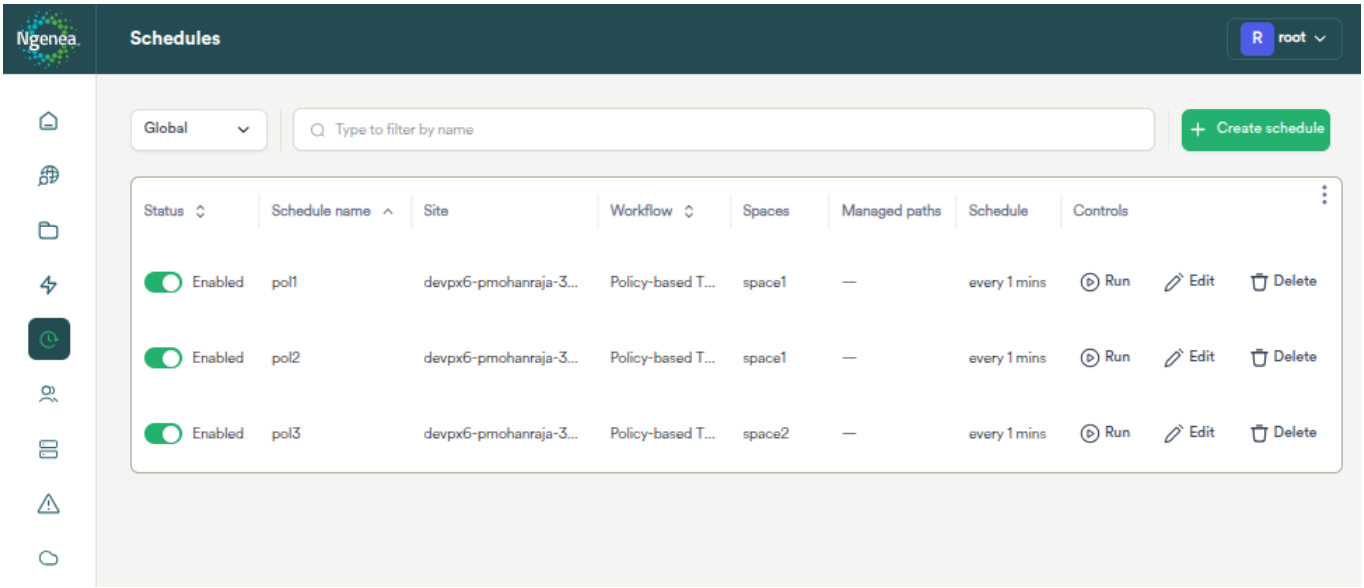
Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Alternatively `Go back` and change the proposed configuration as required or `close` the wizard to cancel the creation of the Policy.

Editing a Policy

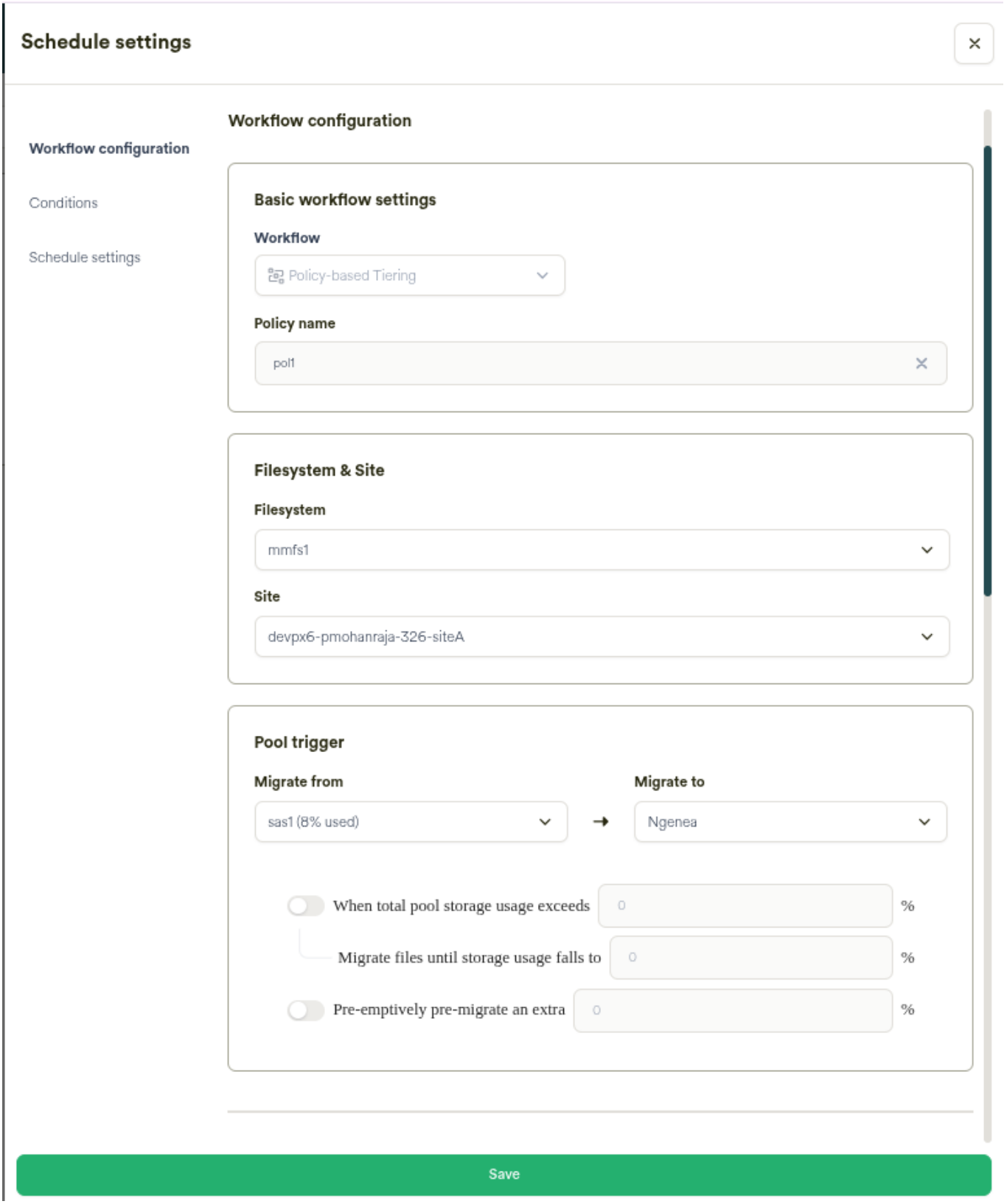
Important: This function can only be performed by a Hub Administrator.

Clicking the Schedules button in the main menu bar displays the list of schedules:



Edit	Click the edit icon on the required Policy row to edit the Policy

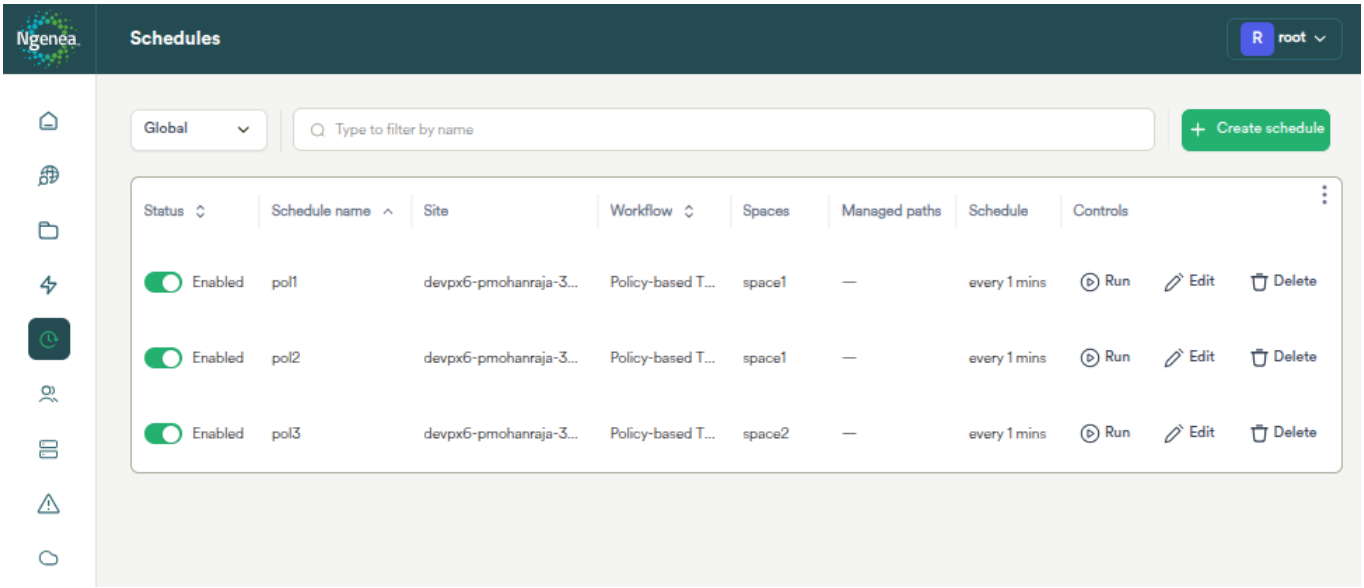
- Modify the Policy settings as required. Refer to Adding a Policy for settings guidance.



Deleting a Policy

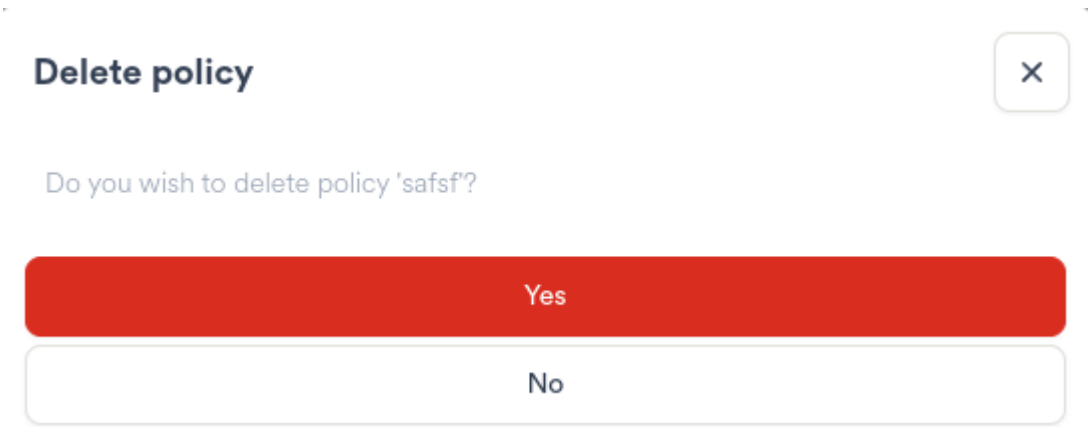
Important: This function can only be performed by a Hub Administrator.

Clicking the Schedules button in the main menu bar displays the list of Schedules:



Delete	Click the delete icon on the required Schedule row to delete the Policy

A confirmation dialog is raised:



- Click Yes to delete the Policy. This action is irreversable.
- Alternately click no, or close the confirmation dialog.

Running a Policy

Policies are run either on a Schedule, as specified in the Wizard on creation or in the Schedule settings.

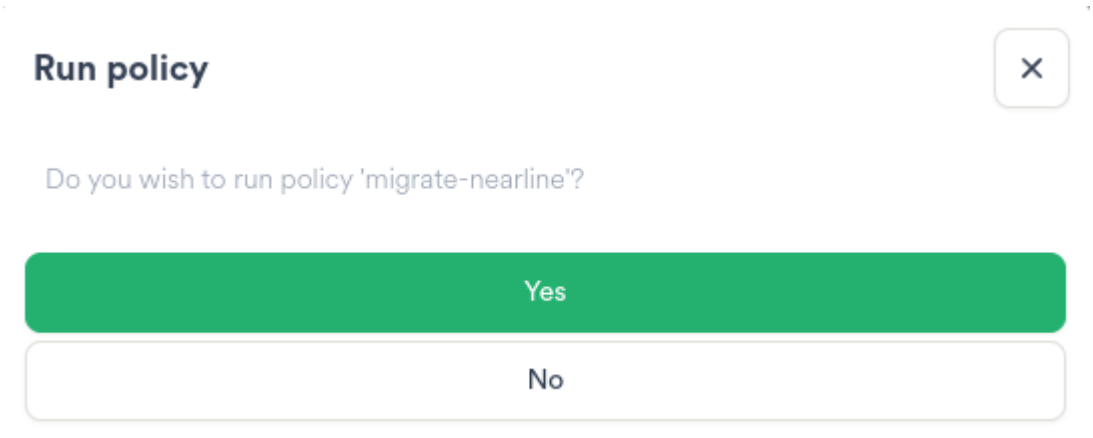
Alternatively, policies can be run on demand by clicking the Run button in the relevant row of the Action column on the Schedules page.

Important:

- Policies cannot be cancelled from the Hub UI once started
- Progress feedback is not visible in the UI
- Ensure the Policy is appropriate prior to running or scheduling

Run	Click the Run button to manually start the Policy run

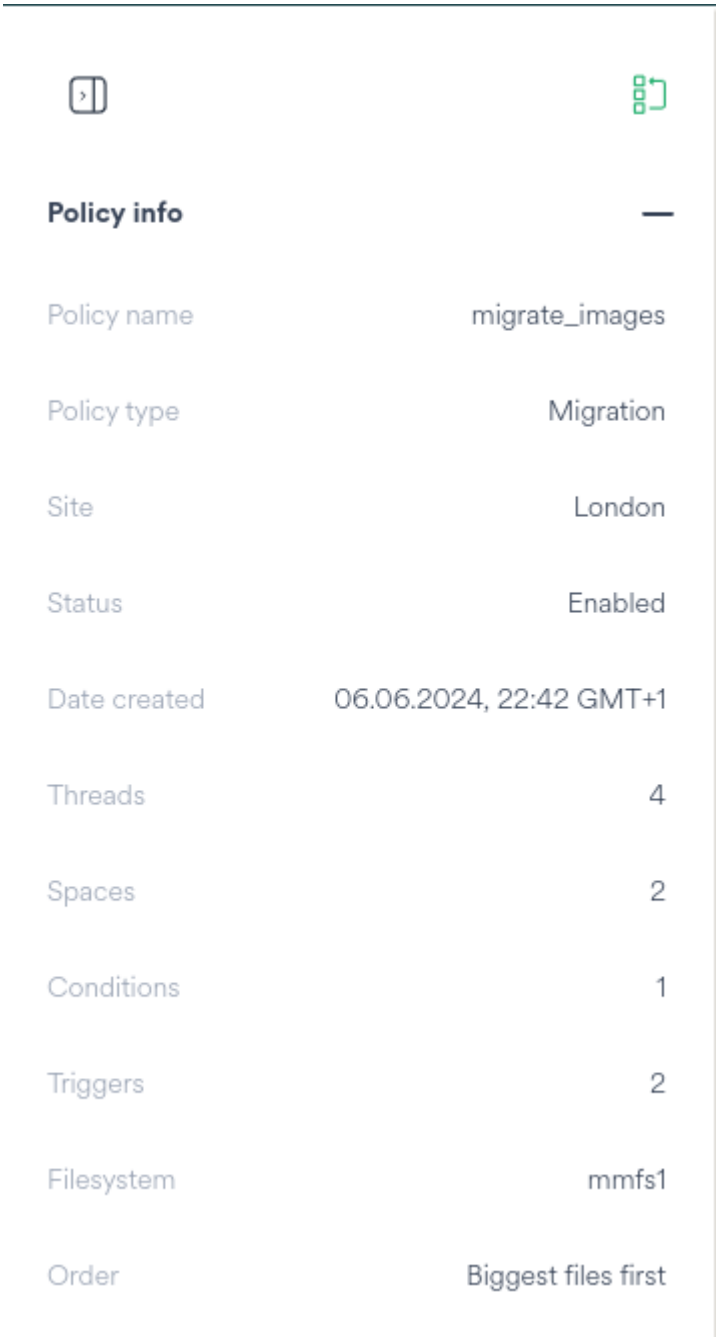
A confirmation dialog is raised:



- Click Yes to run the Policy.
- Alternately click no, or close the confirmation dialog.

Running Policies are shown on the Jobs page, subject to the above constraints. I.E. no controls are available for Polices on the Jobs page.

Policies provide specific Policy information when the Job is selected in the Jobs Page.



Refer to doc:*jobs* for further Jobs guidance.

Cancelling a Policy

Presently there is no capability to cancel a running Policy via the Hub UI.

Running policies can be terminated by issuing a `kill -9` to the `mmapplypolicy` process running on the relevant Ngenea node on the Site.

Upon terminating the running policy via the CLI, the Hub job will also terminate. The resultant Job state is (correctly) Failure - as the running policy was forcibly terminated.

Groups & Users

Hub provides management of Groups and Users internally.

Users may be authenticated by external directory services such as LDAP or Active Directory.

Two default groups are provided:

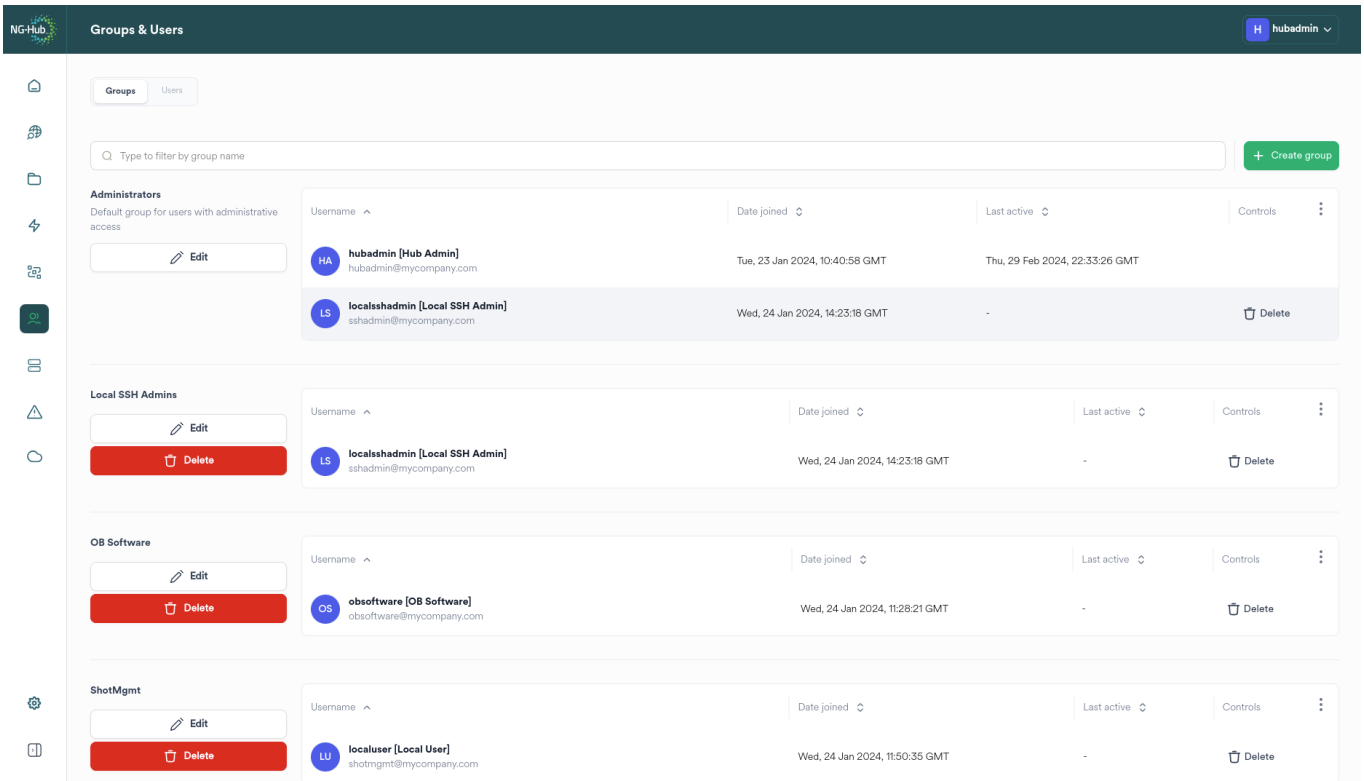
- Members of Administrators can configure and manage all Sites and Spaces
- Members of Users have read-only access to all Spaces

Additional groups can be deployed to provide restricted access for users to Sites and Spaces.

Viewing Groups

To view the available Groups managed in Hub, select the Group tab.

Important: Group management can only be performed by a Hub Administrator.

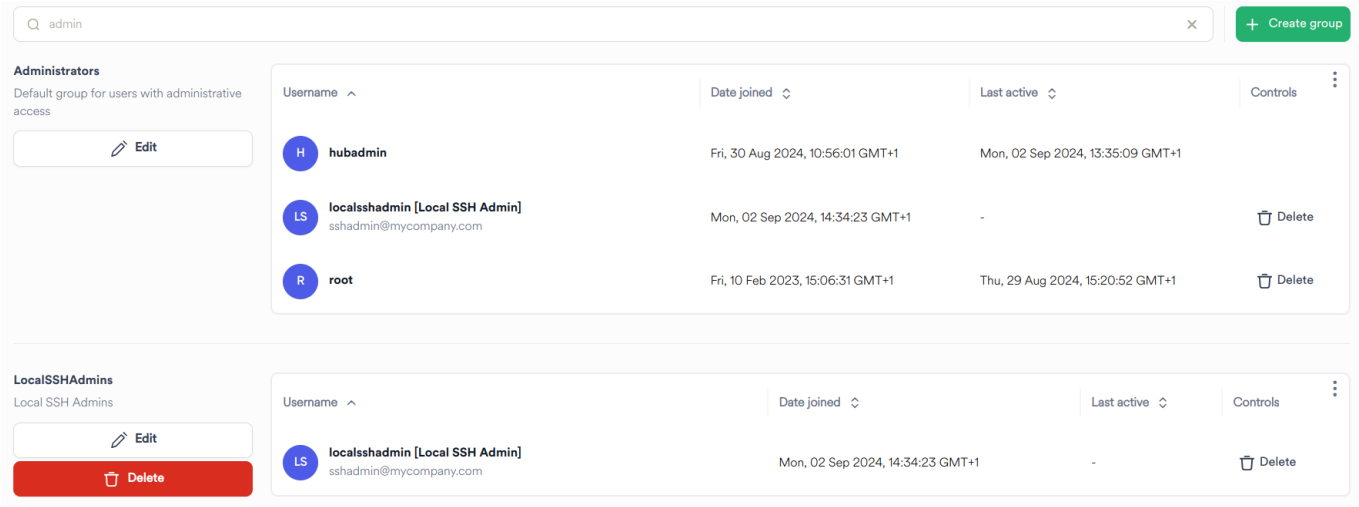


The Groups tab provides the following actions:

- Group creation
- Group deletion
- Group member management

Filtering the Groups View

To display Groups matching keywords, enter the keywords in the filter bar.



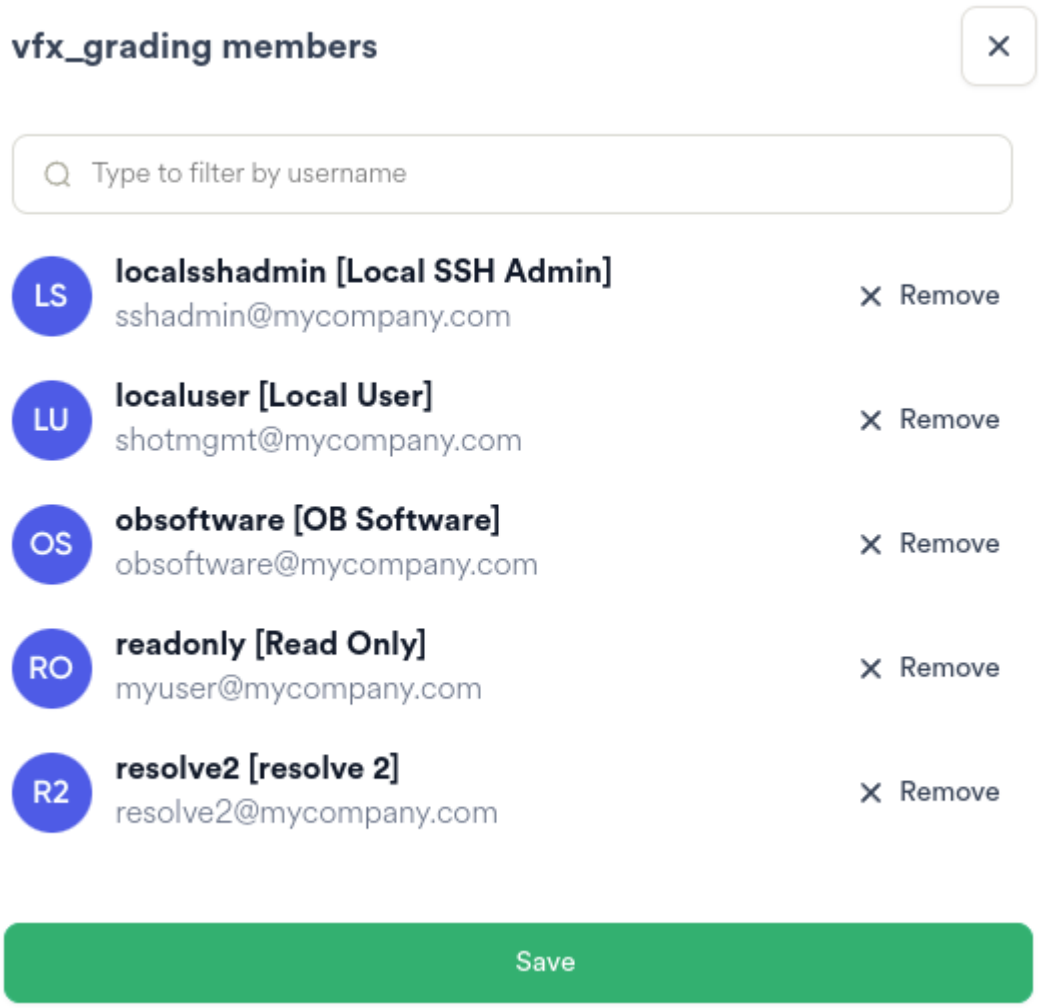
Group Members

The Groups View will only display up to four members per group.

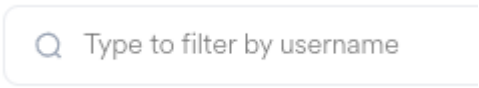
View all members

Click the View all members button to view all the members of the group

Clicking the View all members button raises the Group Membership dialog.



To filter for a User within the list of group members enter a keyword in the Filter for...



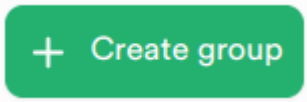
Click the Remove button to remove a User from the Group

Tip: If a User has been inadvertently removed, do not press the Save button, instead click off the Group members dialog to the main area of the screen.

Clicking the Save button at the bottom of the Group members dialog saves any changes made.

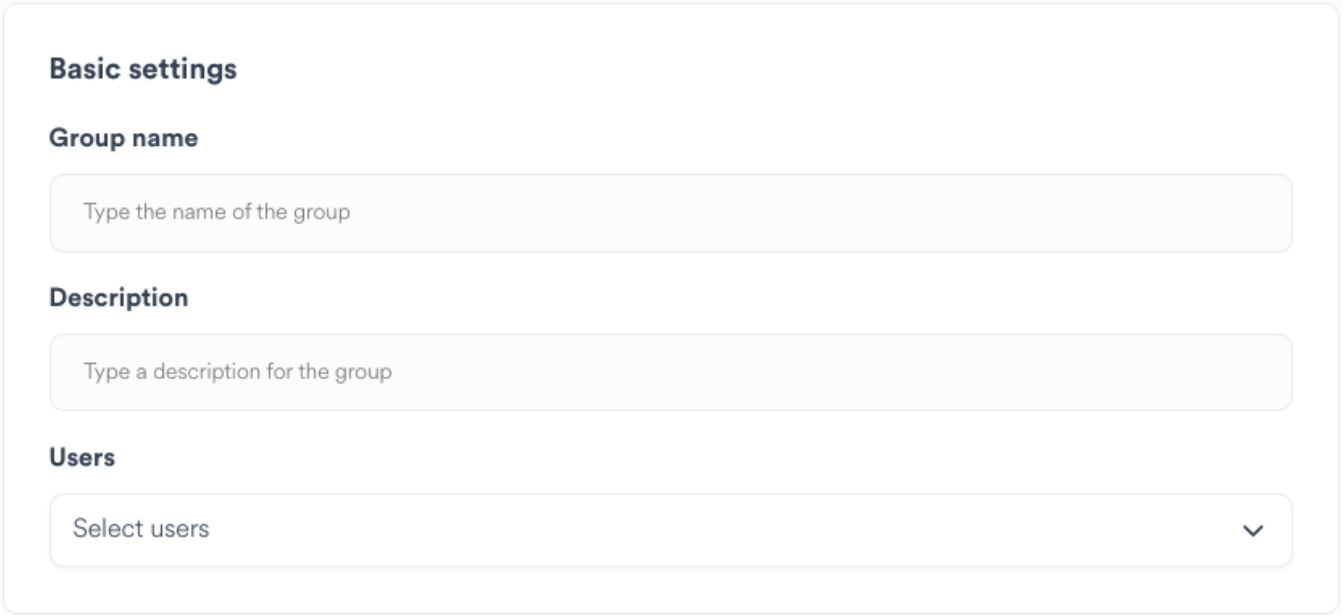
Creating Groups

Click the Create Group button to display a dialog to configure a new group.



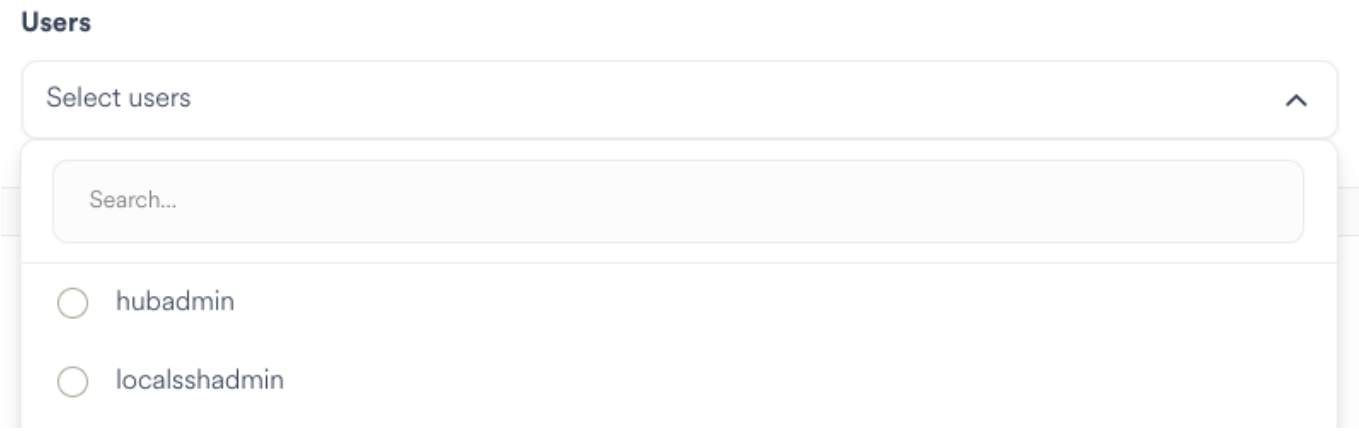
Important: This function can only be performed by a Hub Administrator

Basic Group Options



- Enter a name for the group. The name is case-sensitive and may not contain whitespace
- Enter a human-readable description for the group.

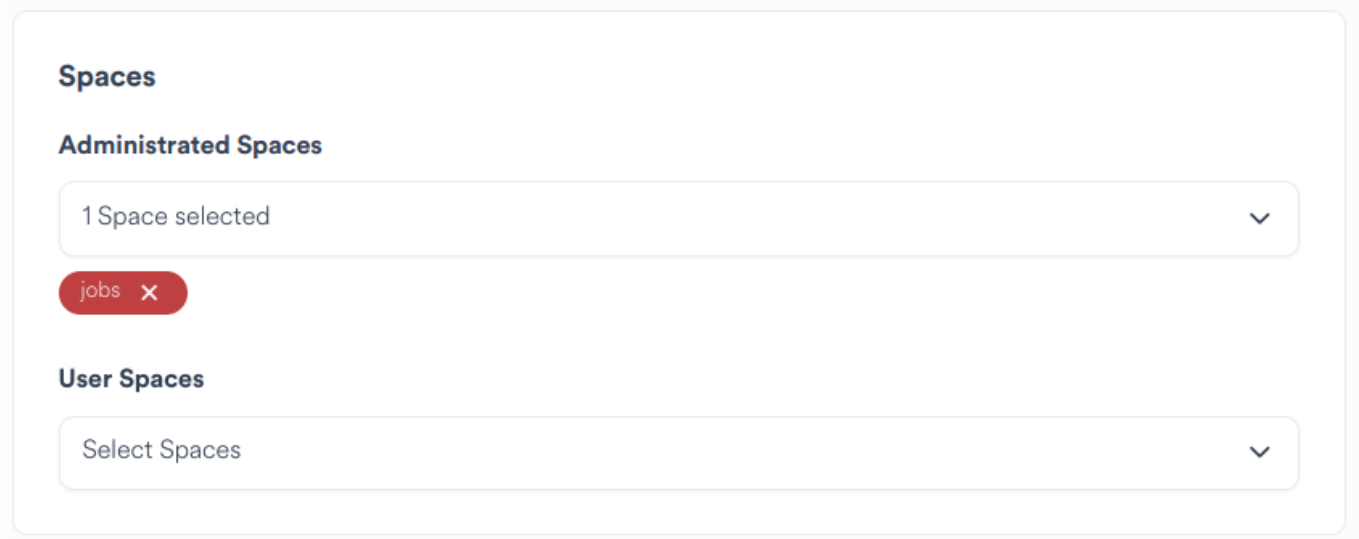
Select zero or more users to assign to the group



- Click a username to select it
- Click the username again to deselect it

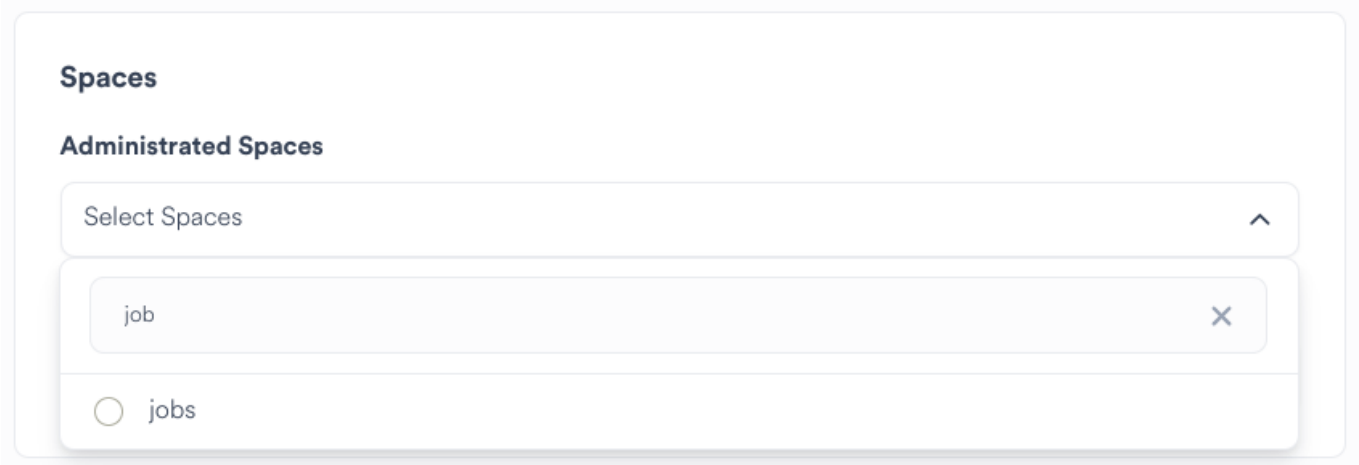
Group Space Options

Select which Spaces can be viewed and administered by users in this group.



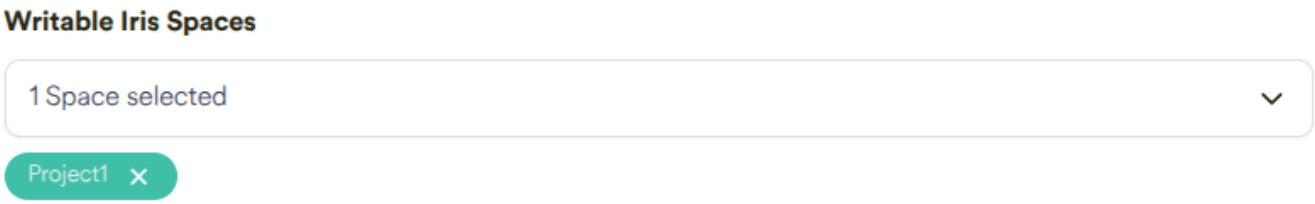
- Administered Spaces can be viewed and browsed, and can have their settings changed
- User Spaces can be viewed and browsed, but cannot have their settings changed

To filter for a Space within the list of spaces enter a keyword to filter for



Selected spaces are shown as chips. Clicking the ‘X’ on a chip deselects the space.

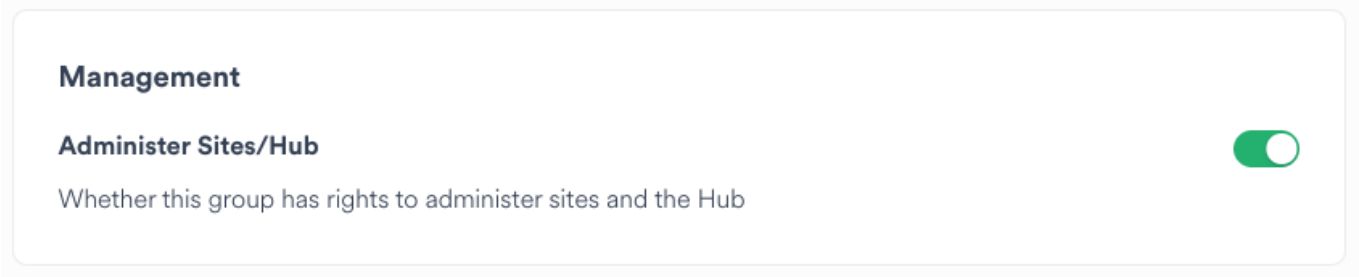
Writable Iris Spaces



Spaces which are selected as “Writable Iris Spaces” can be written in Iris. Any space which isn’t selected is read-only in Iris.

Management Settings

Select whether users in the group have the right to administer Ngenea Hub

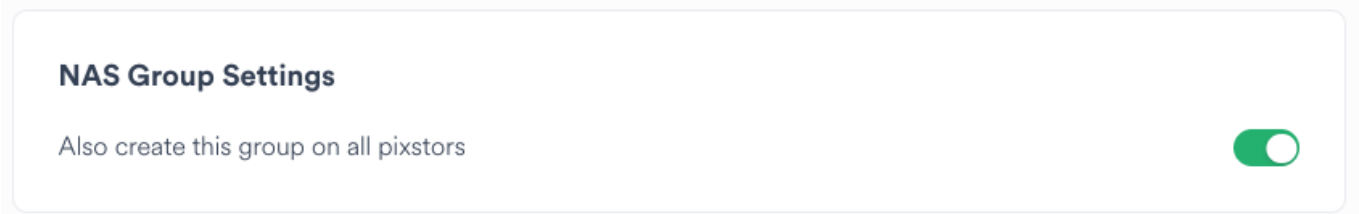


This includes the right to:

- Configure site settings
- Manage and run policies
- Manage Groups and Users
- View Alerts
- Manage, browse, and import from Targets
- Manage Global Settings

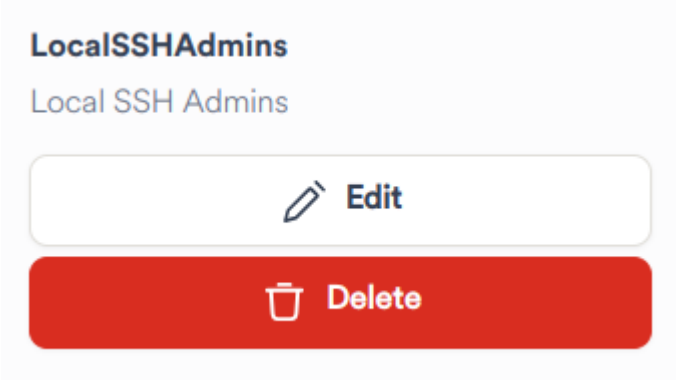
NAS Group Settings

Select whether the group should be created as a local NAS group.

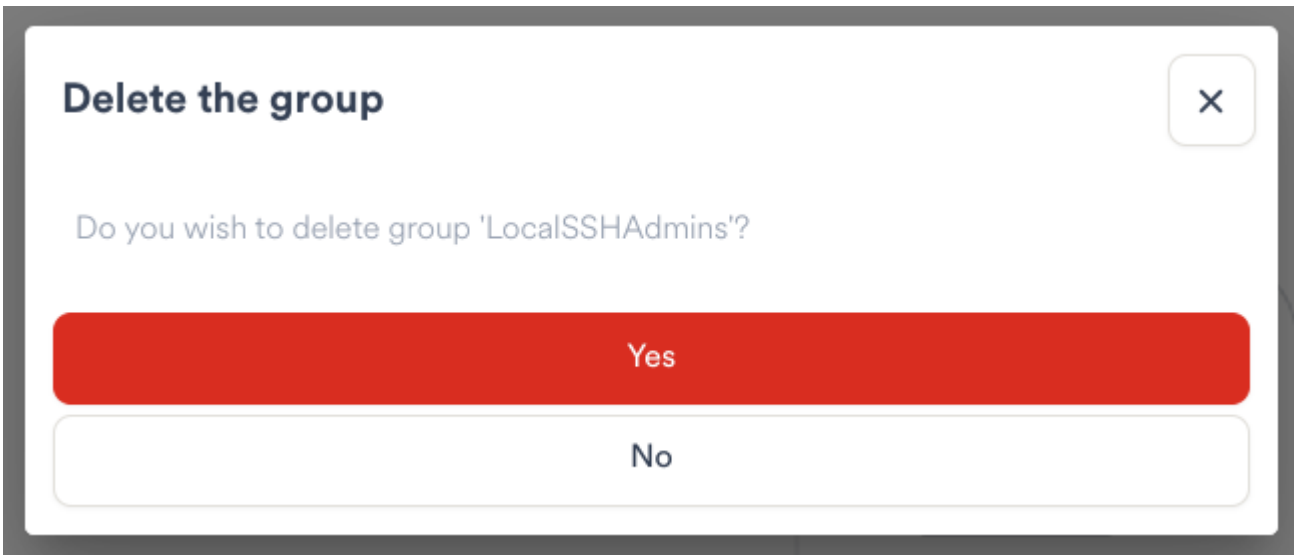


This will create a unix group on PixStor for all sites.

Deleting Groups



A confirmation dialog will be displayed



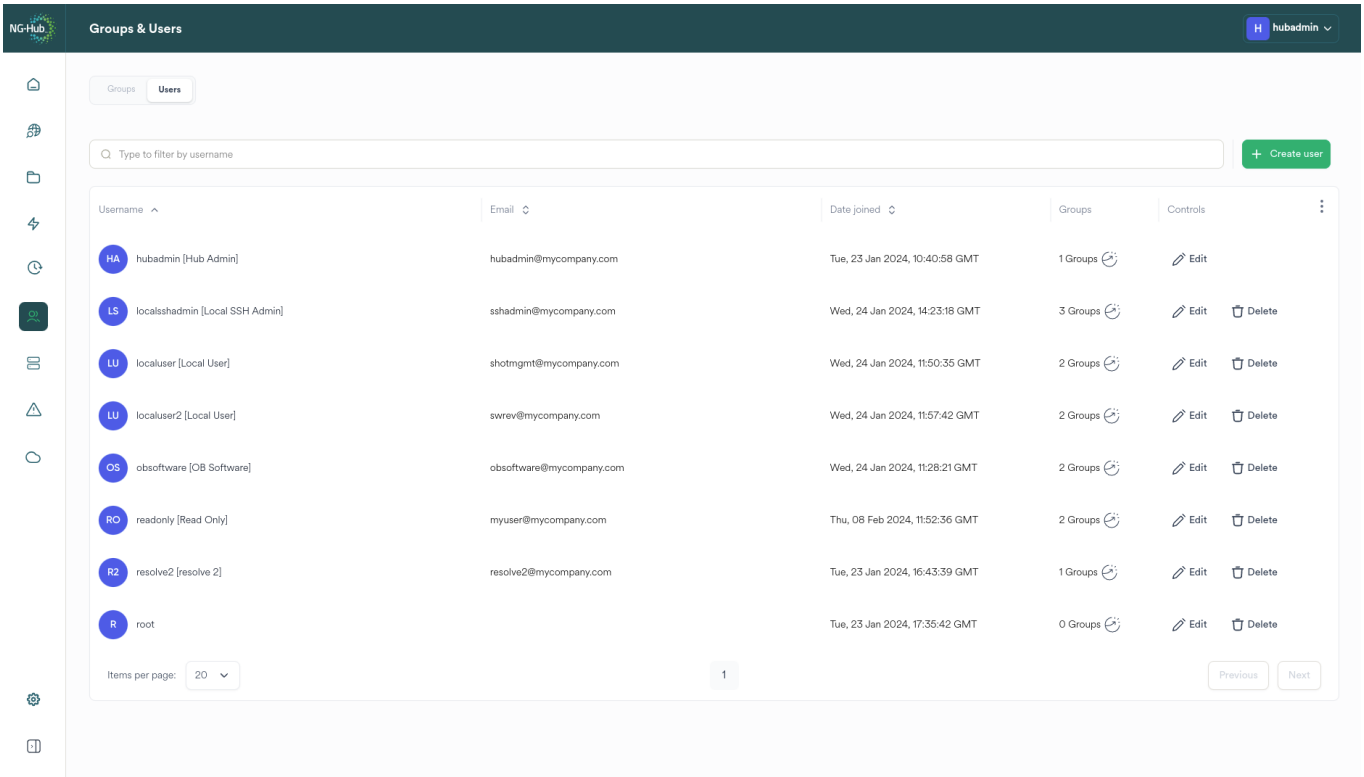
- Click “No” to close the confirmation dialog. The group will not be deleted.
- Click “Yes” to confirm and remove the group.

The default groups `Administrators` and `Users` cannot be deleted.

Viewing Users

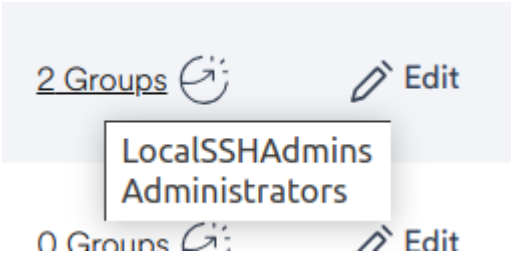
To view the available Users managed in Hub, select the Users tab.

Important: User management can only be performed by a Hub Administrator.



The Users tab provides the following actions:

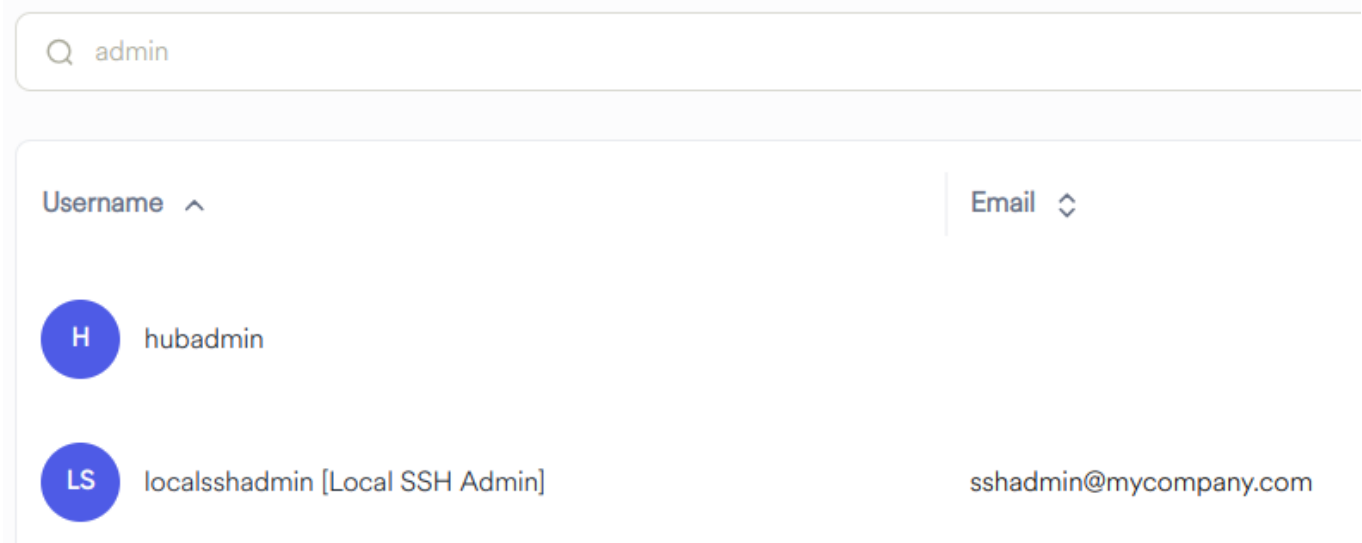
- User creation
- User deletion
- Group member management



Hover over the Groups column to see what groups a user belongs to

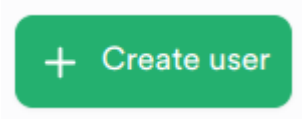
Filtering the Users View

To display Users matching keywords, enter the keywords in the filter bar.



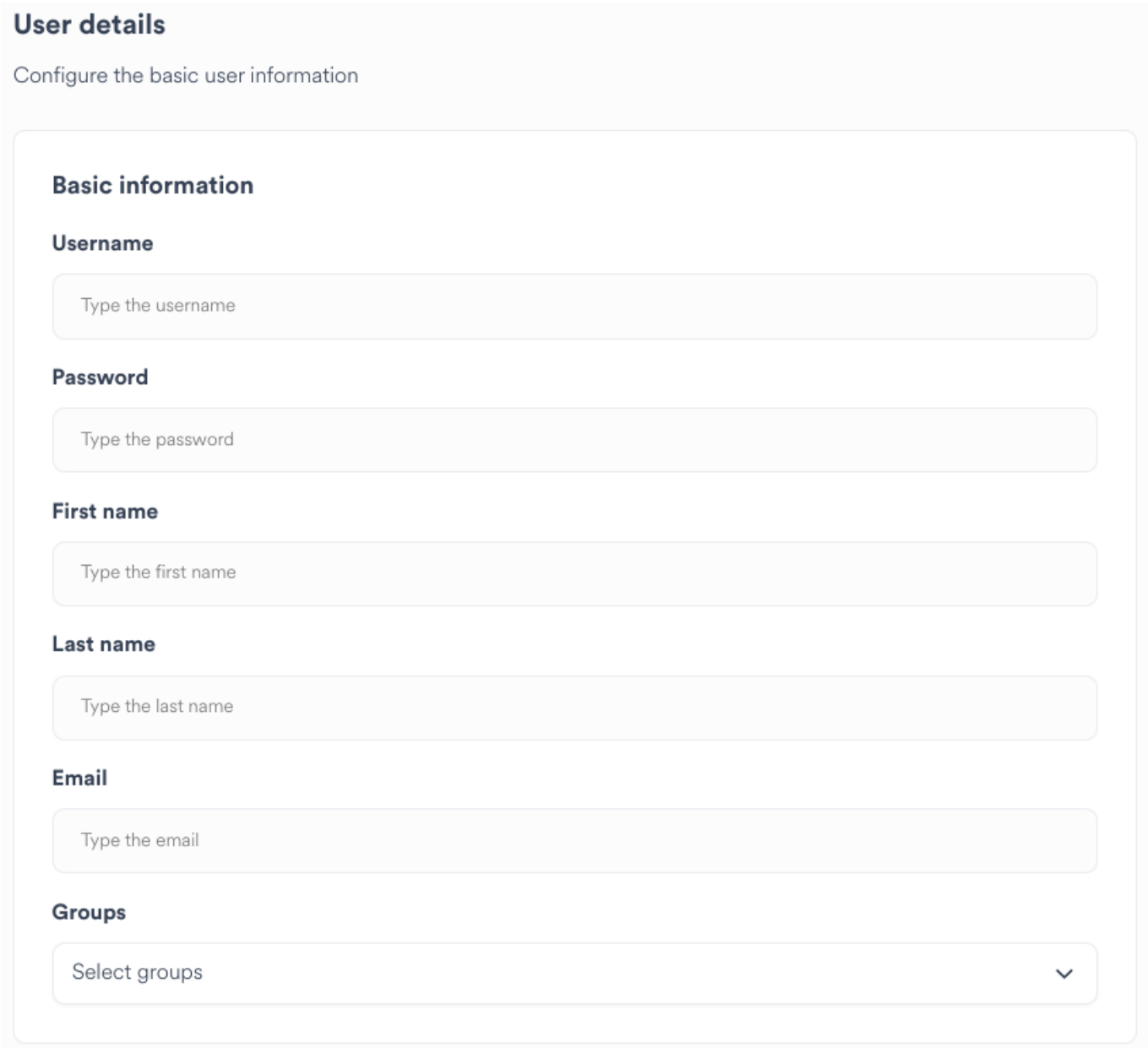
Creating Users

Click the Create User button to display a dialog to configure a new user.



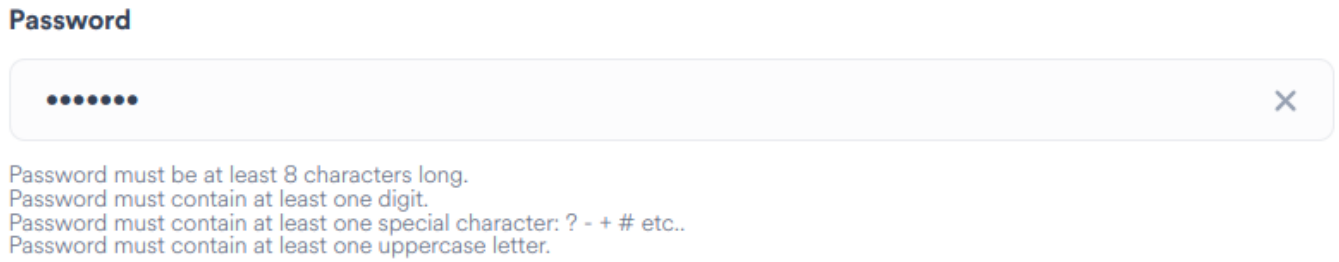
Important: This function can only be performed by a Hub Administrator

Basic User Options

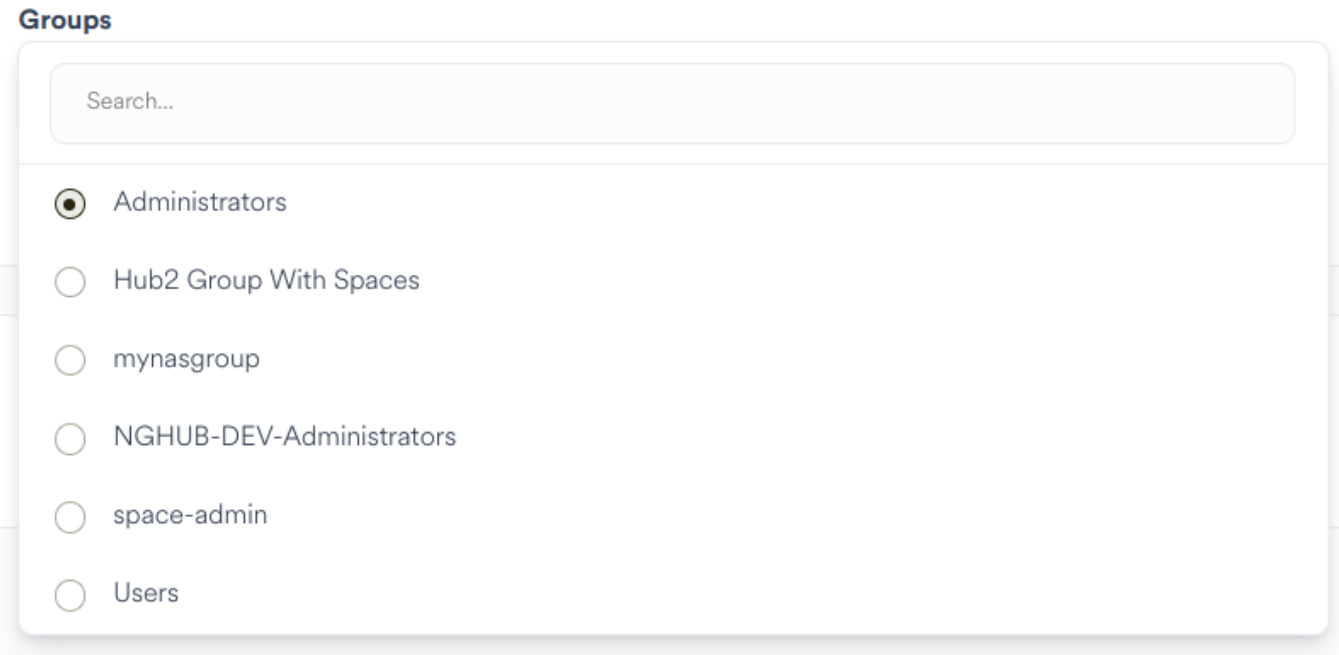


- Enter a name for the user. The name is case-sensitive and may not contain whitespace
- Enter a password
- Enter First and Last name for the user
- Enter an email address for the user

The password must meet all the requirements which are listed below the password box



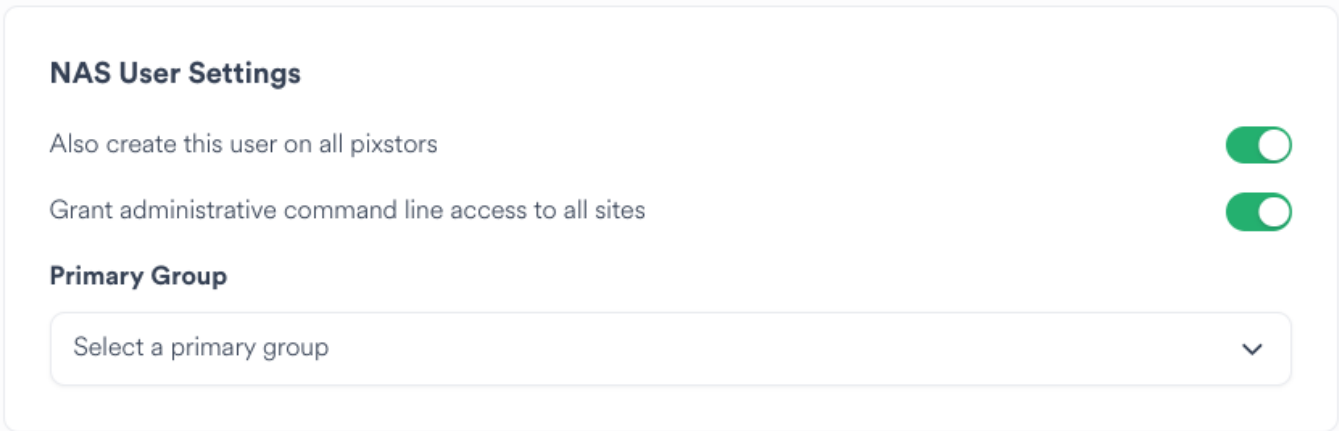
Select zero or more groups to assign the user to



- Click a group name to select it
- Click the group name again to deselect it

NAS User Settings

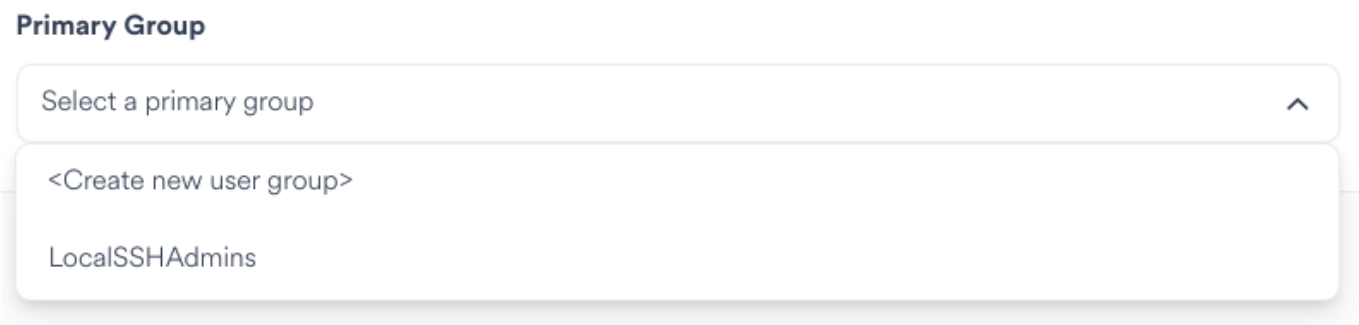
Select whether the user should be created as a local NAS user.



This will create a unix user on PixStor for all sites.

Select whether to grant command line access to the user. This means the user can connect to PixStor nodes via SSH.

Select the Primary Group for the user.



The dropdown will list any existing NAS groups. Alternatively, “Create a new user group” will create a new NAS-enabled Hub group with the same name as this user.

Deleting Users

To delete a user, click the delete “bin” icon in the row of that user.

A confirmation dialog will be displayed

- Click “No” to close the confirmation dialog. The user will not be deleted.
- Click “Yes” to confirm and remove the user.

You cannot delete the user you are currently logged in with.

Restricting Users from Spaces

All users in the group Users have read-only access to all Spaces.

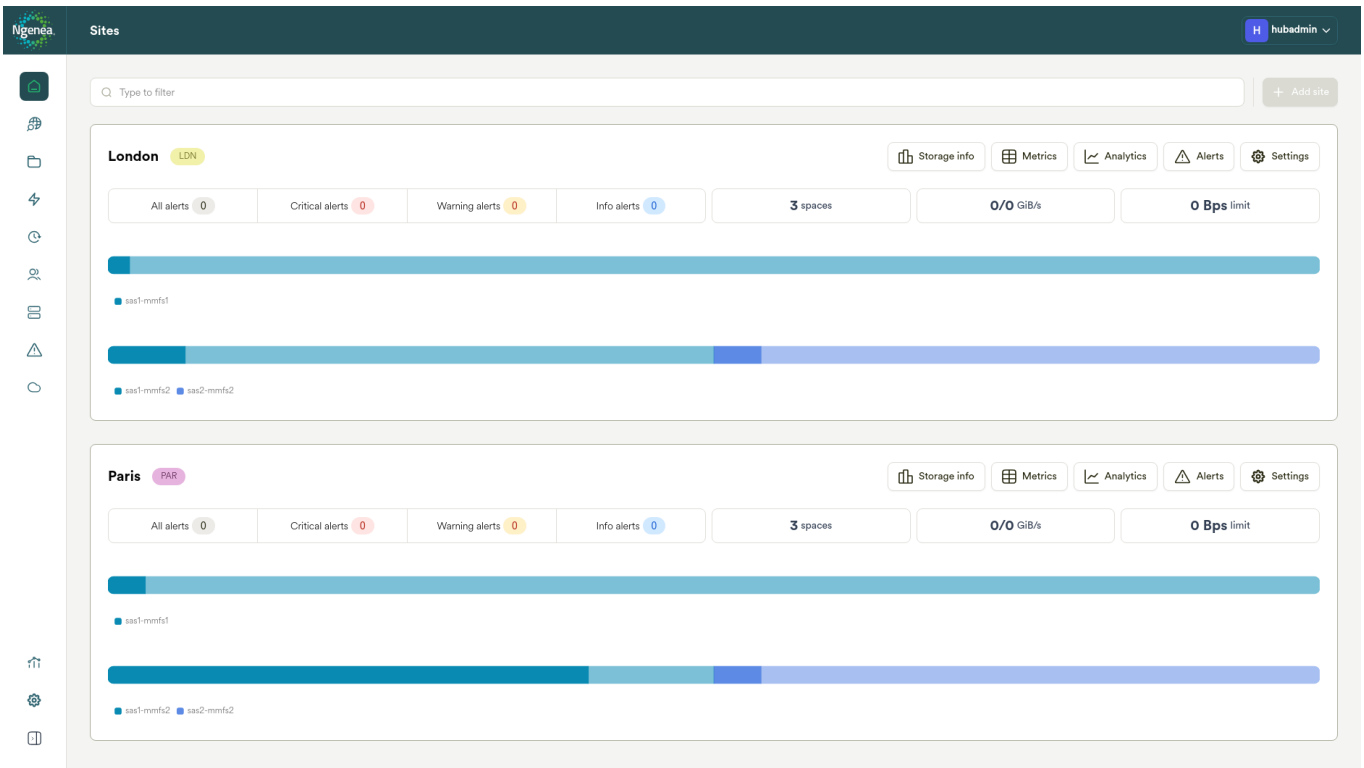
Should it be required to restrict a user from accessing specific spaces Spaces this can be achieved by:

1. Creating a new user group
2. Ensuring that only the specific spaces are assigned to the group
3. Add specific users to the group
4. Ensure the specific users are removed from the group Users

Caution: Adding a Space to a group's Administered Spaces and Used spaces will allow assigned Users to change the settings for a Space. If administrative operations are not required, do not assign Spaces to Administered Spaces - create an additional group to allow specific users to administrate specific spaces.

Sites

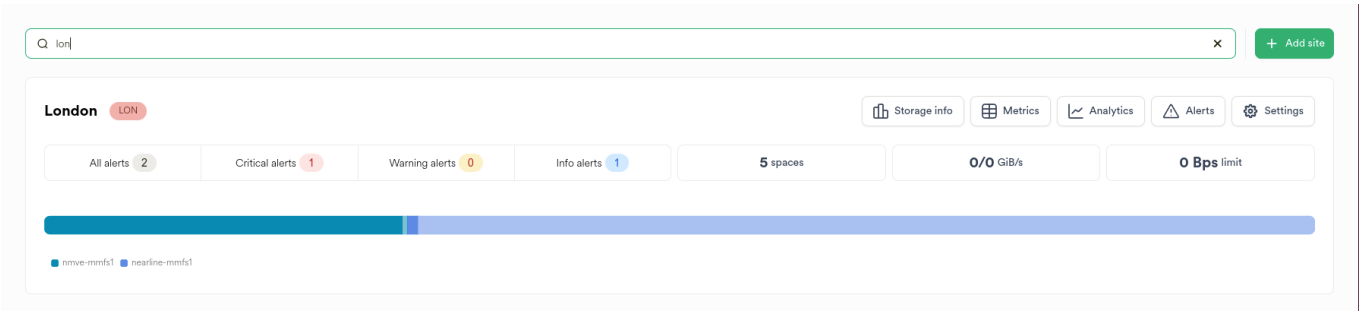
A site is a physical or cloud based pixstor server managed by Hub.



Filtering Sites

To filter for a site within the list of sites enter a keyword in the Search for…

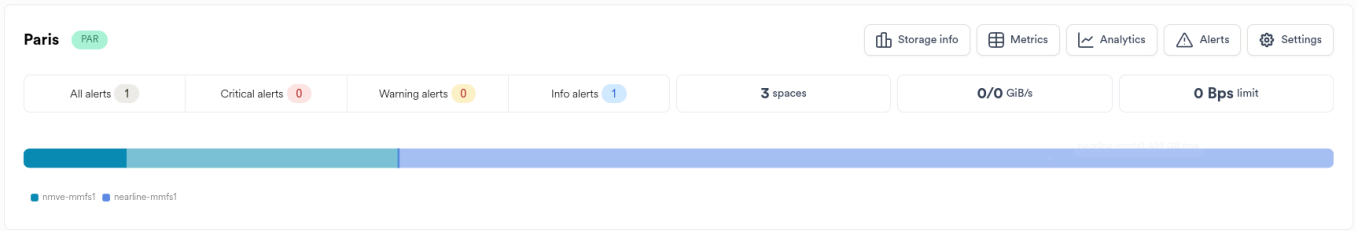
The displayed sites is limited to the sites which match the keyword(s).



The Site Card

Each site is displayed as a site card.

A Site is displayed as a card in the Sites view.



A Site Card comprises:

Site Name

London LON	Displays the designated friendly name of the site with the Ngenea site chip

Site Summary

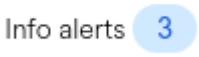
The site summary displays high level totals for the number of files and folders present, the Ngenea hydrated and dehydrated states and the number of Spaces the site hosts.

Alert Types

Numbered filter buttons display the count of each type of Alert for the Site.

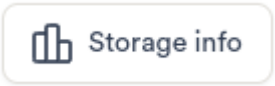
Select a button to open the Alerts for the Site, filtered for the specific Alert type.

All alerts 6	Click the All alerts button to view all alerts
Critical alerts 3	Click the Critical alerts button to filter for all Critical alerts
Warning alerts 0	Click the Warning alerts button to filter for all Warning alerts



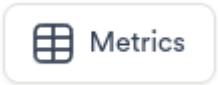
Click the Info alerts button to filter for all Info alerts

Storage Info



Click the Storage info button to open the Storage Info browser for the Site.

Site Metrics



Click the Metrics button to open the pixstor nexus site metrics in a new browser tab.

Site Analytics

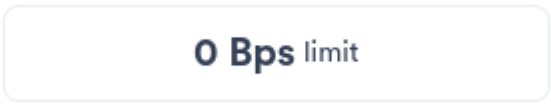
Click the Analytics button to open the pixstor nexus site analytics in a new browser tab.

Site Alerts

Click the Alerts button to open the Alerts for the Site.

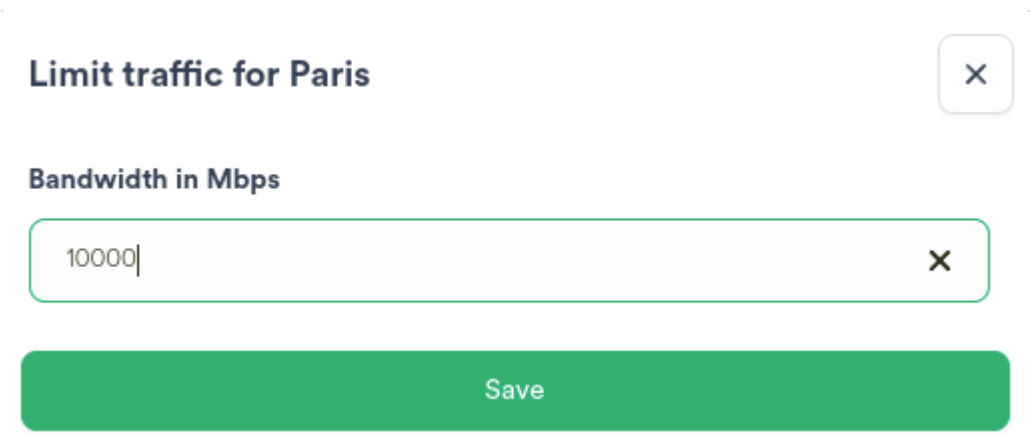
Bandwidth Control

The bandwidth of a site can be limited to a defined value via the UI. The current value is observed on the bandwidth limit button.



Click the bandwidth limit button to display the bandwidth control dialog.

The bandwidth control dialog allows limiting the bandwidth of a site to a defined value. Enter the limit in Megabits per second (Mbps) and press Save to apply the limit.

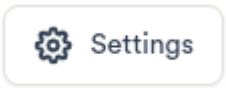


Hint: If the bandwith for a site has been inadvertently set do not press the Save button, instead click off the Bandwith limit dialog to the main area of the screen.

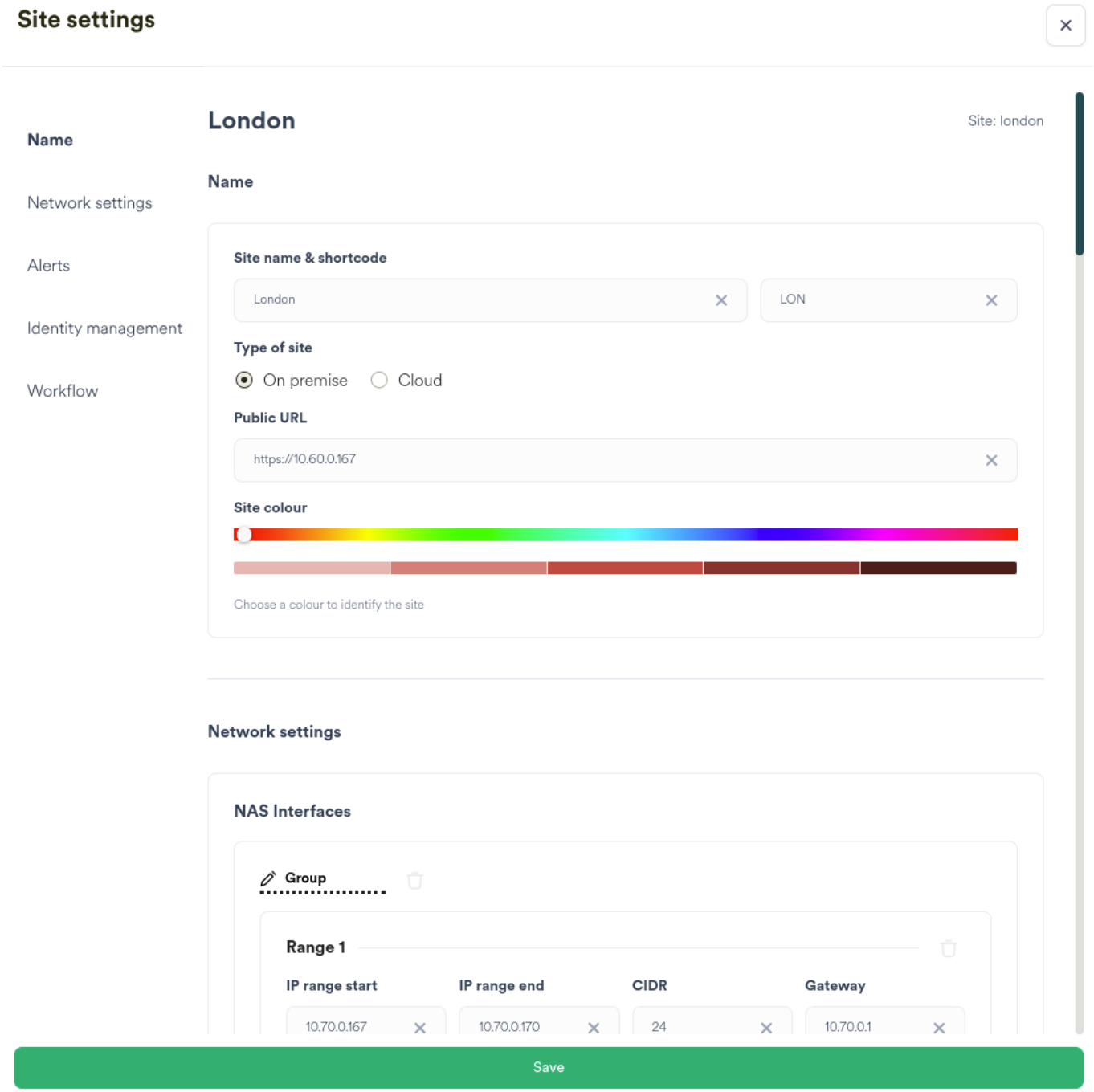
Important: This function can only be performed by a Hub Administrator.

Settings

Important: This function can only be performed by a Hub Administrator.



Click the site's Settings button to display a dialog to configure the selected site.



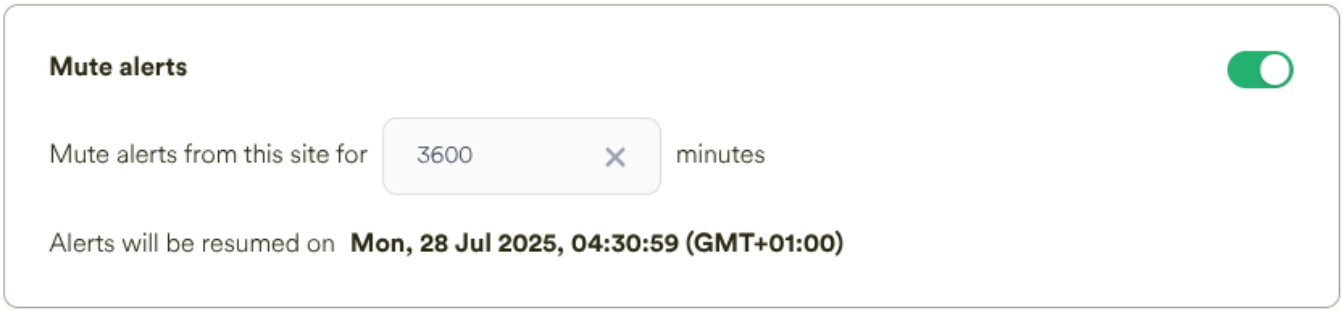
- Modify the *Site* settings as required. Refer to Adding a Site for settings guidance.

Maintenance Mode

Maintenance mode settings can be found under the Alerts section.

While maintenance mode is enabled for a site, all alerts will be muted for that site.

Maintenance mode automatically ends after a configured timeout.



- Set how long maintenance mode should be enabled for in minutes
- The time when maintenance mode will end and alerts will resume is displayed
- Maintenance mode begins when site settings are saved

If maintenance mode is already enabled, the end time will be displayed.

- Switch the toggle off to end maintenance mode before its configured end time
- Click “edit” to extend or shorten the maintenance window

Pool Space

One or more pixstor storage pools which comprise the pixstor file system are represented.

Hovering over the pool percentage bar provides the remaining capacity for the pool.

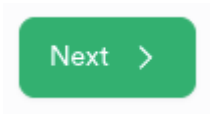
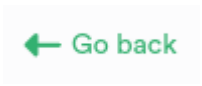
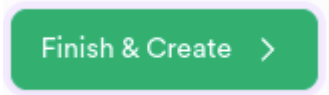
Adding a Site

Add Site Wizard

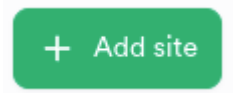
Hub allows remote configuration of all participating pixstor sites.

New sites are automatically joined to Hub awaiting optional configuration via the Site Wizard.

Navigating the Wizard

	Click the close button to exit the wizard. Changes are not saved.
	Click the Next button to advance to the next page of the wizard. The Next button is disabled until all required page elements are completed.
	Click the Go Back button to return to the previous wizard page.
	Click the Finish & Create button to apply the changes displayed on the wizard summary page.

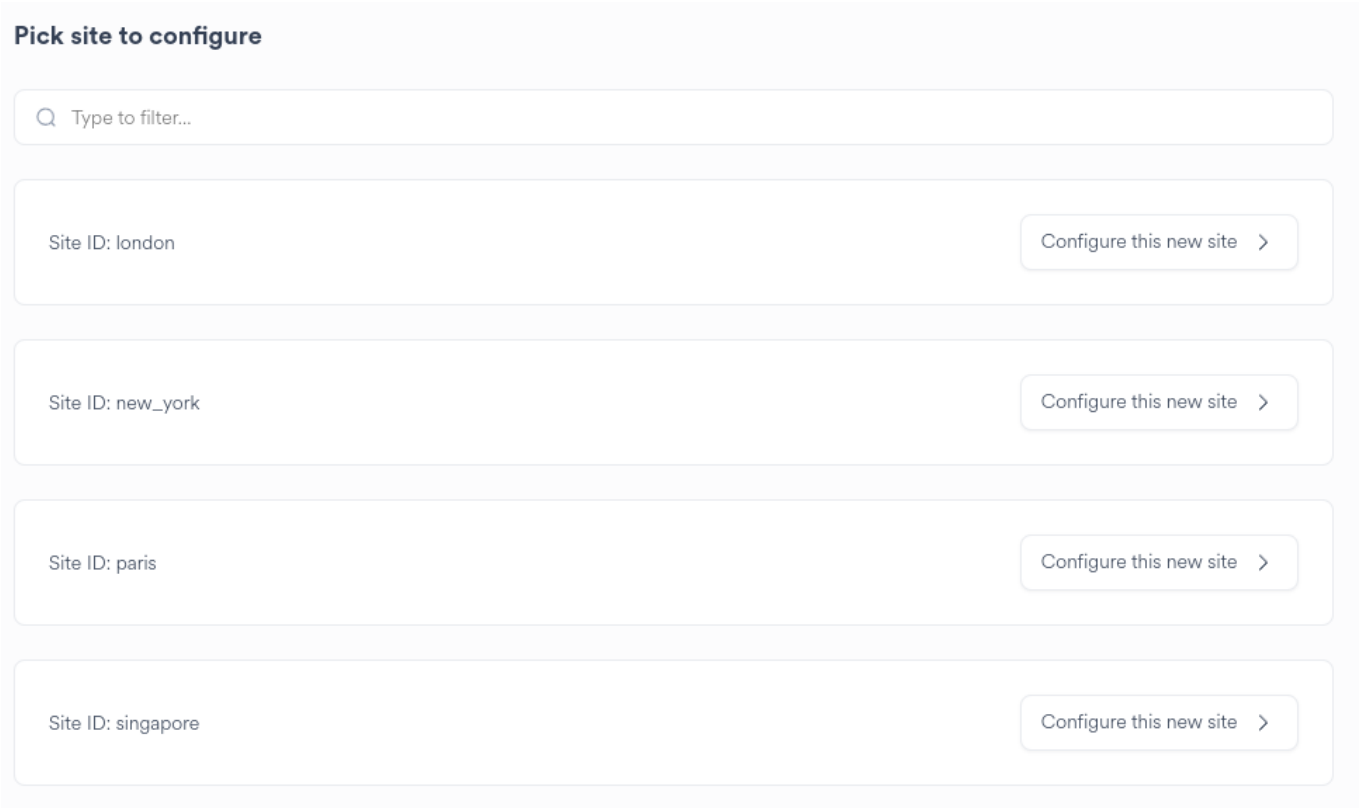
Add Site

	Click the Add site button to display a dialog to configure the selected site.

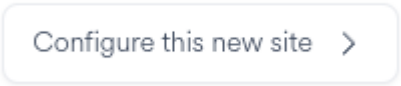
Important: This function can only be performed by a Hub Administrator.

Pick site to configure

Sites which have been automatically registered to Hub but not yet configured are shown:

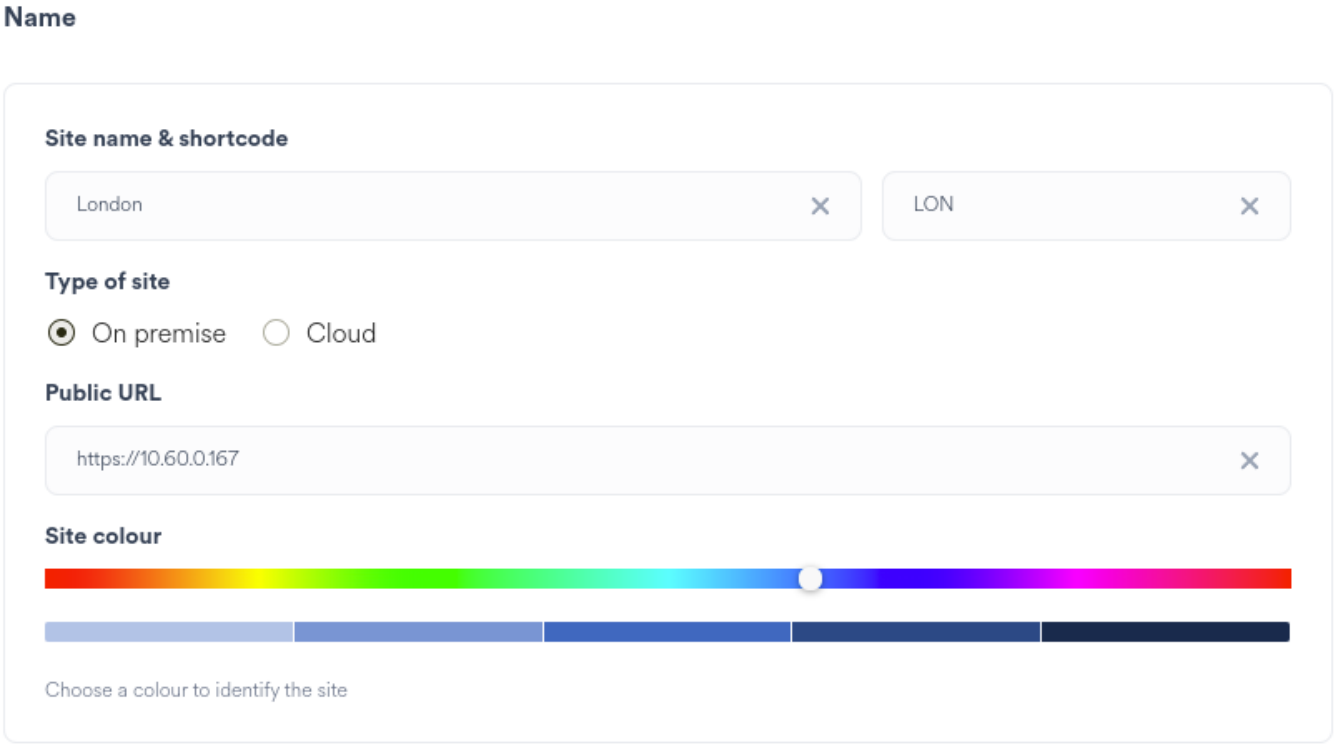


- Select a site to configure



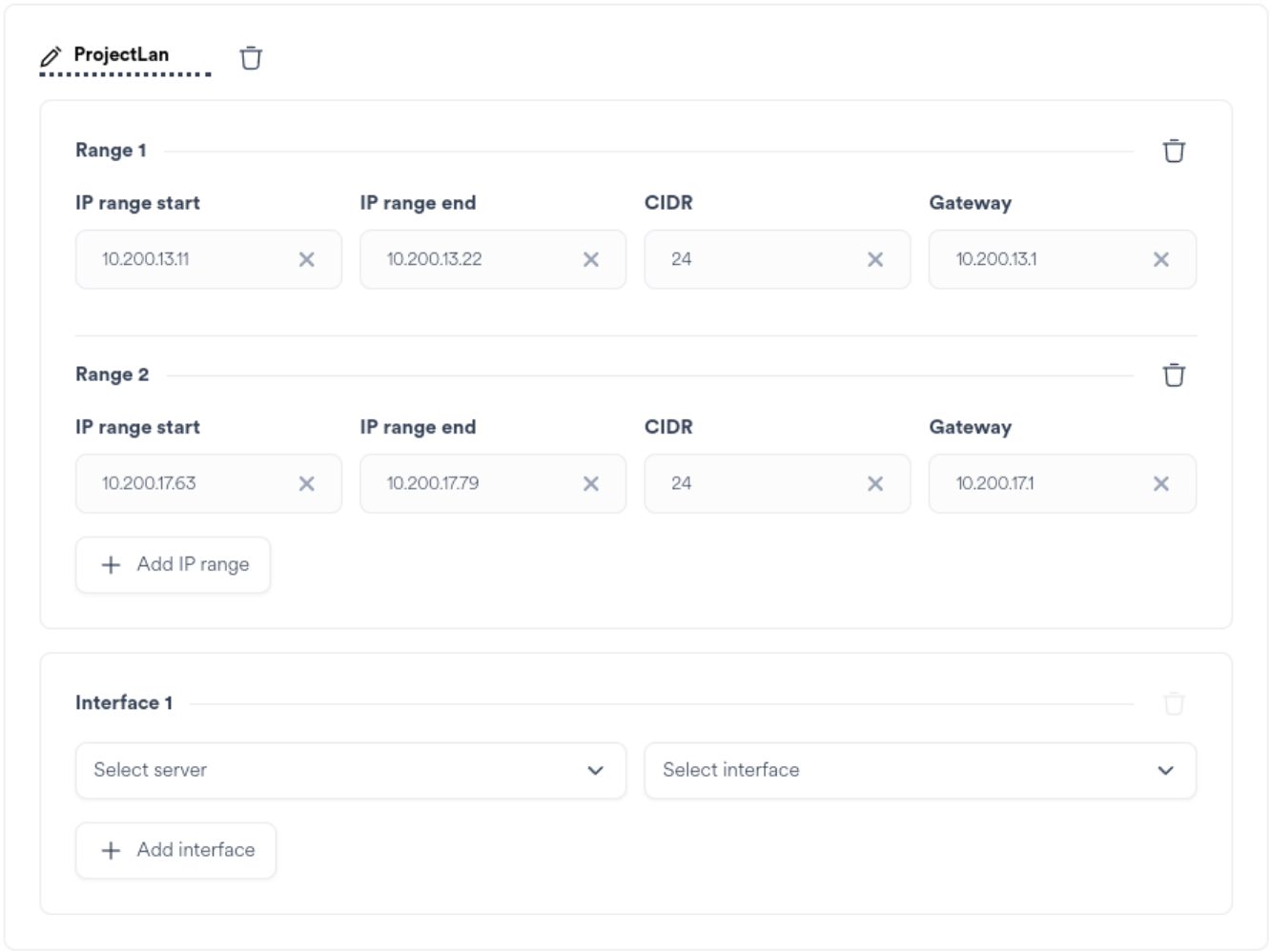
Click the Configure this new site button to proceed

Site Name



- Provide a friendly name for the Site
- Provide a 3 character short code for the site. The shortcode is displayed as the label on the Site's chip. E.G. LDN for London
- Specify whether the site is on-premise or a pixstor cloud deployment. Each site type provides different Network Setting options.
- Provide the URL IP or FQDN which refers to the pixstor management node of the Site
- Select a colour for the Site chip

Network Settings (On premise)



- Add the required IP address or network range and specify a valid CIDR mask to apply the restriction
- Specify a gateway, if required
- Select the server(s) and interface(s) of the server where the IP range will be configured

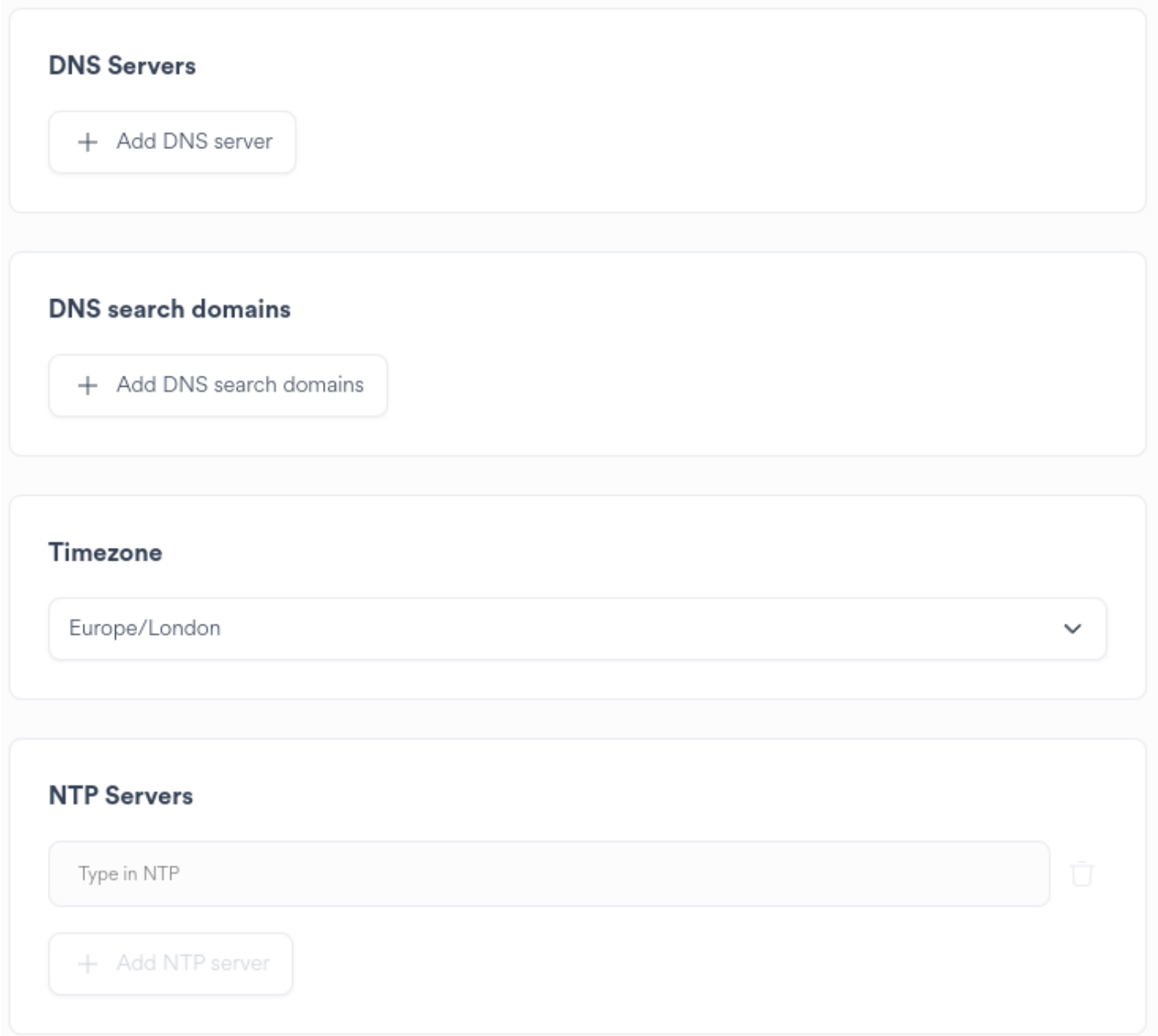
+ Add IP range	Click the Add IP range button to add additional restrictions
+ Create another group	Click Create another group to add additional IP range to interface mapping groups
	Click the delete button to remove an IP range or Interface group

Network Settings (Cloud & General)

pixstor cloud systems use predefined network architectures.

Unlike on-premise pixstor systems there is no requirement to create IP ranges or interface groups. IP addressing is externally managed by the cloud / virtual environment.

Both on-premise and cloud systems share common network configuration for DNS, Timezone and NTP.



- Specify the IP address or FQDN hostname of one or more DNS servers
- Specify one or more DNS search domains
- Specify the Timezone in which the server resides, or will participate in
- Specify the IP address or FQDN hostname of one or more DNS servers

Hint: If the pixstor site will be joined to an external Identify Mapping service such as Active Directory or LDAP, best practice is to ensure that the DNS and NTP servers match those of the service, or point at the service hosted DNS and NTP if it provides such capabilities. Should the pixstor become out of time sync with the Identify Mapping service login failures can occur.

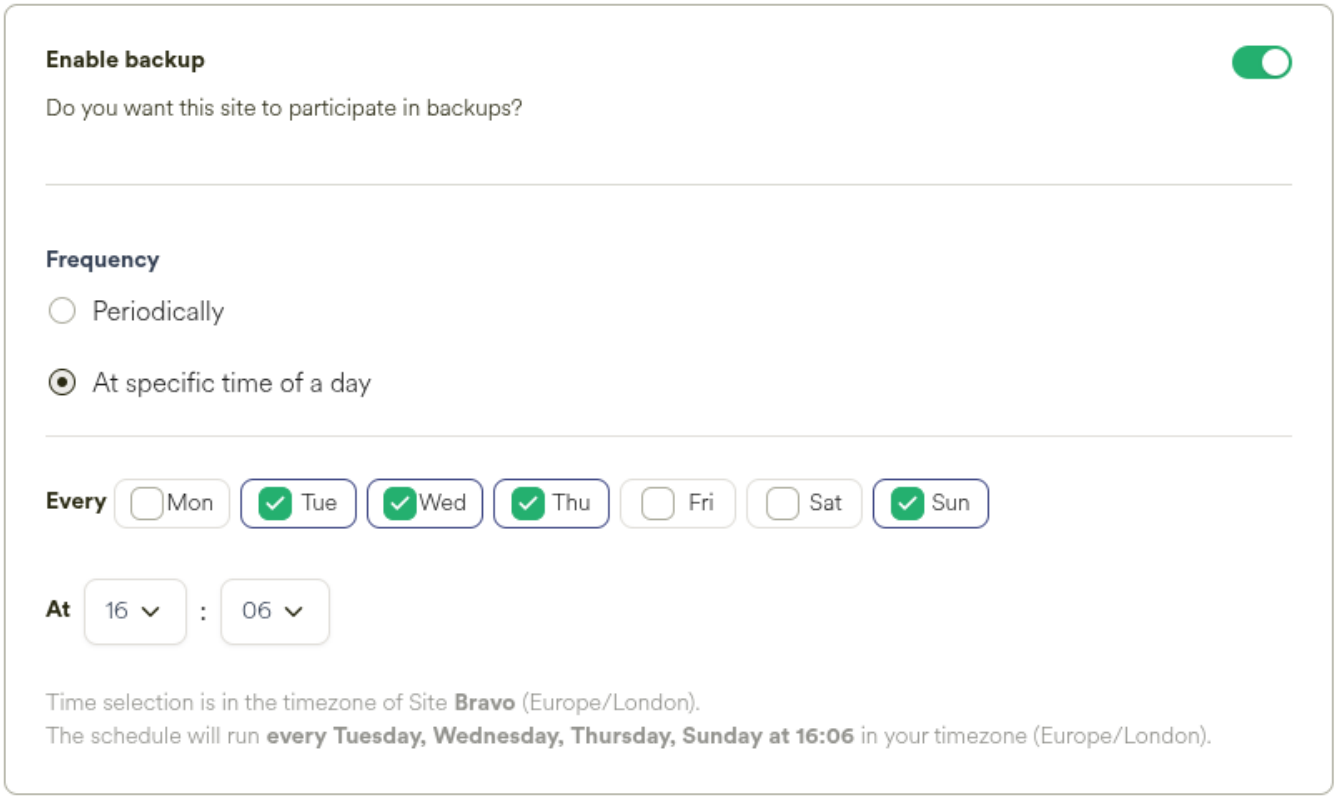
Backup

pixstor provides the capability to backup data within a Space on a per-Site basis to specific Ngenea Targets.

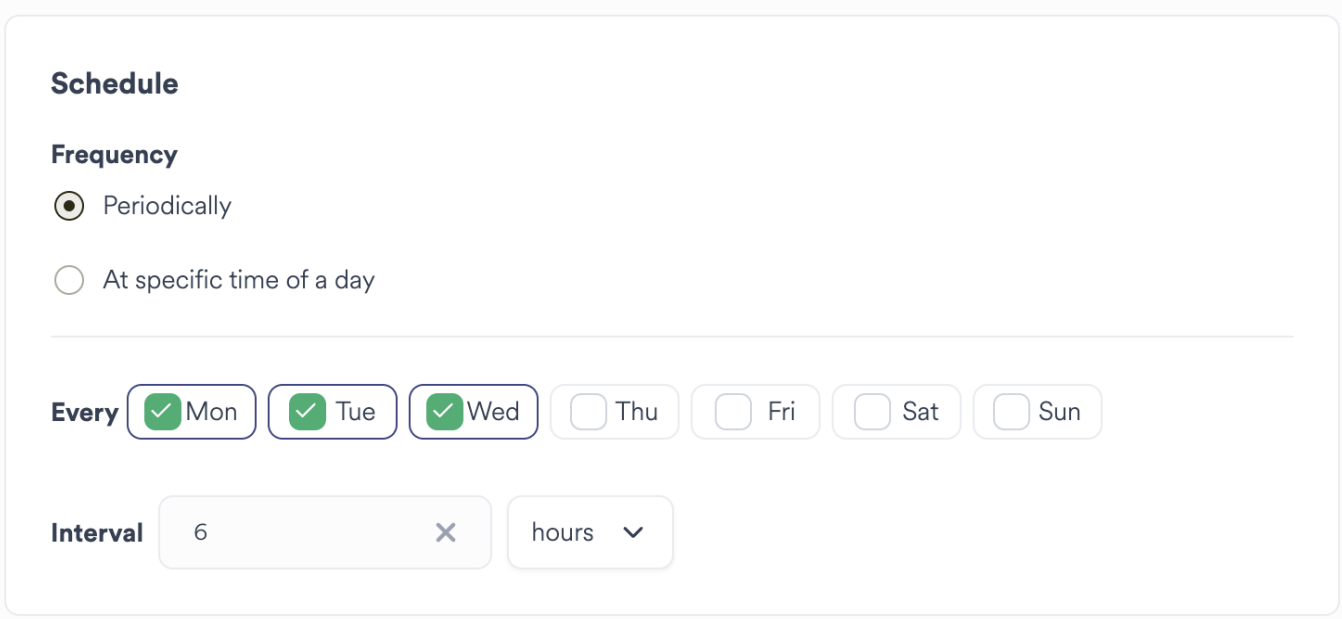
Hub enables configuration to be set for the Ngenea Backup service running on pixstor sites.

If the Site is enabled to participate in backups, each Space requires additional configuration to enable the per-Space backup.

For more information refer to Backups.



Determine the required frequency of the backup.

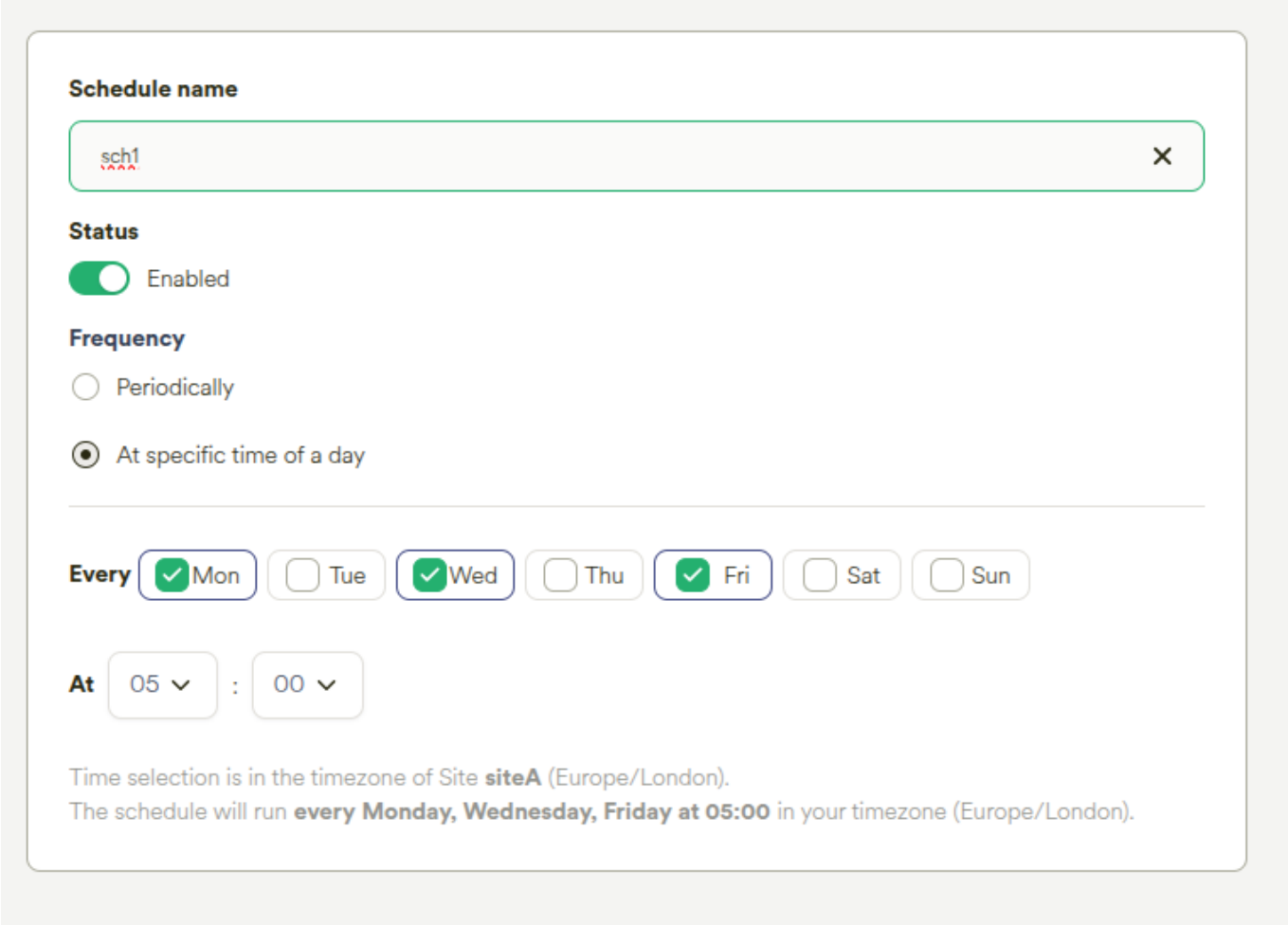


Choosing Periodically will ensure that the schedule will run on the next interval set.

E.G:

- 1 hour: The backup will run on the next hour (12.00, 13.00)

- 15 mins: The backup will run on the next 15 minute interval past the hour (15, 30, 45, 00)

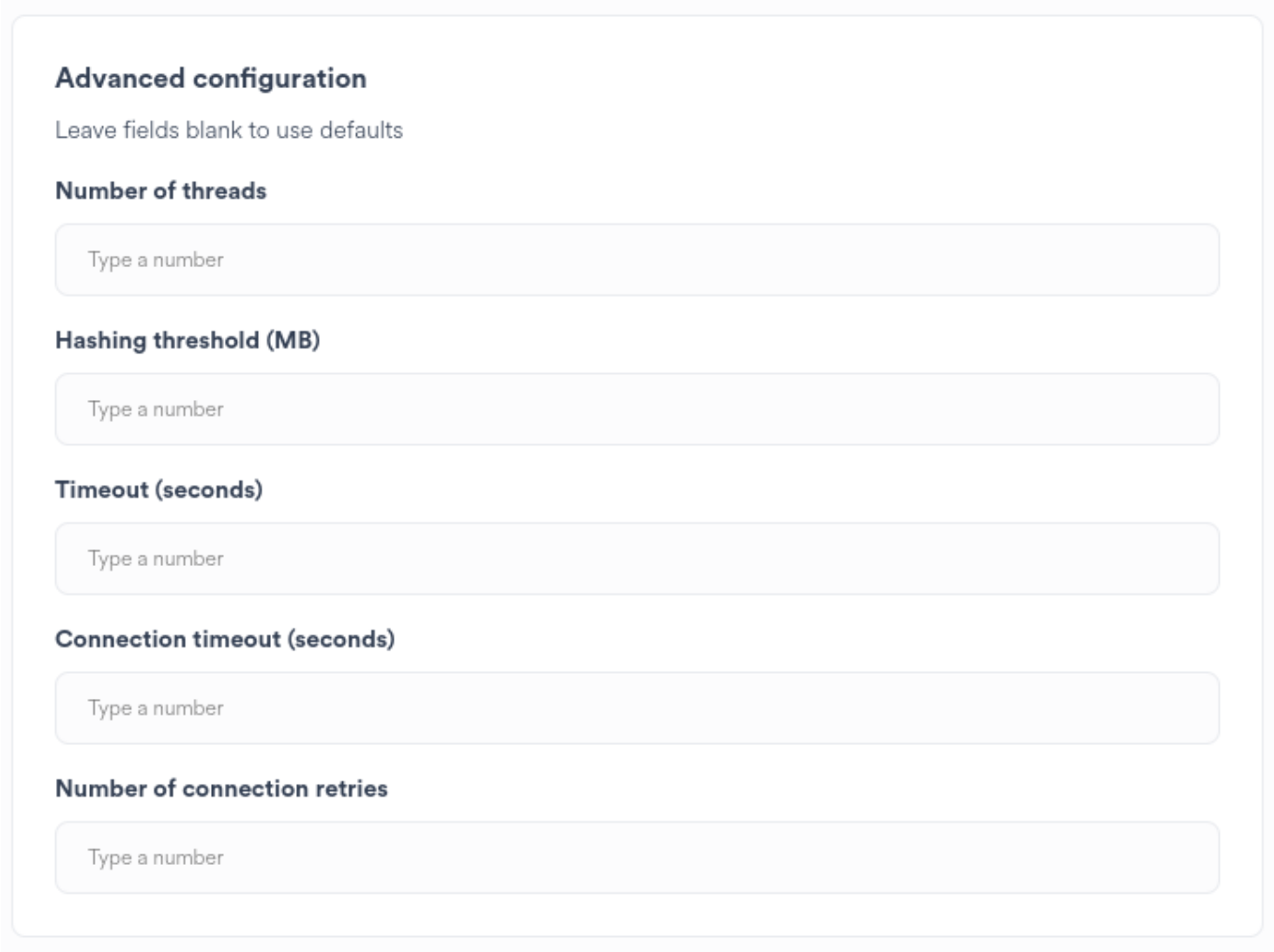


Choosing At Specific Time of A Day allows the Policy to be scheduled once per chosen day at a specific time of day.

Hint: The schedule time is set in the Site's local timezone, but will be stored in the UTC timezone.

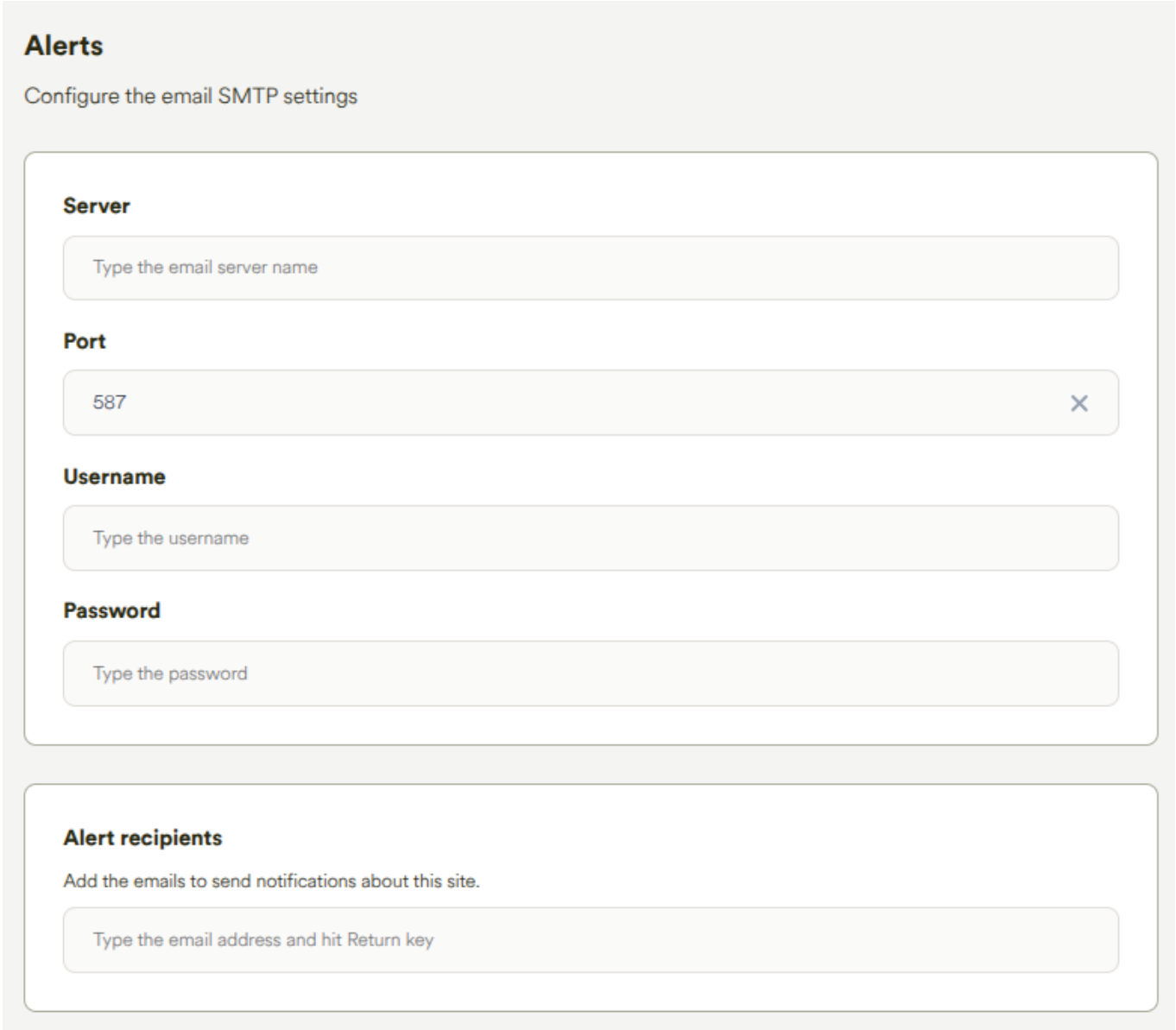
Optional tuning can be set for the backup operation in the Advanced Configuration.

Refer to the Ngenea Backup documention prior to applying any parameters.

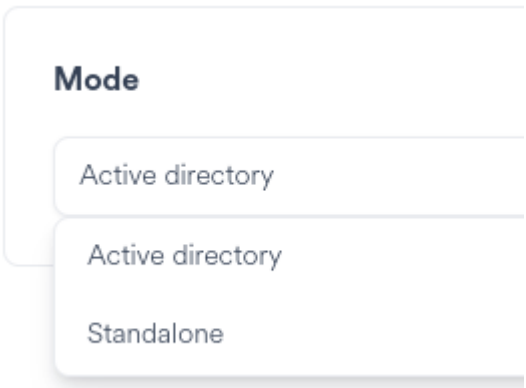


Alerts Settings

pixstor provides the capability to notify an inbox if service issues arise.



- Specify the SMTP configuration of an email server to which to send notification emails
- Specify one or more valid email addresses to receive the notification emails

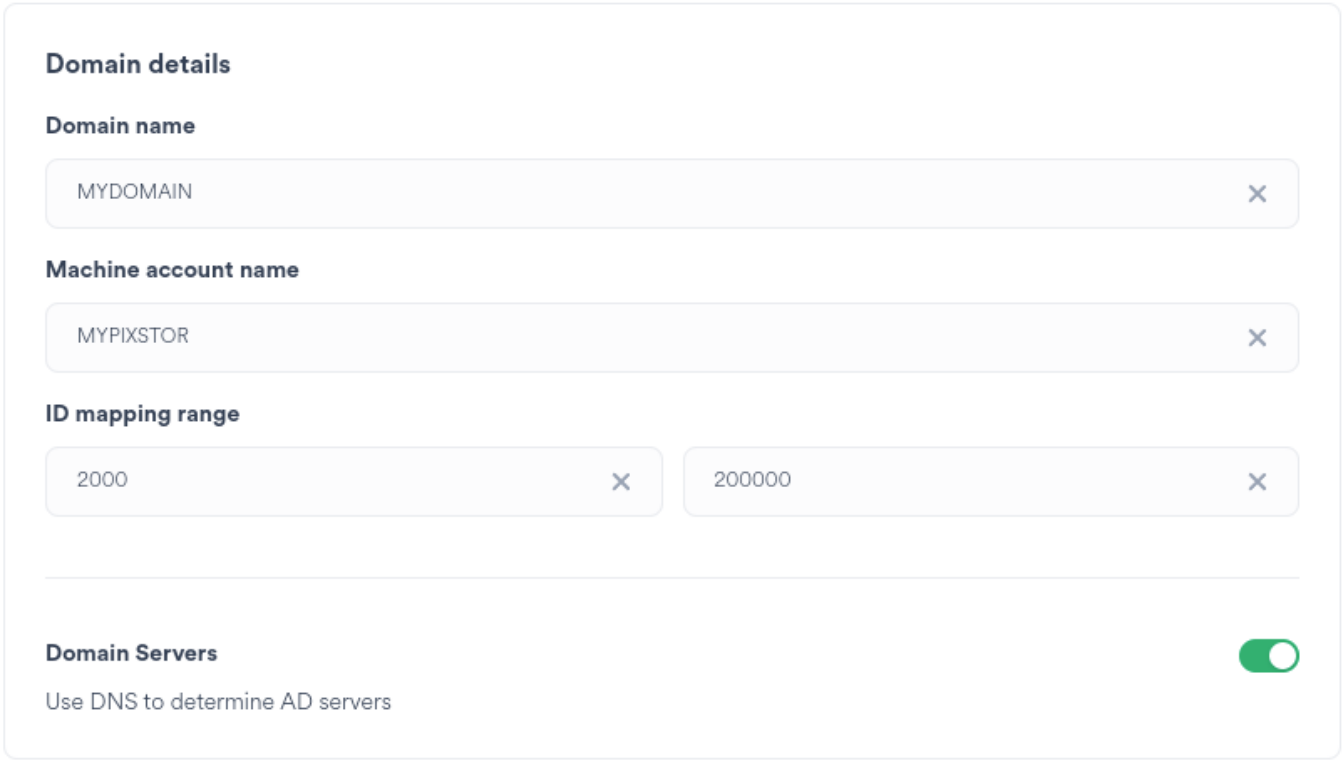


Identity Management

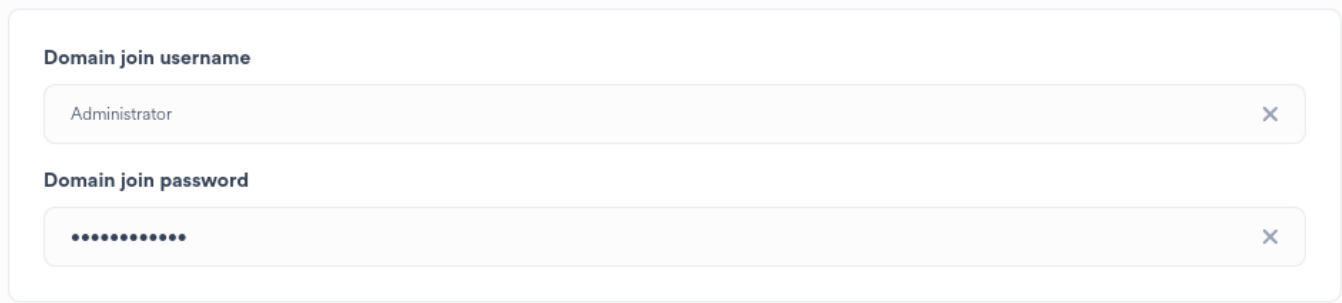
- Specify the Identity Management mode as appropriate:

Mode	Description
Active Directory	pixstor uses RFC2307 compliant identity mapping with Active Directory

Mode	Description
Standalone	pixstor generates local UIDs and GIDs mapped to Active Directory SIDs

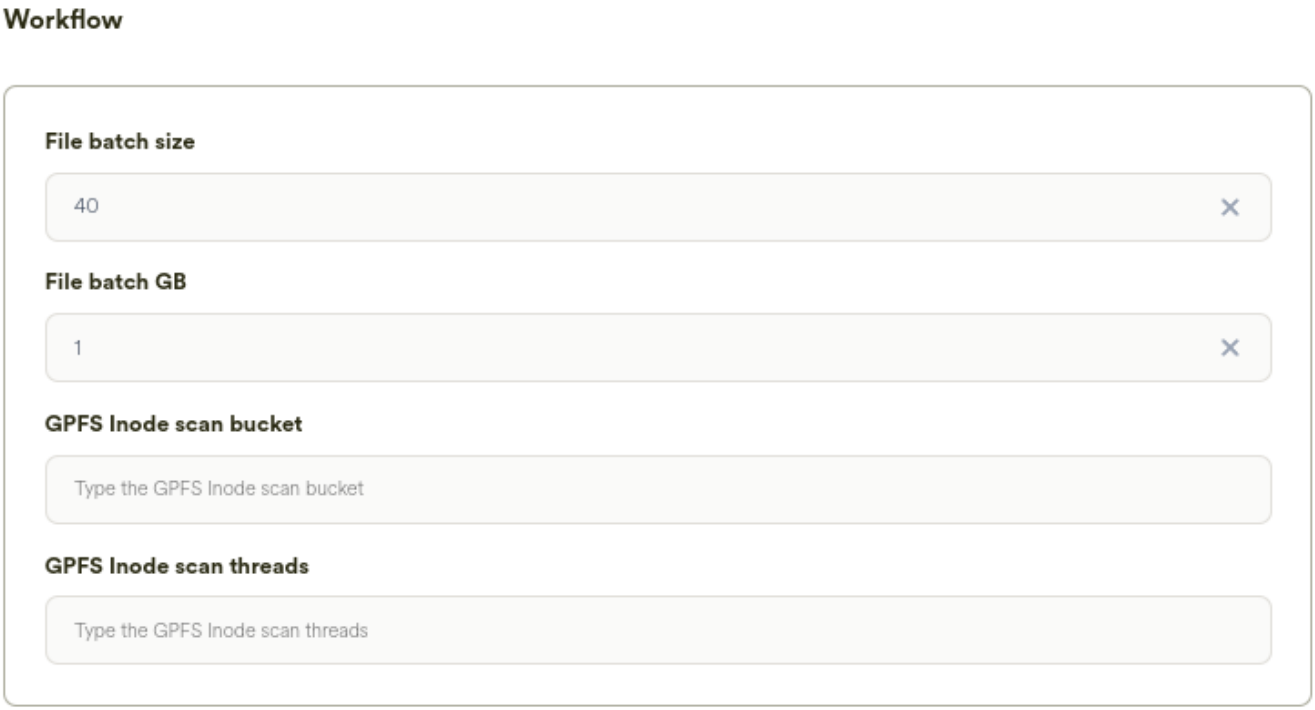


- Specify the Domain to join
- Specify the Machine account name
- Specify the ID range to map to
- Specify whether to use DNS to locate an Active Directory Domain Controller or alternatively specify an IP address or hostname



- Specify a valid username and password with domain join capability

Workflow



File Batching

Whichever limit results in a smaller batch size applied. For examples given 100 files of 500MB each, a File batch GB of 1 and a 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).

File batch settings apply to all workflows run on a Site.

Alternatively enable Dynamic File Batching.

For more information refer to the Administration Guide.

- Specify the File batch size

Hint: Default = 40. Increasing this value reduces the number of tasks processed in a job. Reduction of tasks in a job, for the majority of workflows, reduces the total time for the job to run due to decreasing the total task processing overhead. It is extremely recommended to start with small values and increase based on observation of system load and resources and adding too many files to collate into a single task can cause the task payload to be rejected by Hub due to the size of the payload.

- Specify the File batch GB

Hint: Default = 1. Increasing this value causes more files to be added to a task upto the total file size specified, after which subsequent files are added to the next task, and so on.

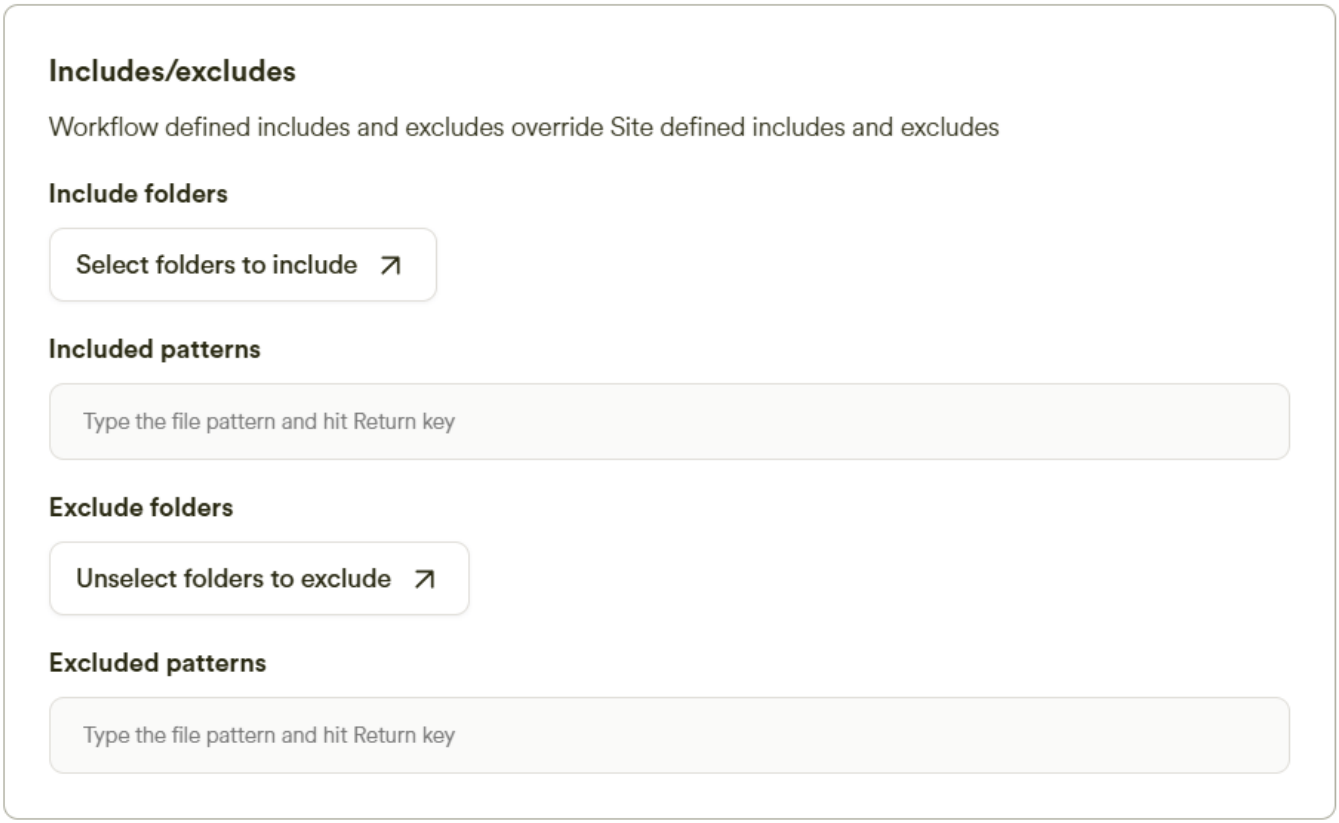
GPFS Inode Scanning

- Specify the GPFS Inode Scan bucket

Hint: Default = auto-generated per scan. The number of bucketed groups of collated inodes to be processed by the parallel scanner. An optimal value is the total number of files to be scanned divided by one million so each bucketed inode group has approximately one million files.

- Specify the GPFS Inode scan threads

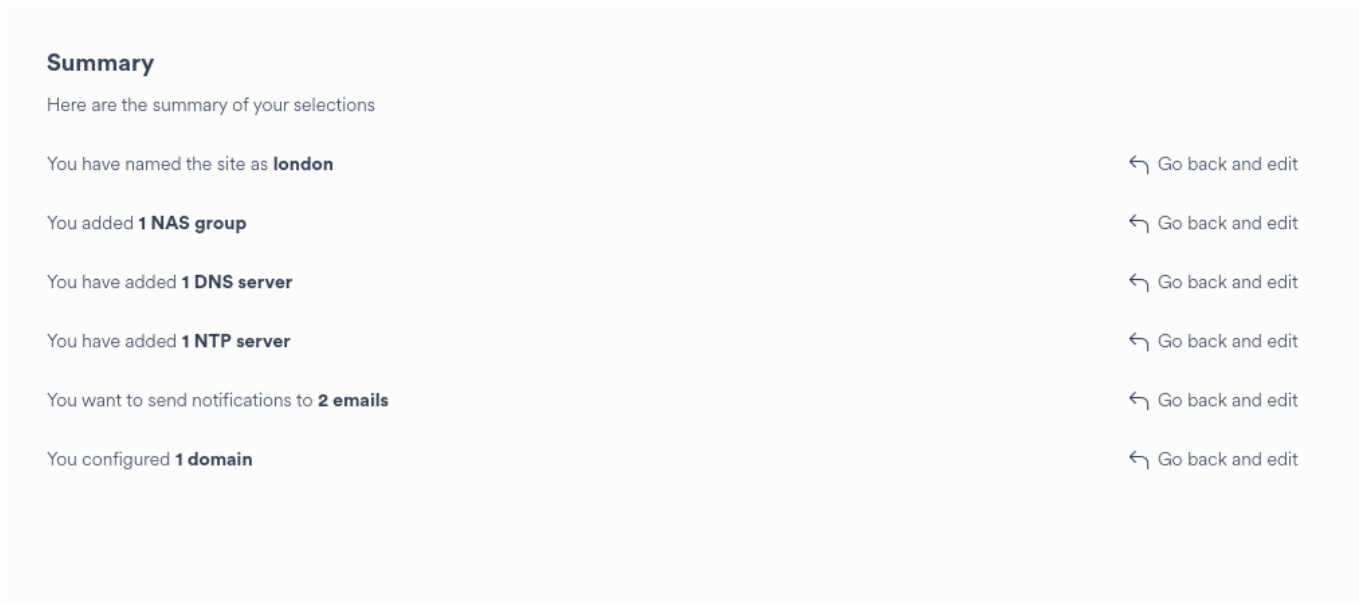
Hint: Default = 2. The number of inode scan threads. Setting this value to the number of CPU cores can provide optimal performance with non-contending workloads. It is extremely recommended to start with small values and increase based on observation of system load and resources.

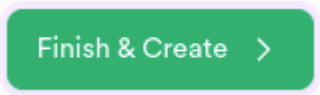


- Select folders to include in the workflows run on the Site
- Specify file pattern which should be included
- Select folders to exclude in the workflows run on the Site
- Specify file pattern which should be excluded

Summary

Upon completing the wizard steps a summary is presented:





Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Alternatively `Go back` and change the proposed configuration as required or `close` the wizard to cancel the creation of the Ngenea target.

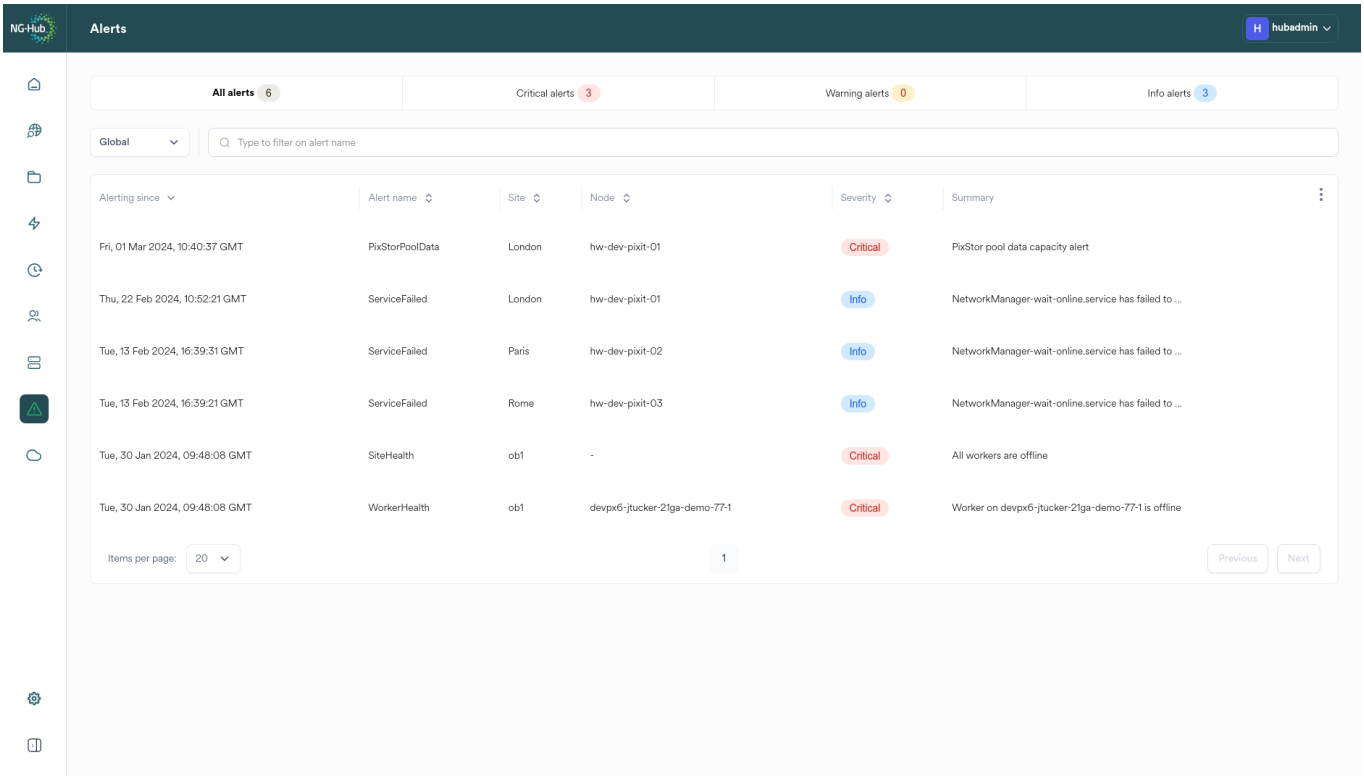
Alerts

Hub provides a view of alerts across all Sites.

- Only active alerts are displayed
- Muted alerts are not displayed

Tip: To mute alerts, refer to Maintenance Mode under Site settings

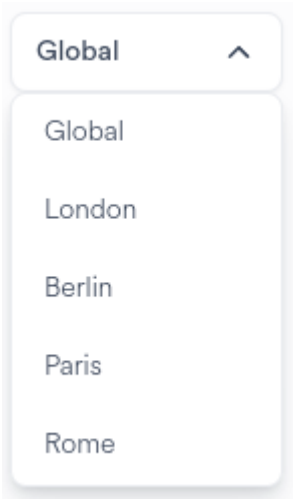
Click the alerts button to navigate to the Global Alerts screen



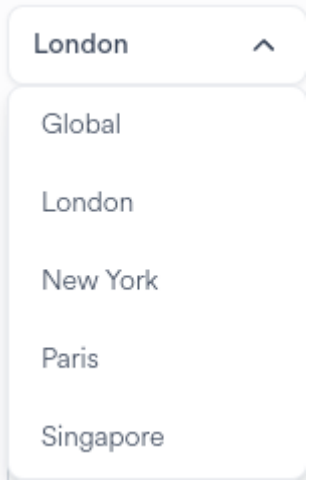
Hub provides two views of Alerts - Global and Local.

- Global displays all Alerts on all Sites
- Local displays the Alerts on a specific Site

The default view of Alerts is Global.

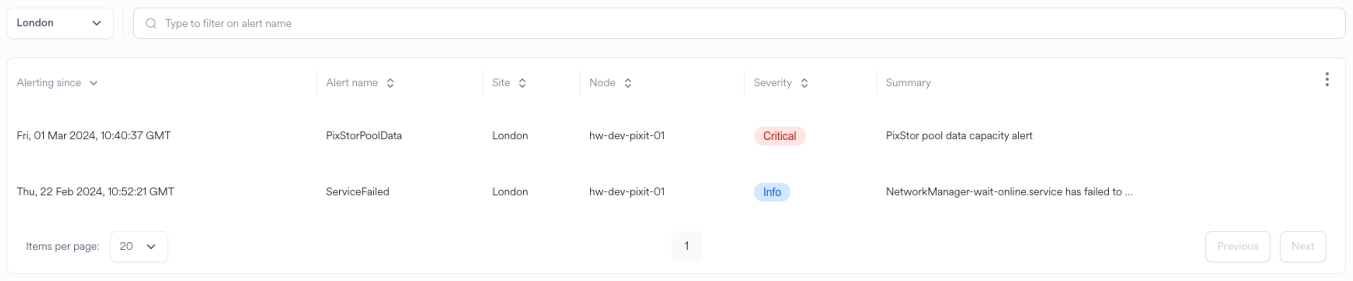


To switch to a site-centric view, select the specific site from the Alerts drop-down menu.



To switch to Global view, select Global from the Alerts drop-down menu.

Choosing a specific Site displays Alerts only from the chosen Site:



Alert Types

Above the Alert table, numbered filter buttons display the count of each type of Alert.

Select a button to filter for the specific Alert type.

All alerts 6	Click the All alerts button to view all alerts
Critical alerts 3	Click the Critical alerts button to filter for all Critical alerts
Warning alerts 0	Click the Warning alerts button to filter for all Warning alerts
Info alerts 3	Click the Info alerts button to filter for all Info alerts

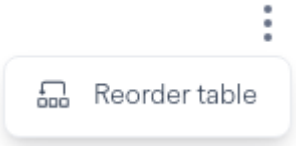
Categories

To filter for an Alert category, enter text into the filter bar.

Alerts are filtered where the Alert name matches the text in whole or part.

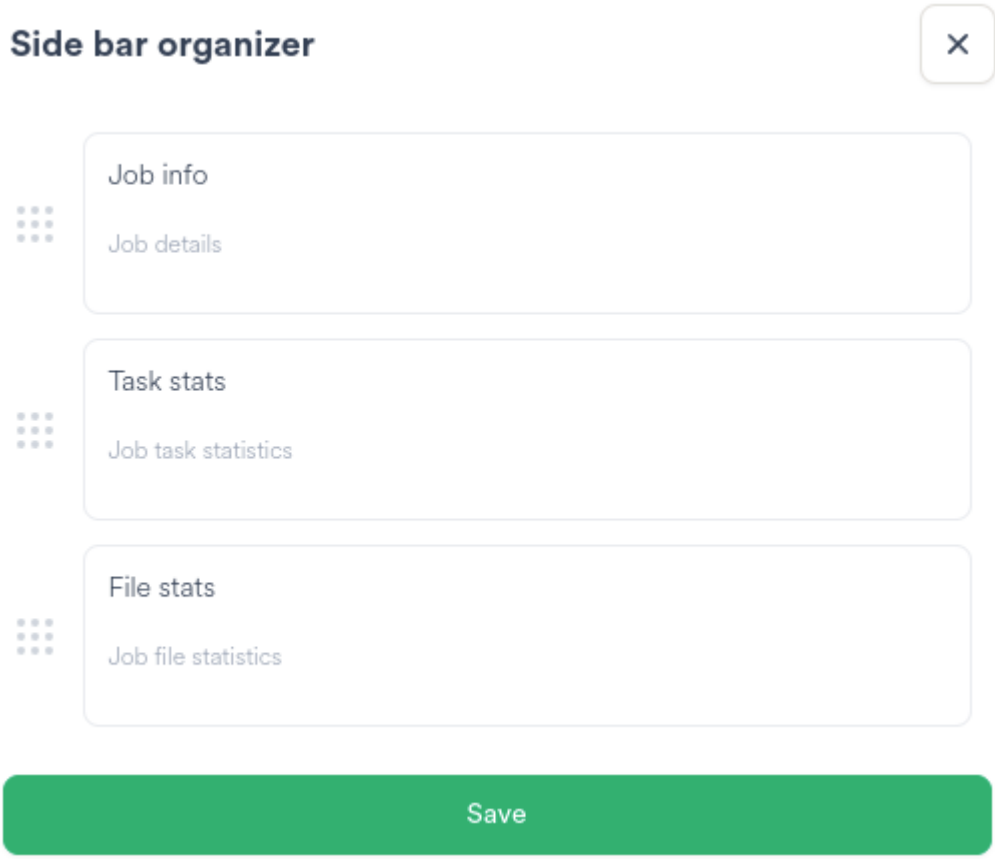
Reordering

The alerts table columns can be reorganised for user preference.



Click the vertical dots on the right hand side of the table and select the Reorder table button to open the table organizer

Re-arrange items in top to bottom order by dragging vertically up or down.



Ngenea

Ngenea 2.6+ introduces changes to how Ngenea Targets are managed. Buckets [new feature] and Targets work together to provide easy, granular mapping between Spaces and Buckets.

Ngenea provides the capability to off-site data, import data, or send data between pixstor sites.

Concepts

- A Bucket provides the storage for off-PixStor data. This may be an AWS S3, GCS, Azure Blob, POSIX or Blackpearl DS3 protocol. For versions prior to 2.6, these were previously the types of Ngenea Target.
- A Target is the mapping for how data is transferred to and from a Bucket for a Space. For versions prior to 2.6 this was available as part of the Ngenea Target setup.

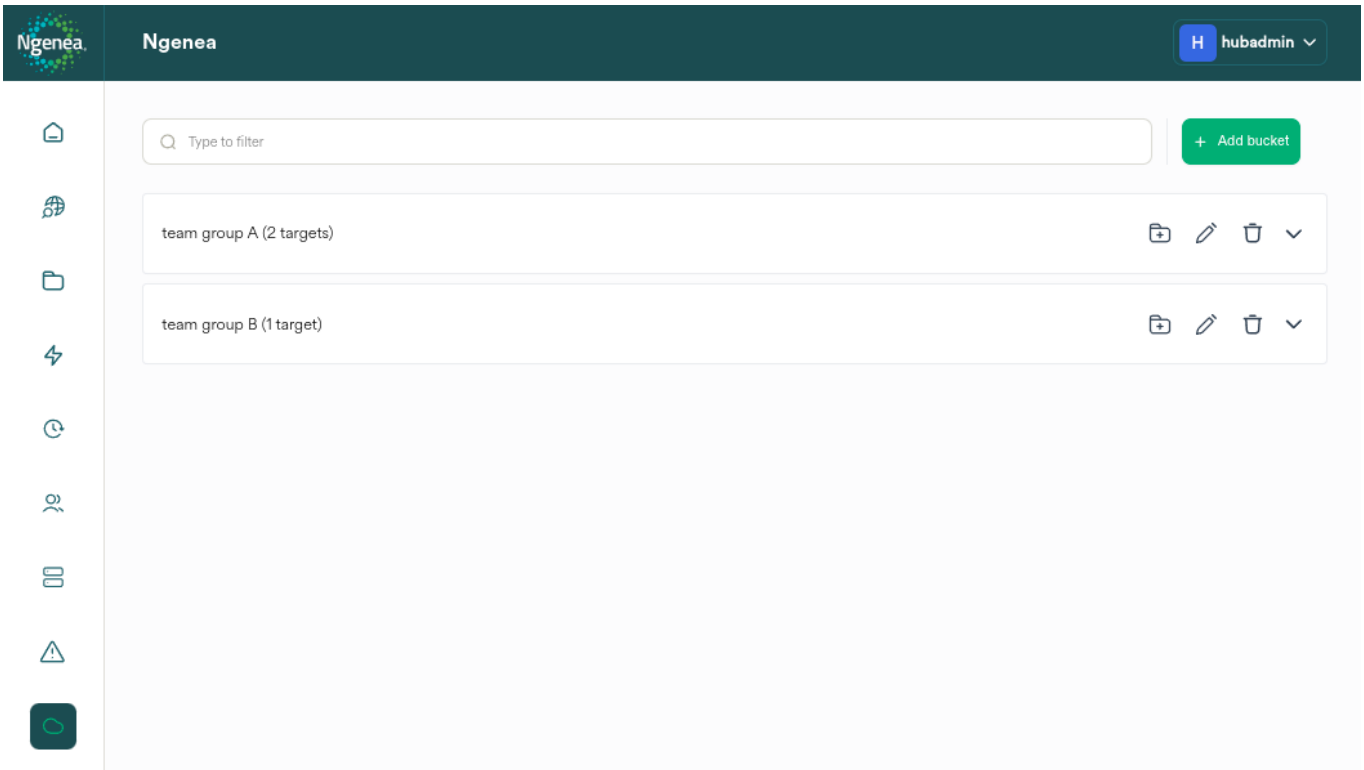
Viewing Buckets

A bucket is area of storage off-PixStor which comprises the following capabilities:

- Type
- location
- Credentials
- Bucket specific settings

Targets	Click the Ngenea button to navigate to the Ngenea screen

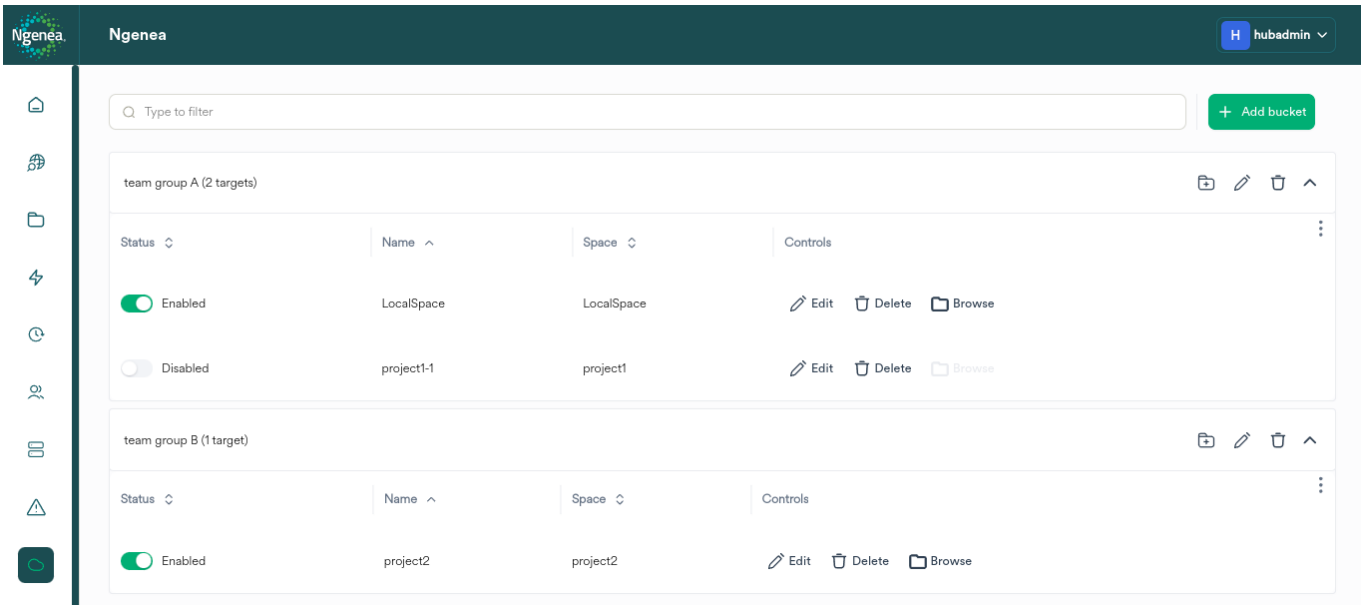
Clicking the Ngenea button in the main menu bar displays the list of Ngenea Buckets:



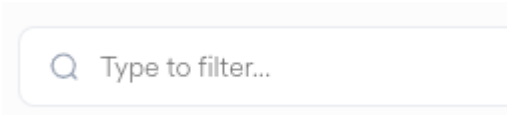
Viewing Targets

A Target is the mapping for how data is transferred to and from a Bucket for a Space.

Click the drop down button of a Bucket to view the associated Targets



Filtering Buckets

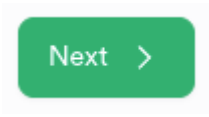


To filter the list of Ngenea Buckets, type the Bucket name or part of a Bucket name in the filter bar.

Bucket Wizard

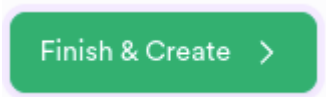
Navigating the Wizard

Click the close button to exit the wizard. Changes are not saved.



Click the Next button to advance to the next page of the wizard. The Next button is disabled until all required page elements are completed.

Click the Go Back button to return to the previous wizard page.



Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Adding a Bucket

Alternatively, a bucket for a Space can be created during the Space Creation Wizard. See Managing Spaces

Important: This function can only be performed by a Hub Administrator.

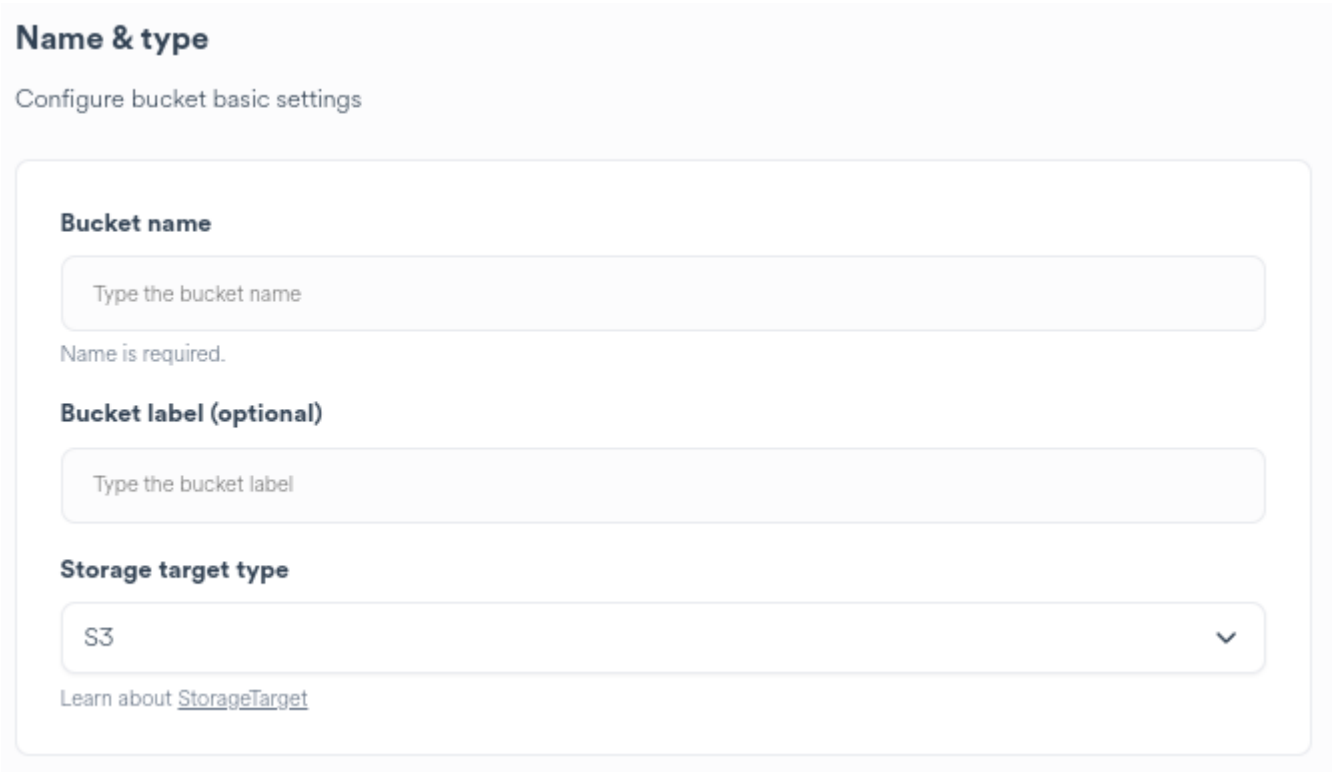
Click the Add Ngenea Bucket button to start the Ngenea Bucket Wizard

Type & reference

Ngenea supports the following storage target types:

Bucket Type	Description
S3	AWS S3 and S3 compatible buckets
Microsoft Azure	Azure Blob Storage
SpectraLogic BlackPearl	Spectralogic DS3 buckets
Google Object Storage	Google Cloud Storage
POSIX file system mount	Locally mounted targets, such as NFS

Select the required Bucket type from the drop down menu.



Enter the name of the Ngenea Bucket name.

E.G.:

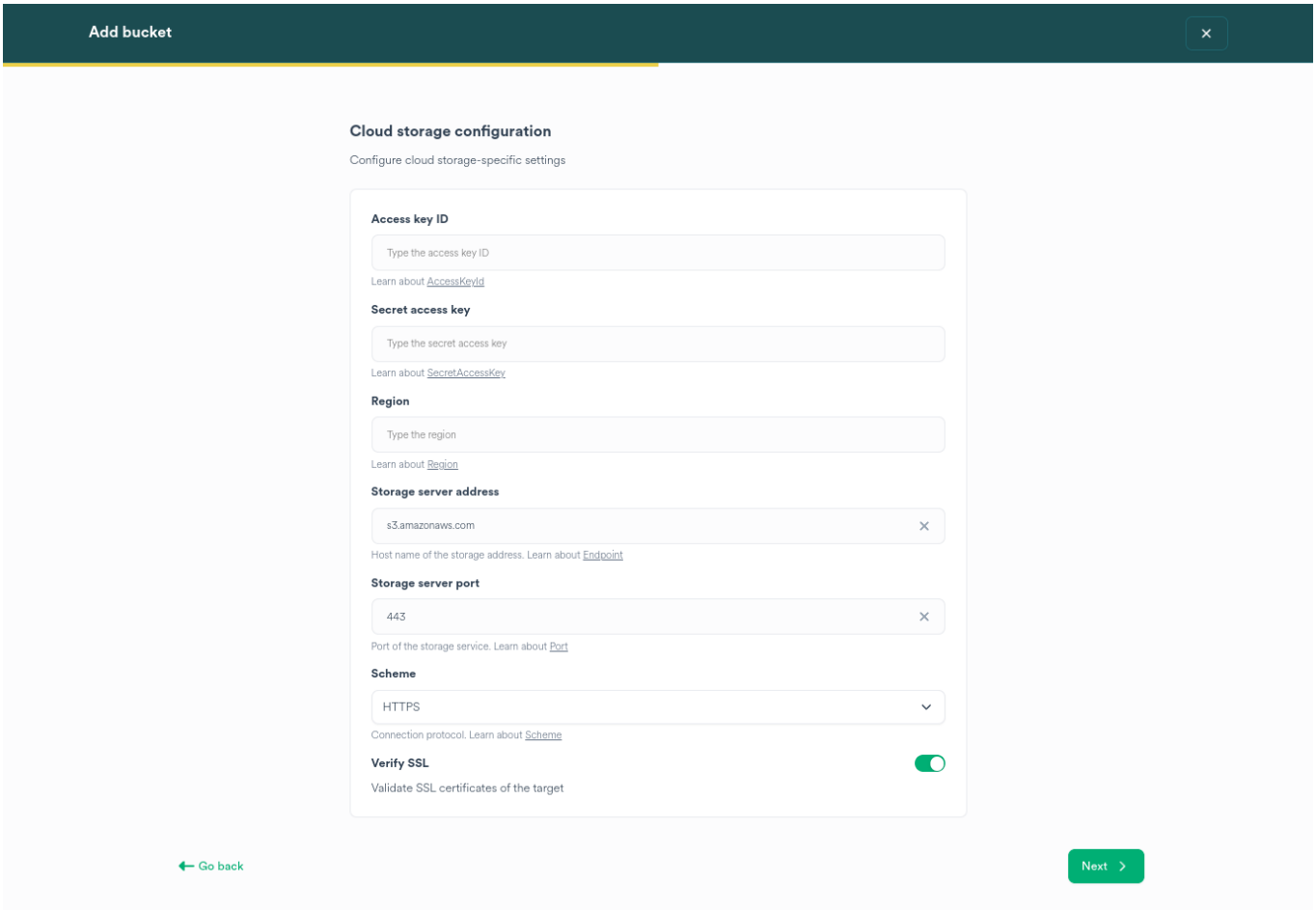
- mybucketname

Provide an optional friendly name as a Bucket label. If specified, users will see the Bucket label when referring to the Bucket in the Hub UI.

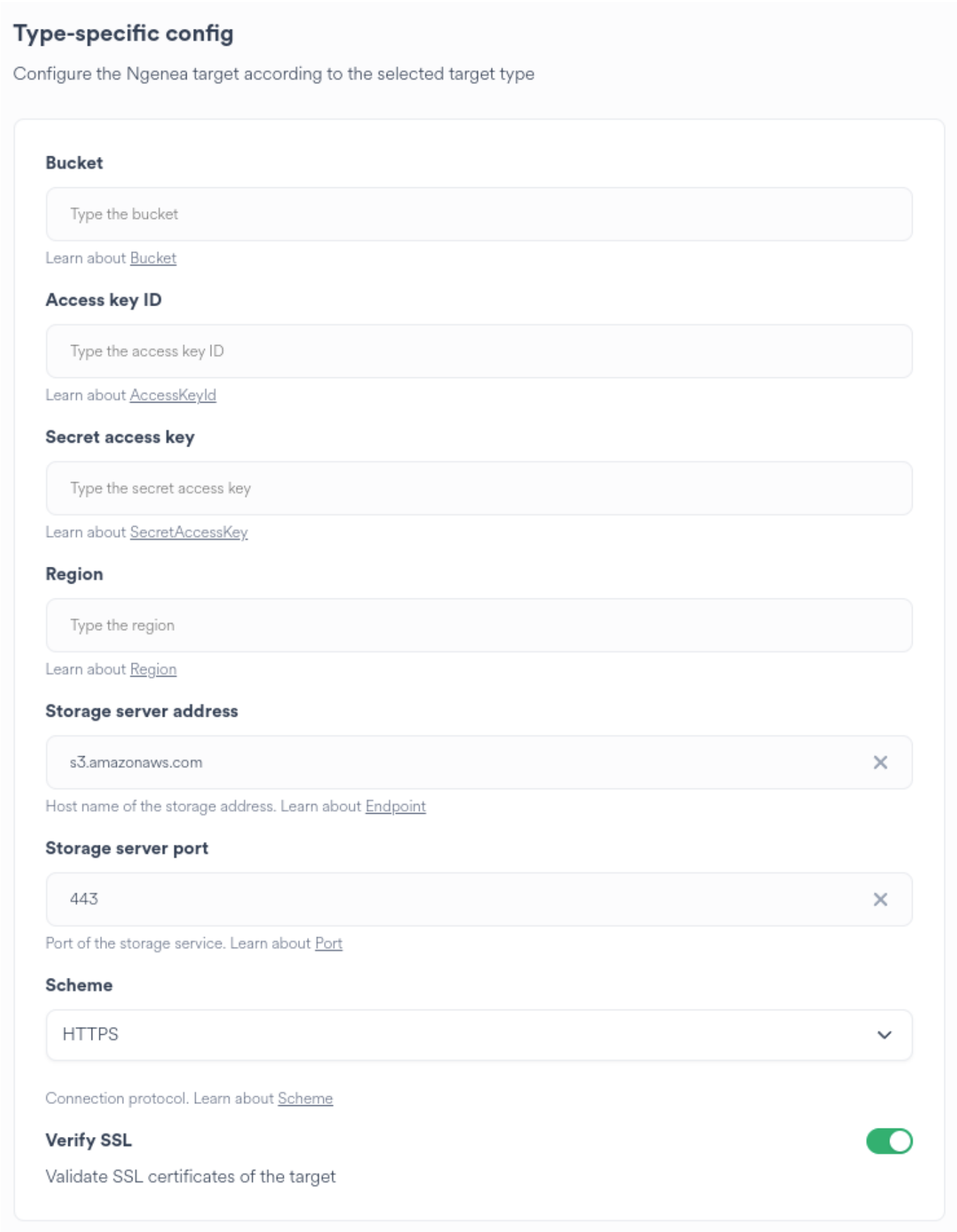
E.G.:

‘mybucketname data’

Depending on the Bucket type selected, a type-specific configuration page will be shown after selecting *Next*.


Add bucket
Cloud storage configuration
Configure cloud storage-specific settings
Access key ID
Type the access key ID
Learn about AccessKeyId
Secret access key
Type the secret access key
Learn about SecretAccessKey
Region
Type the region
Learn about Region
Storage server address
s3.amazonaws.com
Host name of the storage address. Learn about Endpoint
Storage server port
443
Port of the storage service. Learn about Port
Scheme
HTTPS
Connection protocol. Learn about Scheme
Verify SSL
Validate SSL certificates of the target
Go back
Next

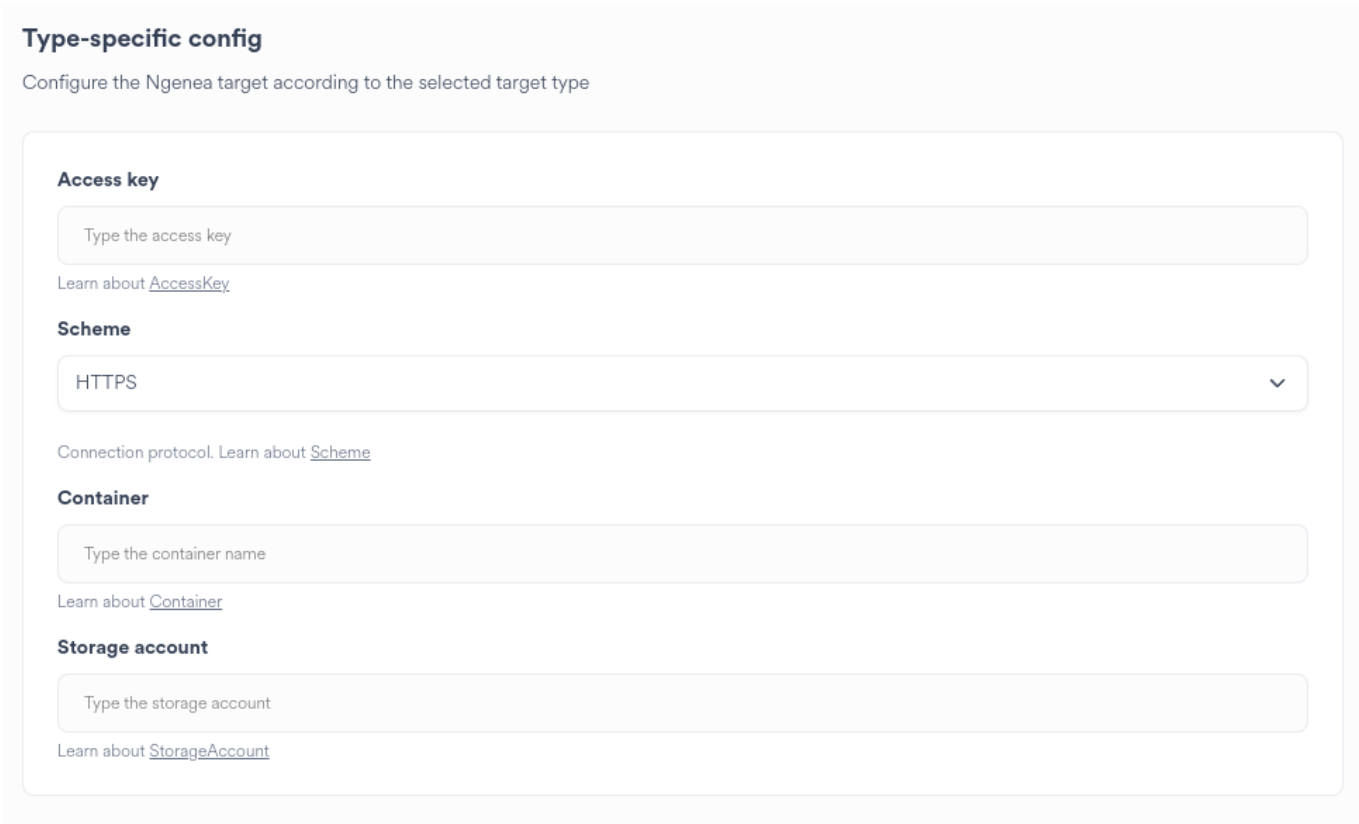
Type-specific config AWS S3 or compatible



Enter the settings as required, which must match those set in the S3 object storage provider:

Setting	Description
Bucket	The name of the storage bucket as specified at the object storage service.
Access key ID	The unique access key for the AWS user account performing data transfers. This will be stored encrypted.
Secret access key	The unique security key for the AWS user account performing data transfers. This will be stored encrypted.
Region	The AWS (or S3 compliant provider) region hosting the S3 Cloud Storage
Storage server address	Not used for Amazon S3 Cloud Storage. For services which reside at specific IPs, such as AWS Snowball, MinIO or LocalStack, specific the host or IP address to connect to.
Storage server port	The TCP/IP port used to communicate
Scheme	HTTP or HTTPS transfer. HTTPS is recommended. Data integrity cannot be guaranteed over HTTP transfer schemes.
Verify SSL	Whether to verify the SSL connection of the target. Disabling the SSL verification allows connections to storage targets which do not provide valid SSL certificates. Connecting to invalid SSL certificates is insecure.

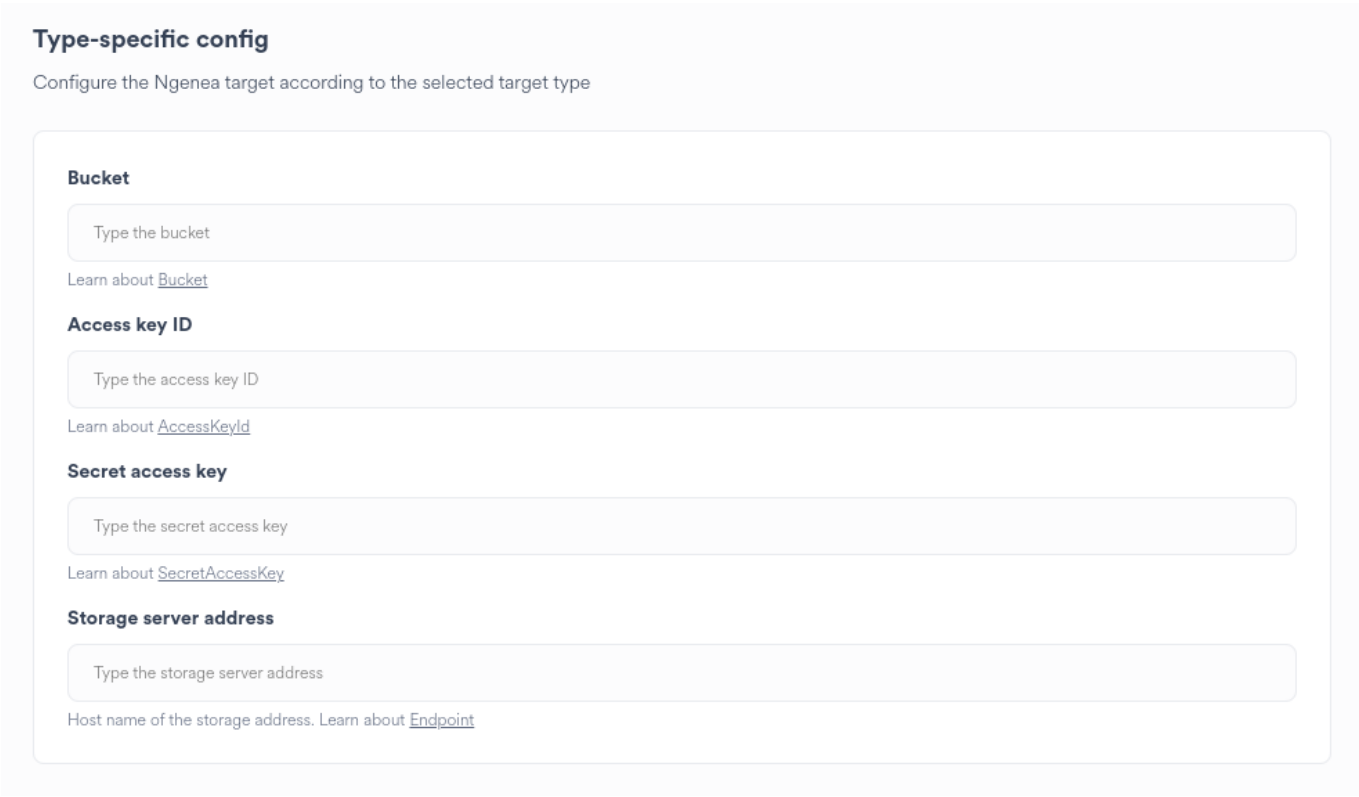
Type-specific config Microsoft Azure



Enter the settings as required, which must match those set in the Azure object storage provider:

Setting	Description
Access key ID	The unique access key for the Azure user account performing data transfers. This will be stored encrypted.
Scheme	HTTP or HTTPS transfer. HTTPS is recommended. Data integrity cannot be guaranteed over HTTP transfer schemes.
Container	The storage container for the blob data.
Storage account.	The Azure namespace containing the Container

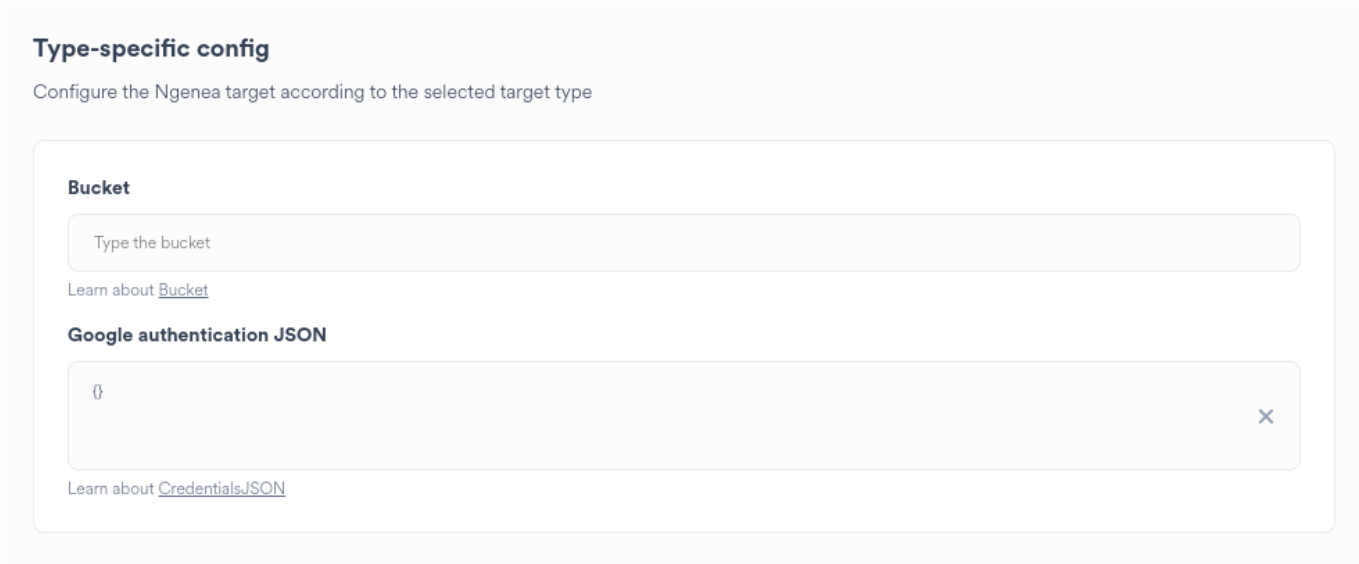
Type-specific config Spectra Logic BlackPearl



Enter the settings as required, which must match those set in the BlackPearl DS3 object storage provider:

Setting	Description
Bucket	The name of the storage bucket as specified at the object storage service.
Access key ID	The unique access key for the BlackPearl user account performing data transfers. This will be stored encrypted.
Secret access key	The unique security key for the BlackPearl user account performing data transfers. This will be stored encrypted.
Storage server address	The FQDN hostname of the BlackPearl.

Type-specific config Google Object Storage

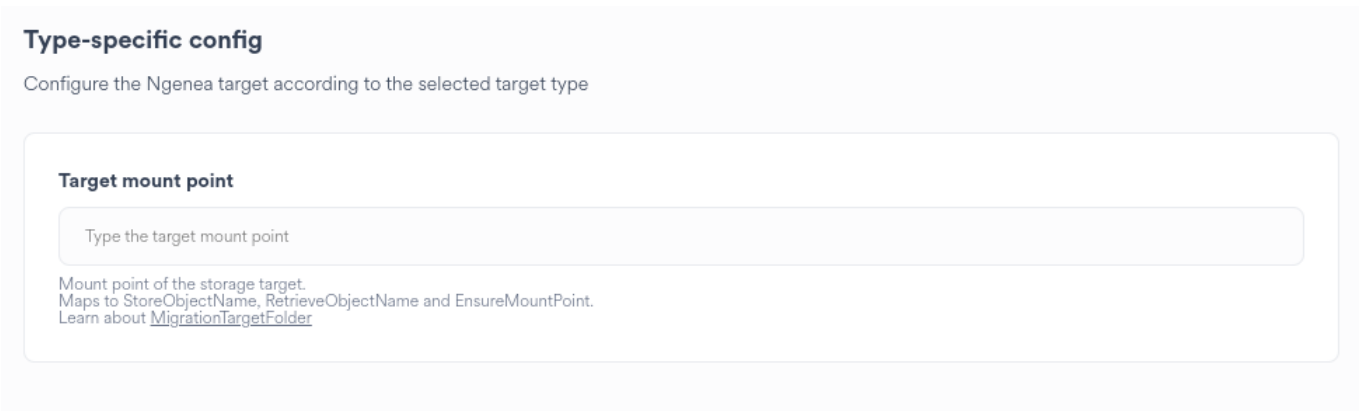


Enter the settings as required, which must match those set in the Google Cloud Storage object storage provider:

Setting	Description
Bucket	The name of the storage bucket as specified at the object storage service.
Google authentication JSON	Enter the contents of the JSON key for the user or service account granted permission to transfer data to Cloud Storage bucket. This will be stored encrypted. For more information refer to Google documentation

Important: If the Ngenea Bucket is to be used as a backup only bucket, *do not* define JSON information in the Google Authentication JSON. Instead define keyword `CredentialsFile` with the path to the file containing JSON credentials on pixstor. For more information refer to Backup-Only Targets.

Type-specific config Filesystem Mount

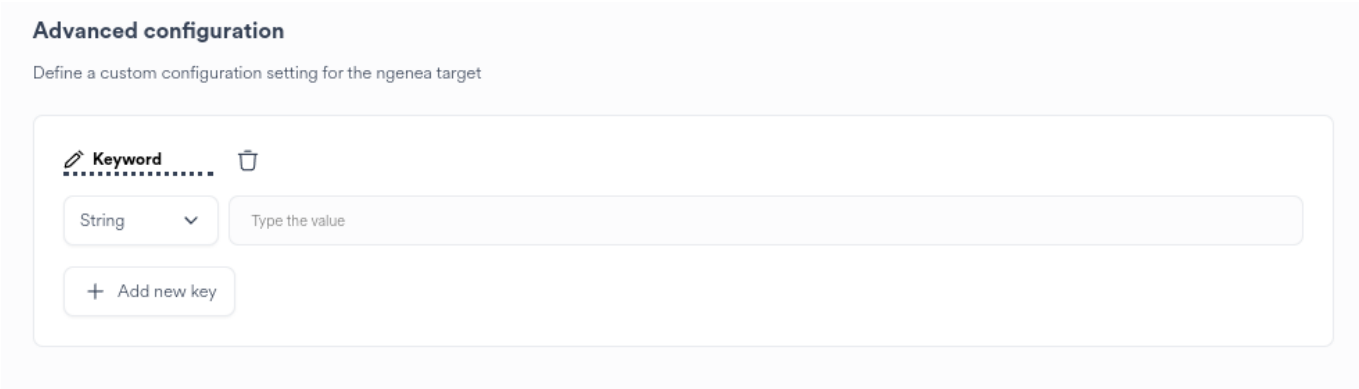


Enter the settings as required, which must match the location of the NAS mount point on the pixstor:

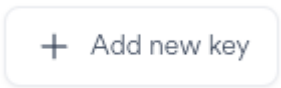
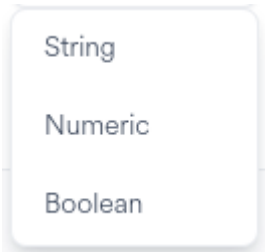
Important: Bucket mounts points are site-specific however Hub will create the target across sites associated with Space.

Setting	Description
Target mount point	The location of the POSIX compliant mounted storage. E.G. If the external POSIX storage is mounted on `/mmfs1/myexternalnas` enter `/mmfs1/myexternalnas`

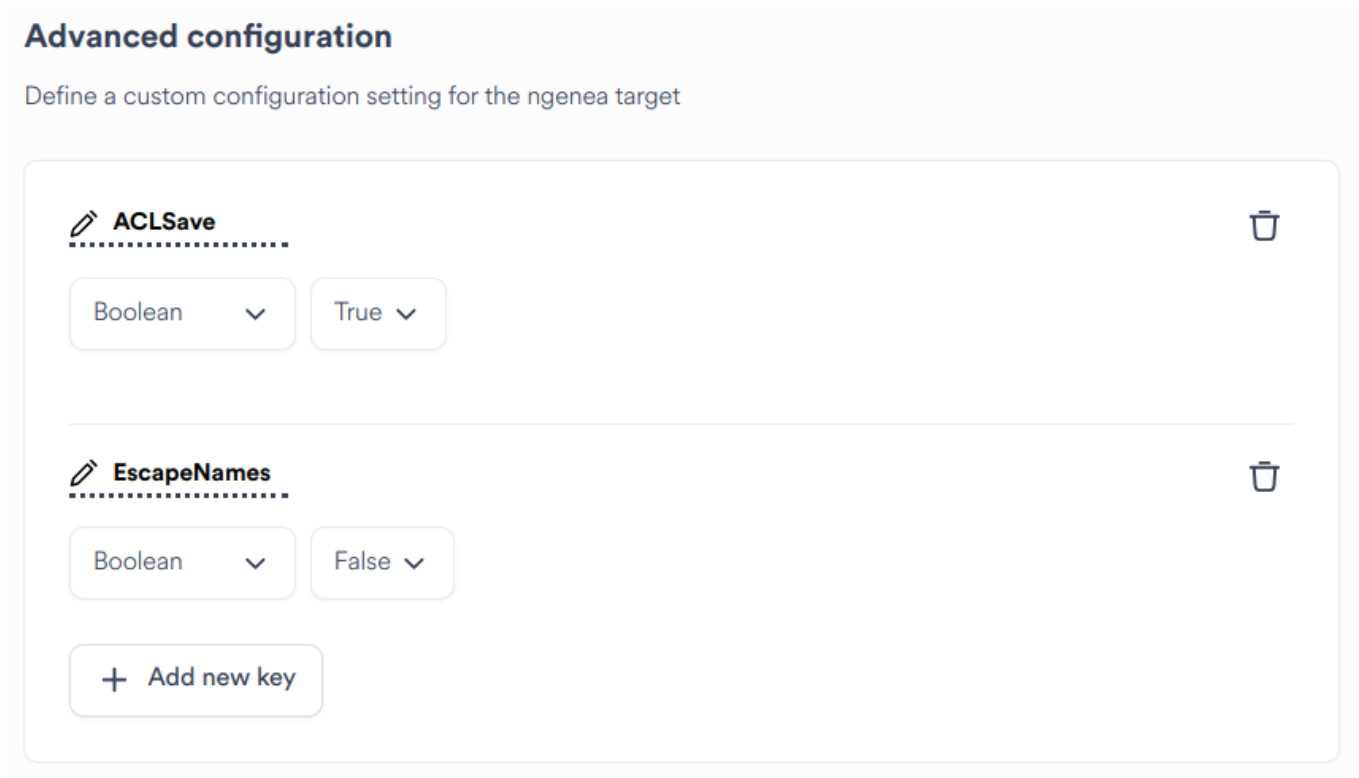
Advanced configuration



To add configuration settings to a target to control specific behaviour during Ngenea data operations.

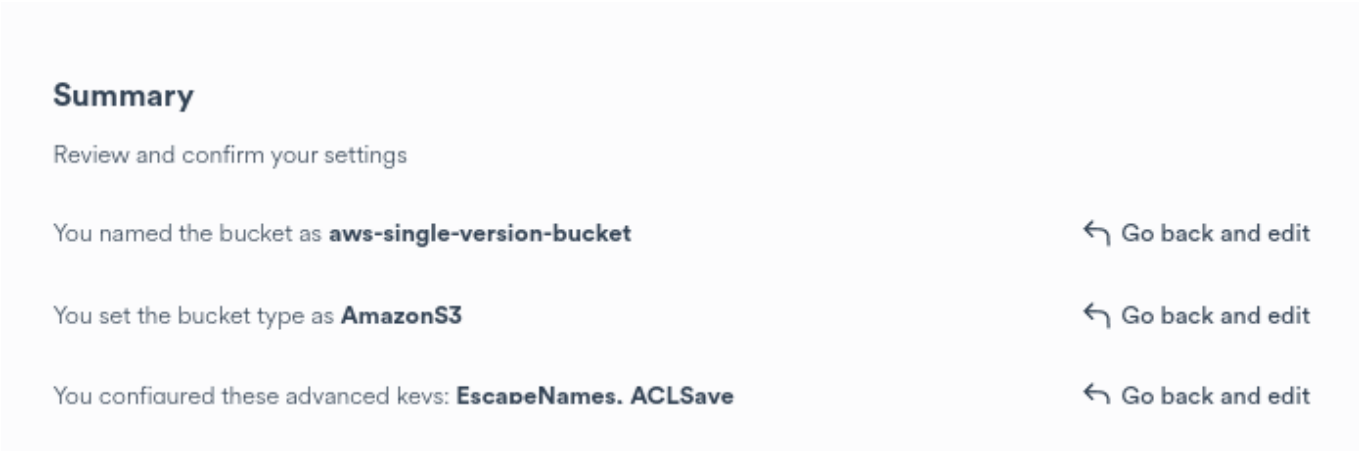
	Click the Add new key to define a new configuration setting
	Enter the name of the configuration setting in the Keyword field
	Select the type of configuration setting from the drop down menu. Choose or enter the value for the configuration setting.

Example of added Advanced configuration settings:



Summary

Upon completing the wizard steps a summary is presented:



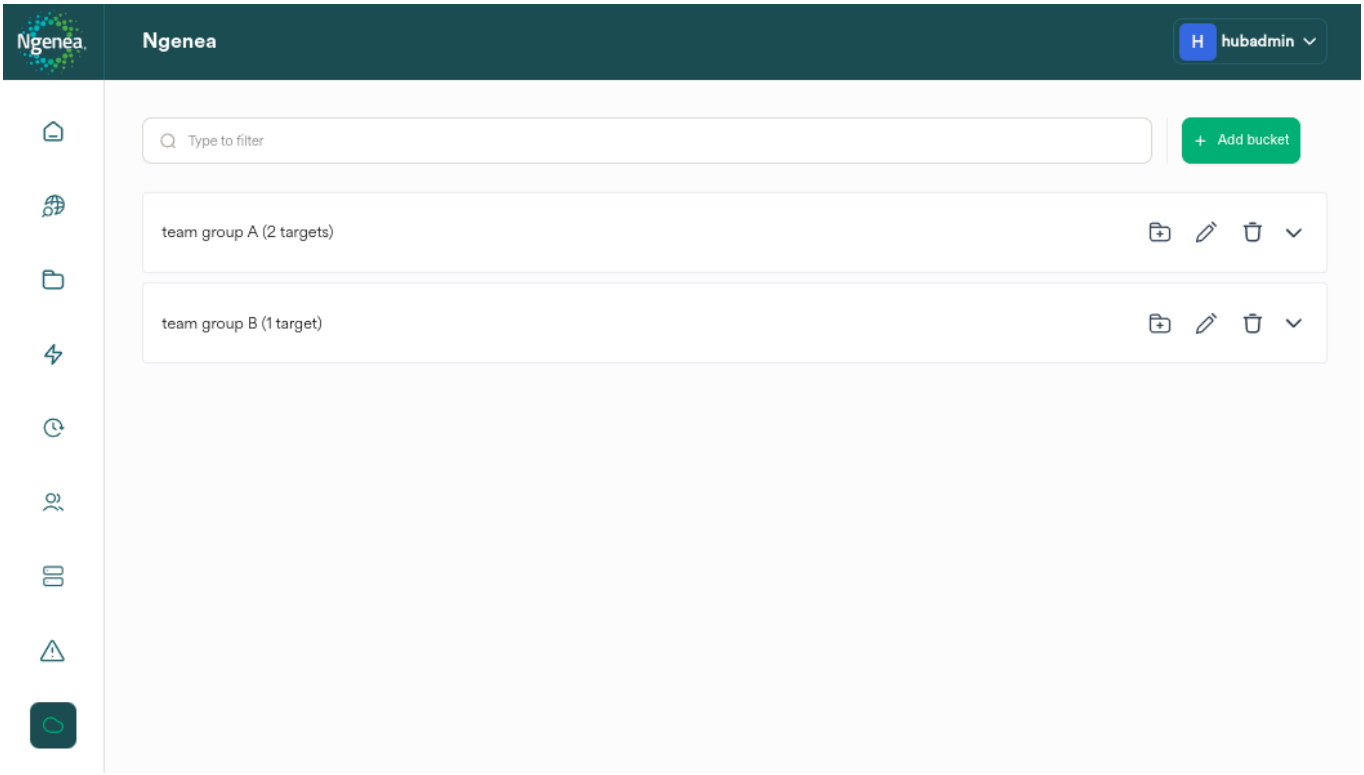
Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Alternatively `Go back` and change the proposed configuration as required or `close` the wizard to cancel the creation of the Ngenea target.

Editing a Bucket

Important: This function can only be performed by a Hub Administrator.

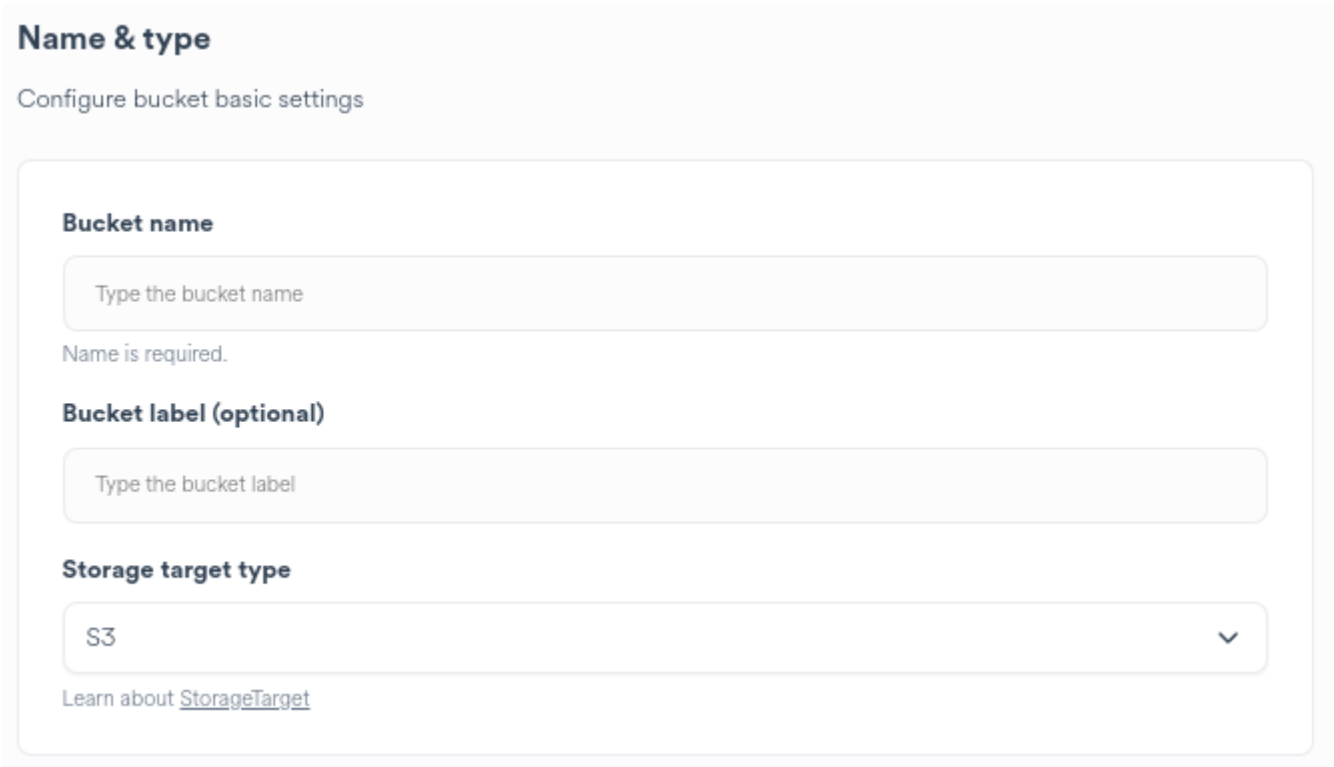
Clicking the Ngenea button in the main menu bar displays the list of Ngenea Buckets:



Edit

Click the edit icon on the required Ngenea Bucket row to edit the Ngenea Bucket

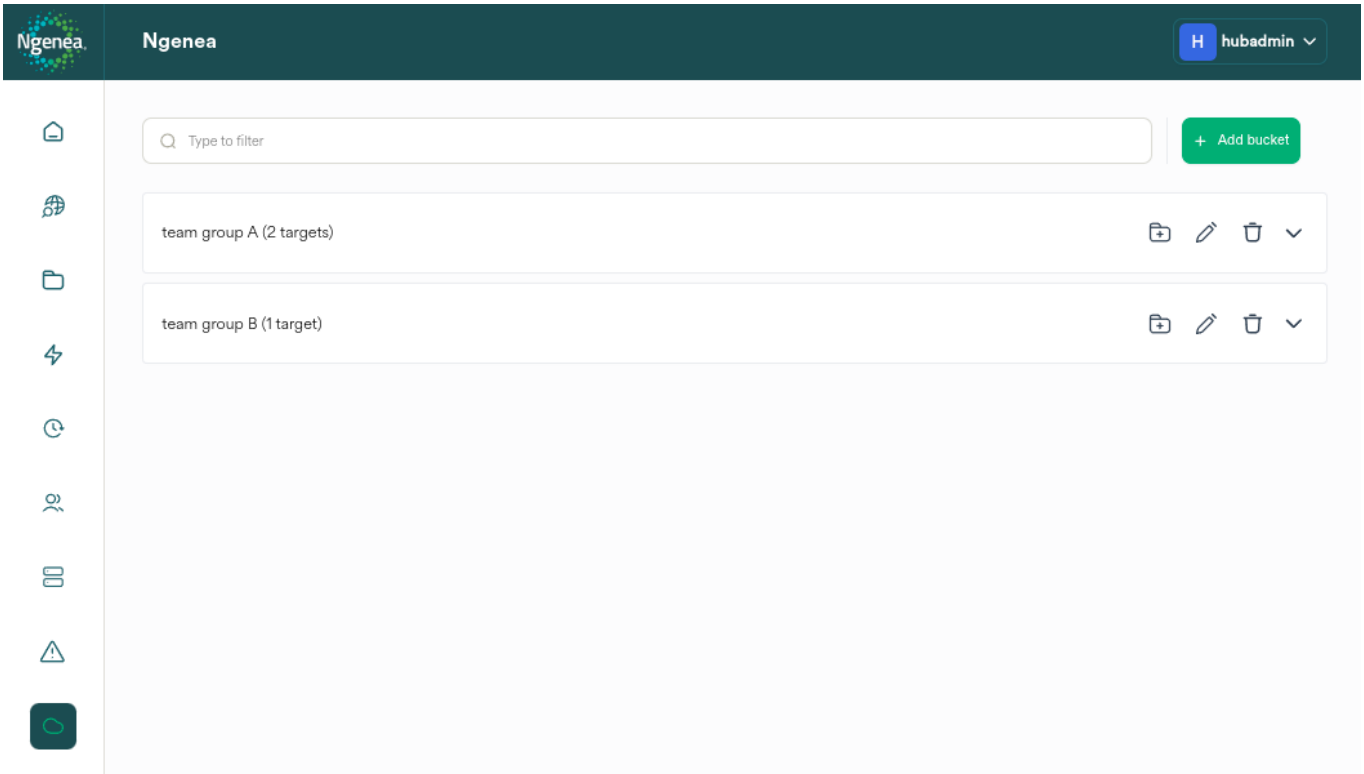
- Modify the Ngenea Bucket settings as required. Refer to Adding a Bucket for settings guidance.

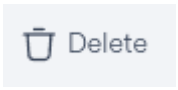


Deleting a Bucket

Important: This function can only be performed by a Hub Administrator.

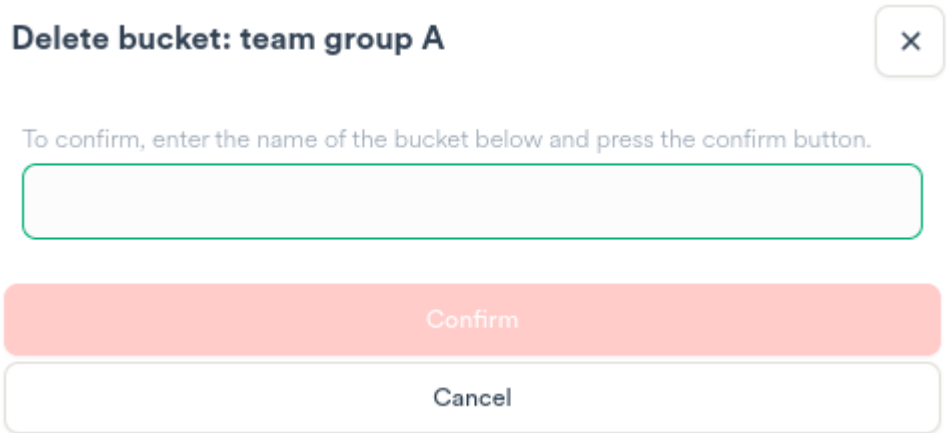
Clicking the Ngenea button in the main menu bar displays the list of Ngenea Buckets:





Click the delete icon on the required Ngenea bucket row to delete the Ngenea Bucket

A confirmation dialog is raised:



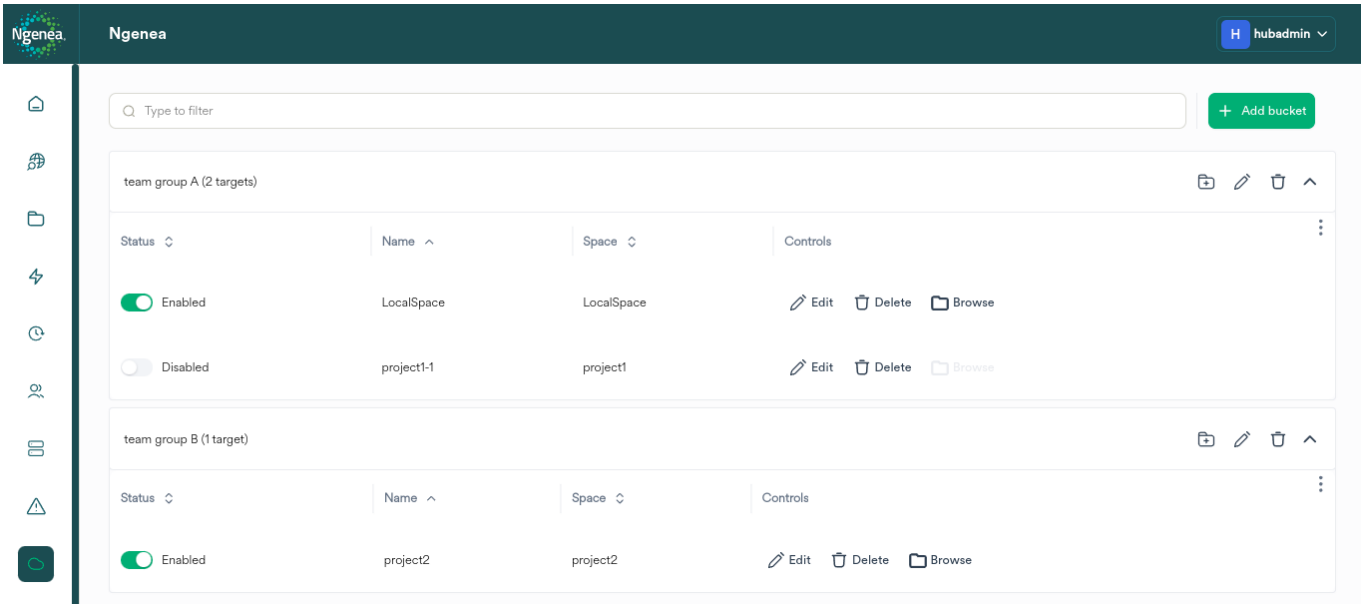
- Type the name of the Ngenea Bucket
- Click Yes to delete the Ngenea Bucket. This action is irreversable.
- Alternately click no, or close the confirmation dialog.

Important: Deleting a space automatically deletes associated targets. The parent bucket of the targets is not deleted, the bucket configuration still exists in pixstor and requires manual removal at the pixstor CLI.

Browsing a Target

Important: This function can only be performed by a Hub Administrator.

Clicking the Ngenea button in the main menu bar displays the list of Ngenea targets:



	Click the drop down button of a Bucket to view the associated Targets
Browse	Click the browse icon on the Ngenea target row to browse the Ngenea target

For more information on browsing and importing from an Ngenea target, see Target Import

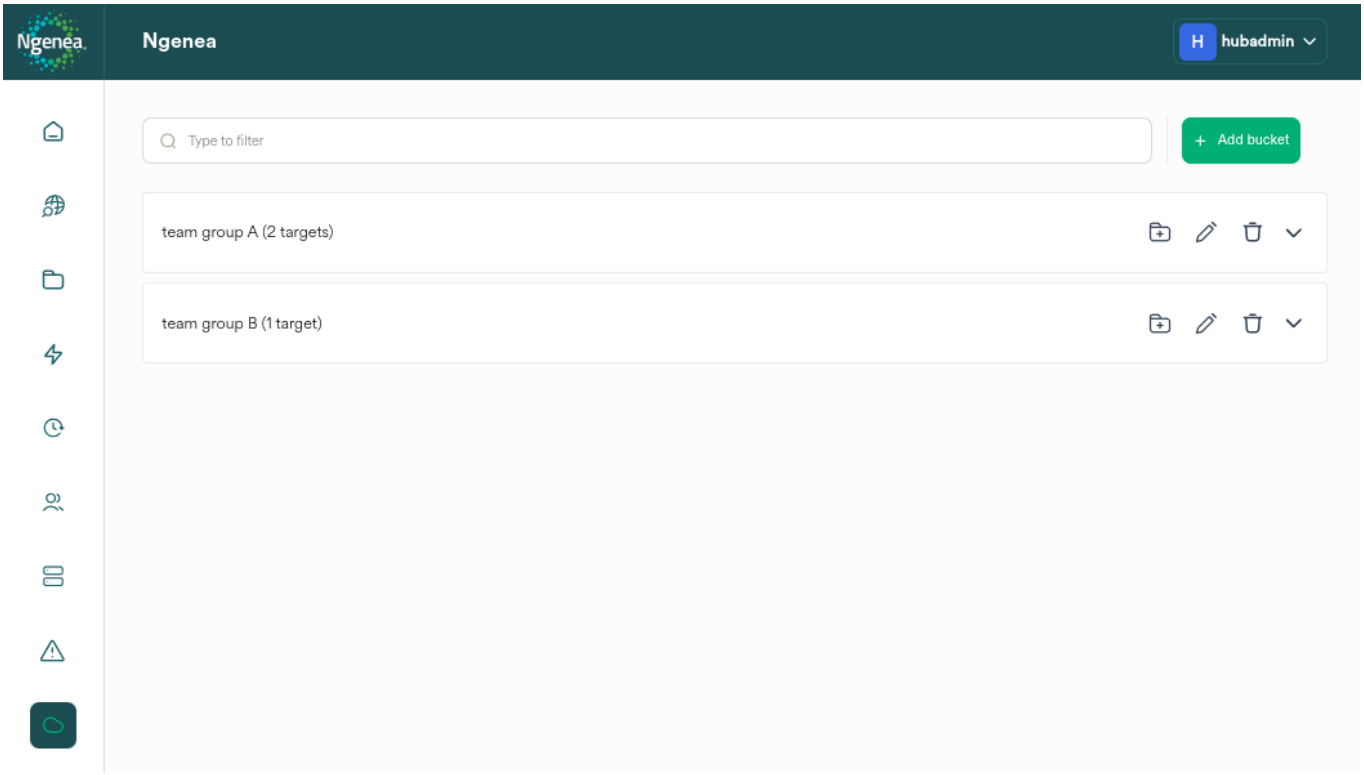
Adding a Target

A Target links a Space to a Bucket. A Target that is linked in this way must:

- have a Regex filter that matches the Space mountpoint which can also include subdirectories with the Space mountpoint.
- be linked to the same sites as the Space.

Targets	Click the Ngenea button to navigate to the Ngenea screen

Clicking the Ngenea button in the main menu bar displays the list of Ngenea Buckets:



	Click the Add Target button to start the External Target Wizard

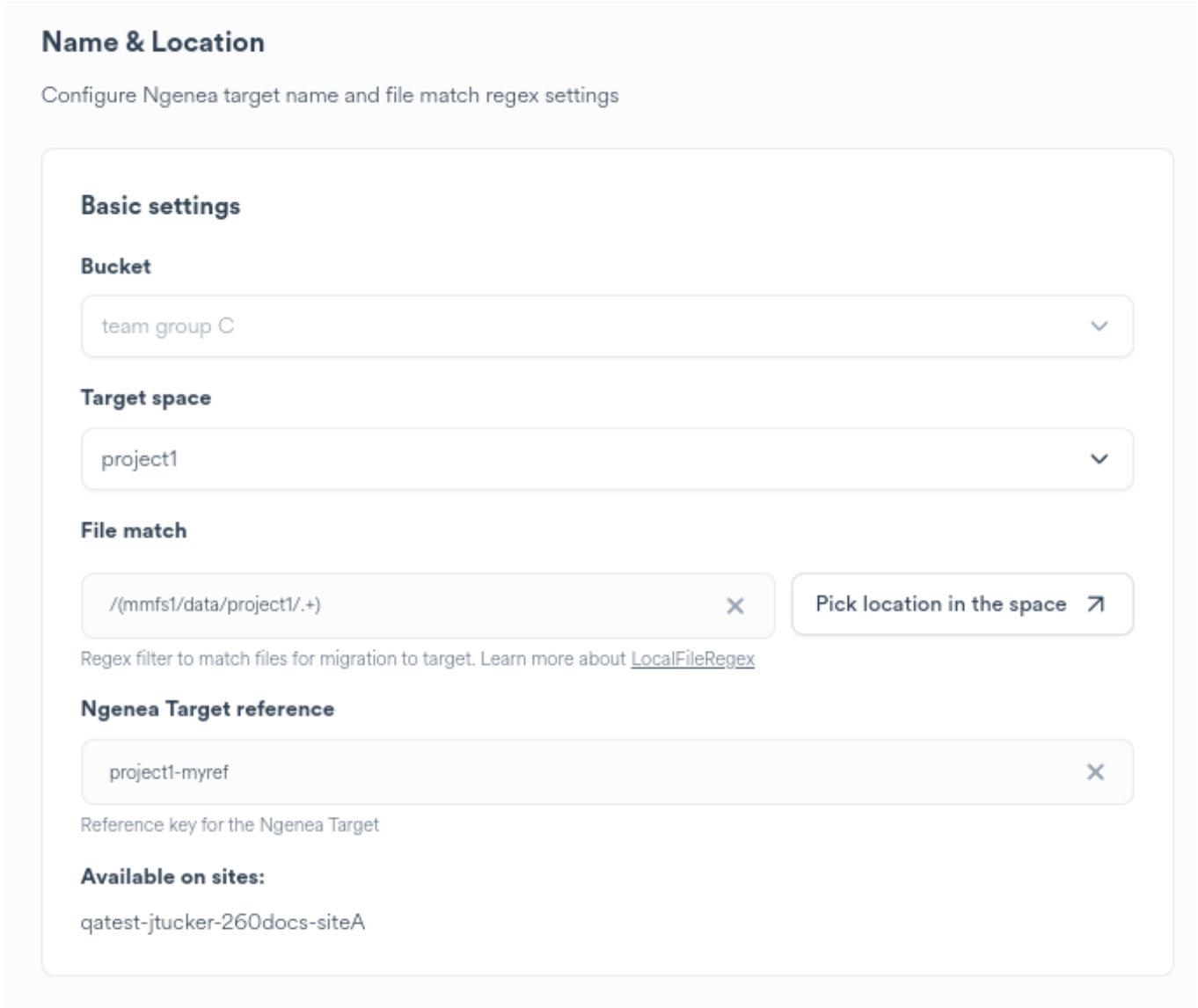
External Target Wizard

Navigating the Wizard

	Click the close button to exit the wizard. Changes are not saved.
Next	Click the Next button to advance to the next page of the wizard. The Next button is disabled until all required page elements are completed.
Go back	Click the Go Back button to return to the previous wizard page.
Finish & Create	Click the Finish & Create button to apply the changes displayed on the wizard summary page.

Name & Location

Upon starting the External target wizard, the user is presented with the Name & Location page.



Configure the migration settings as required to handle data migration accordingly:

Setting	Description
Bucket	The name of the Bucket the Target will be associated with. The bucket name cannot be changed.
Target Space	The name of the Space the Target will be associated with. Selecting the Space generates a suggested File match entry. If no space is selected, you must manually select which sites it is available on.
File match	Refer to the RegEx Filters example table below
Ngenea Target reference	The metadata key to add to all files processed by Ngenea HSM.
Available on sites	The sites the Space is associated with and therefore the Target will be created are listed. The sites cannot be changed.

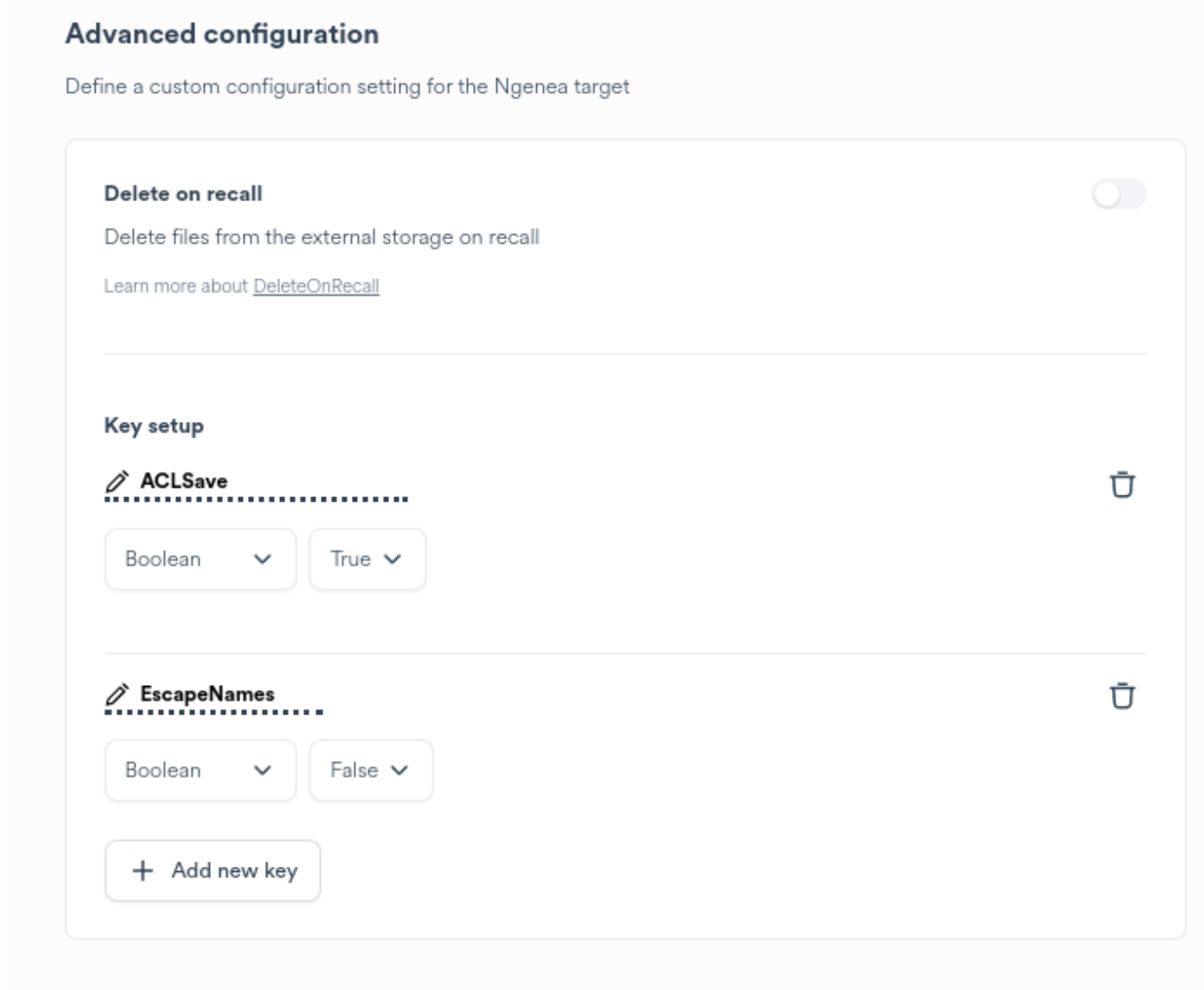
Note: When pre-creating a target not associated to a Space where the Target Space is set to 'No space is selected', Hub will not suggest automatic settings for the dialog. In this case it required to manually set and review the dialog setting so that the wizard can proceed further.

Example regex filters where a space named myspace is present on the pixstor filesystem at location `/mmfs1/data/myspace`:

RegEx Filter Examples	Outcome
`/mmfs1/data/myspace/(.*)`	Dehydrated files within the myspace folder are present at the root of the storage target.
`/mmfs1/data/(myspace/.*)`	The myspace folder is present at the root of the storage target.
`/mmfs1/data/myspace/subdirectory/(.*)`	Dehydrated files within the subdirectory of the myspace directory are present at the root of the storage target. The myspace directory is not present. Data immediately within the myspace directory (other than that within the subdirectory) is not eligible for Ngenea operations.
`/mmfs1/data/myspace/(subdirectory/.*)`	The subdirectory of the myspace directory is present at the root of the storage target. The myspace directory is not present. Data immediately within the myspace directory (other than that within the subdirectory) is not eligible for Ngenea operations.
`/mmfs1/data/(myspace/.+)`	This regex matches paths under /mmfs1/data/ that contain myspace/ followed by one or more characters. The myspace folder is present at the root of the storage target.
`/mmfs1/(data/myspace/.+)`	This regex matches paths containing data/myspace/ after /mmfs1/. The data folder is present at the root of the storage target.
`/(mmfs1/data/myspace/.+)`	This regex matches paths under mmfs1/data/myspace/.

RegEx Filter Examples	Outcome
`/mmfs1/data/((myspace1|myspace2|myspace3)/.+)`	This regex matches paths under /mmfs1/data/ where the space is one of myspace1, myspace2, or myspace3.
`/mmfs1/(data/(myspace1|myspace2|myspace3)/.+)`	This regex matches paths under /mmfs1/ where the path contains data/ followed by one of myspace1, myspace2, or myspace3, and then additional path segments.
`/(mmfs1/data/(myspace1|myspace2|myspace3)/.+)`	This regex is similar to the previous one but matches paths that start directly with mmfs1/data/ and then one of the three space names, followed by additional segments.

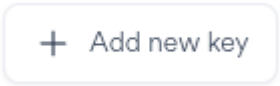
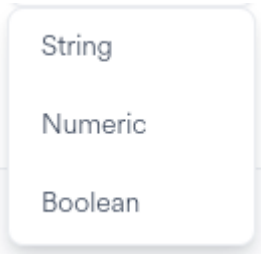
Advanced configuration



Configure the migration settings as required to handle data migration accordingly:

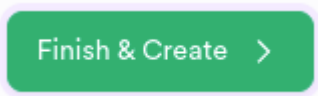
Setting	Description
Delete on recall	Determines whether to delete the recalled data from the Bucket after the data has been successfully recalled.
Key setup	Add configuration settings to a target to control specific behaviour during Ngenea data operations.

Important: Advanced Keys for a Target override those defined for the associated Bucket.

	Click the Add new key to define a new configuration setting
	Enter the name of the configuration setting in the Keyword field
	Select the type of configuration setting from the drop down menu. Choose or enter the value for the configuration setting.

Summary

Upon completing the wizard steps a summary is presented:

	Click the Finish & Create button to apply the changes displayed on the wizard summary page.

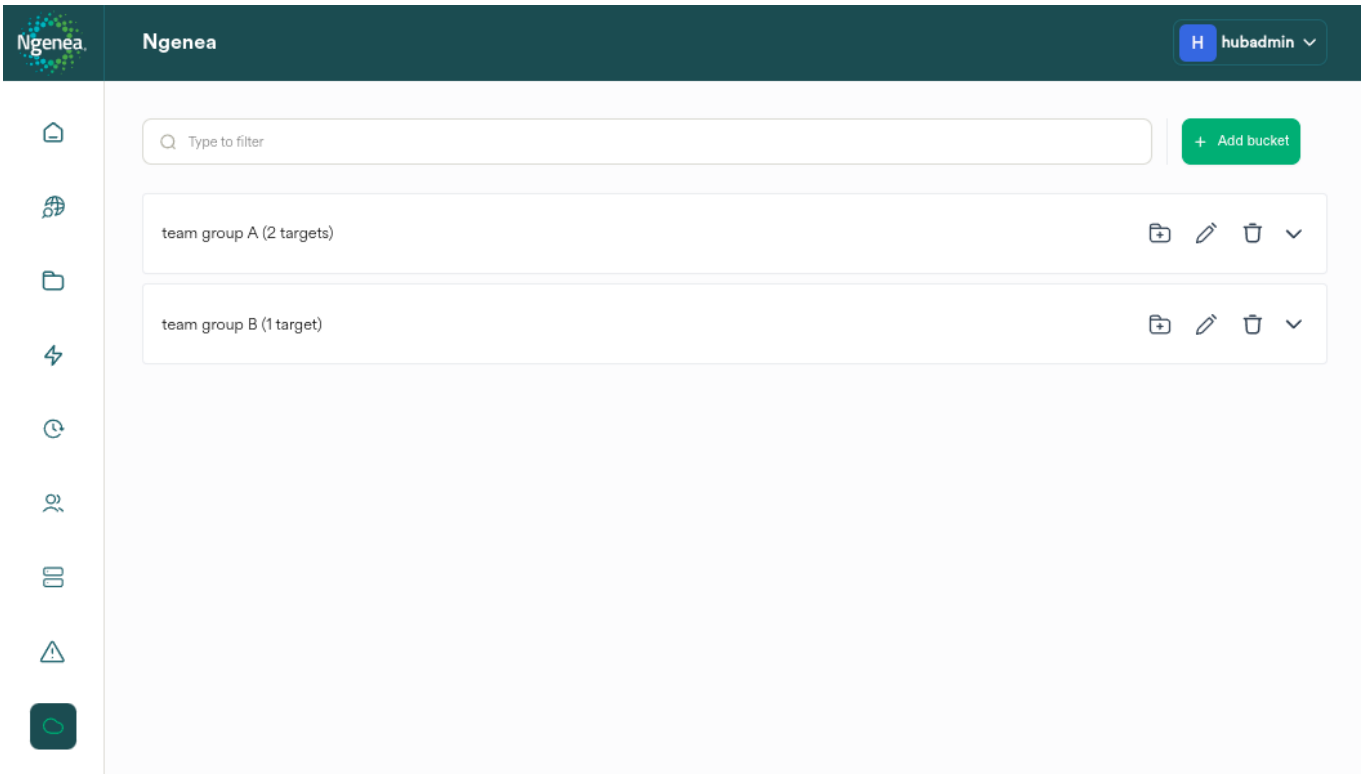
Alternatively `Go back` and change the proposed configuration as required or `close` the wizard to cancel the creation of the Ngenea target.

Target Import

An Ngenea target can be browsed, and files can be imported to the local PixStor file system.

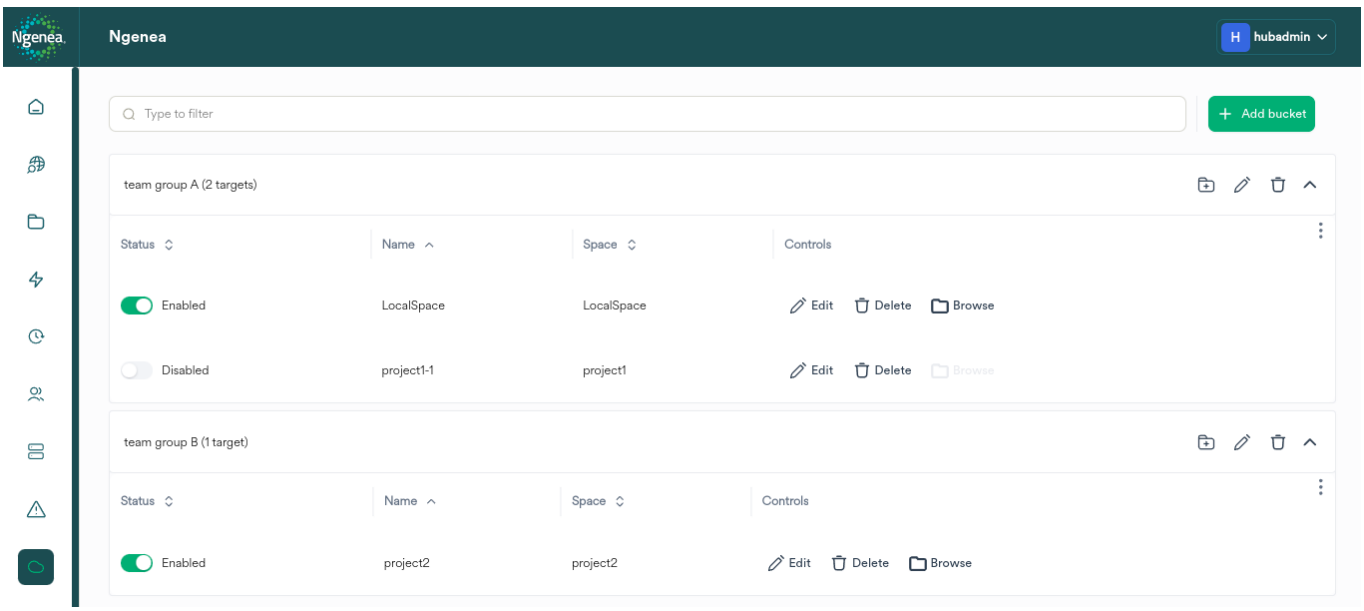
Important: This function can only be performed by a Hub Administrator.

Clicking the Targets button in the main menu bar displays the list of Ngenea buckets:



A Target is the mapping for how data is transferred to and from a Bucket for a Space.

Click the drop down button of a Bucket to view the associated Targets

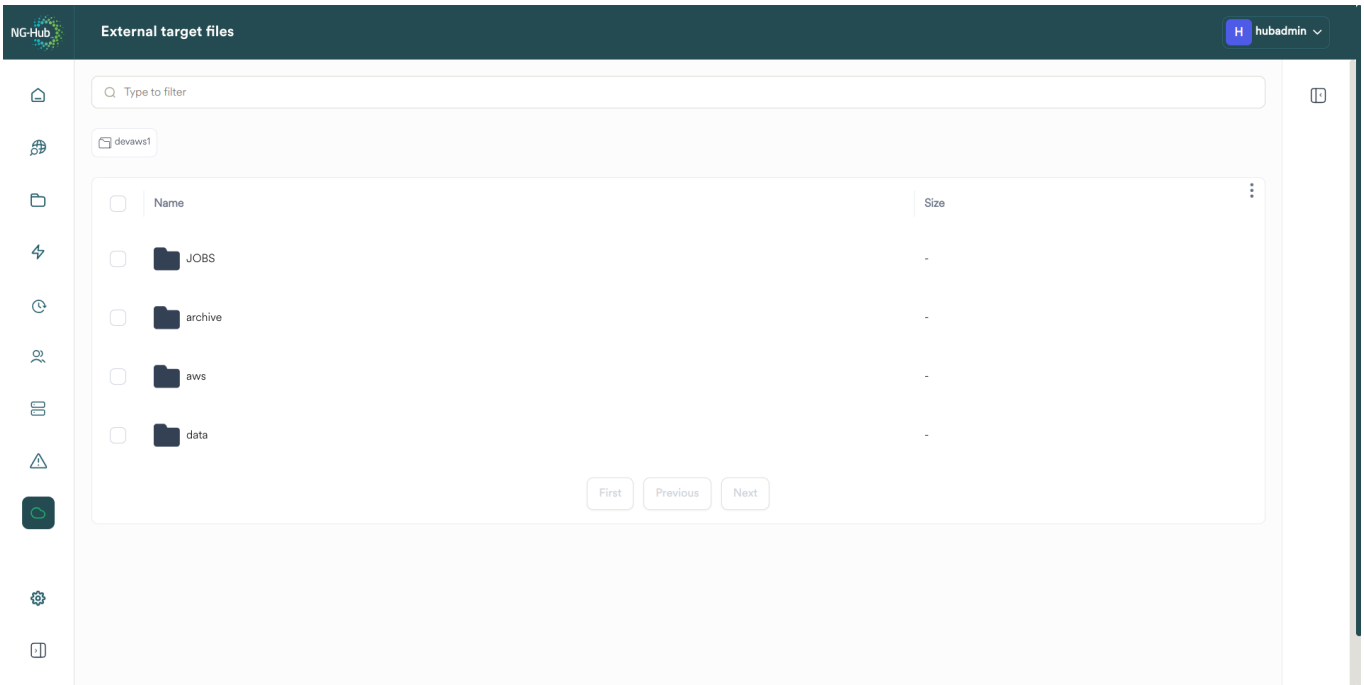


Browse	Click the browse icon on the Ngenea target row to browse the Ngenea target

Browsing a Target

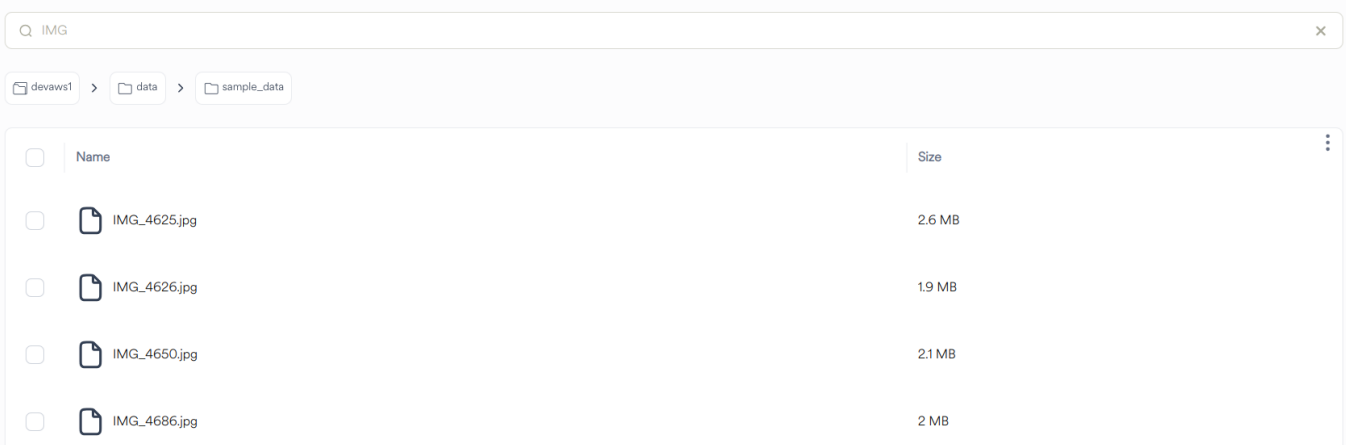
The target browser displays the file and folder contents of a target including additional contextual information such as size.

The target bowser screen is comprised of several areas.



Filtering the Target Browser

To display files or folders matching keywords, enter the keywords in the filter bar.



Target Browser File Attributes

The target browser displays information regarding the files and folders within the Target.

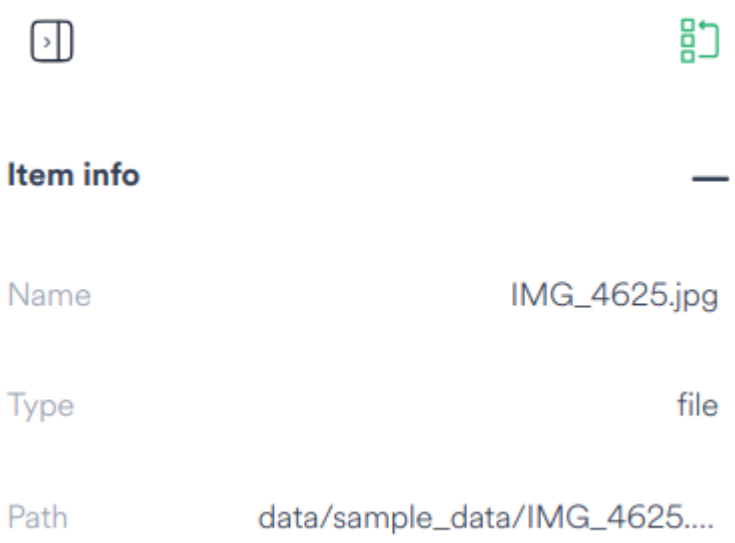
Available information may vary between target types.

Option	Description
Name	The file or directory name
Size	The size of the file on the target

Viewing File or Folder Metadata

Metadata Panel

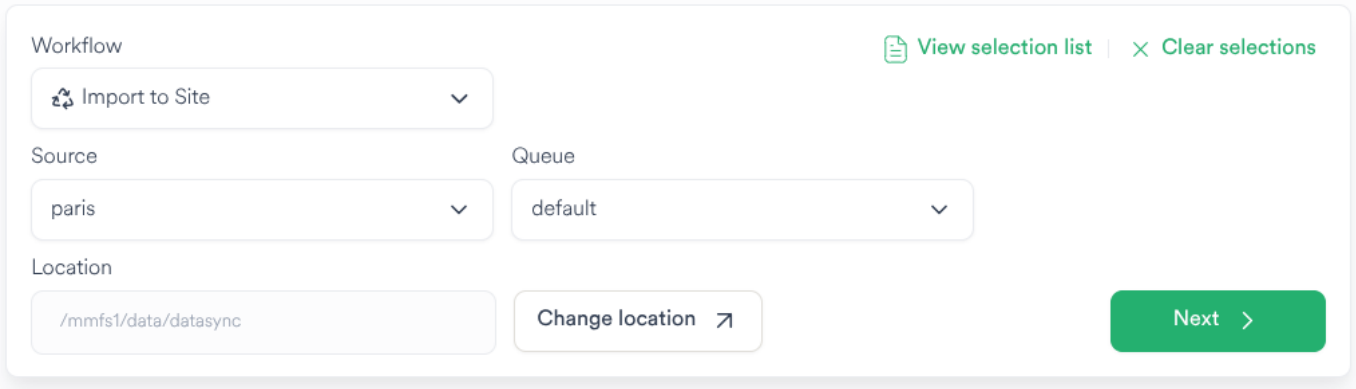
Click a file or directory name to view the associated metadata:



The metadata panel provides a summary of metadata.

Using the Job Creator Panel

Explain panel areas, buttons and operations

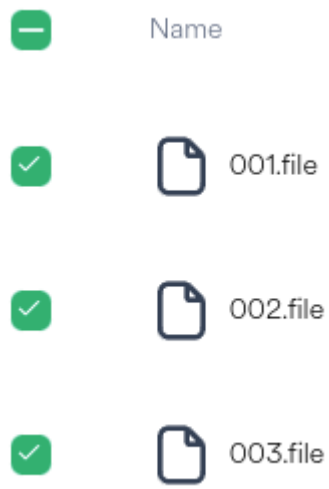


Selecting files and directories

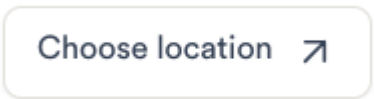
Selecting an individual file or directory populates the selection list with the item. Selecting a directory populates all items within the chosen directory tree.

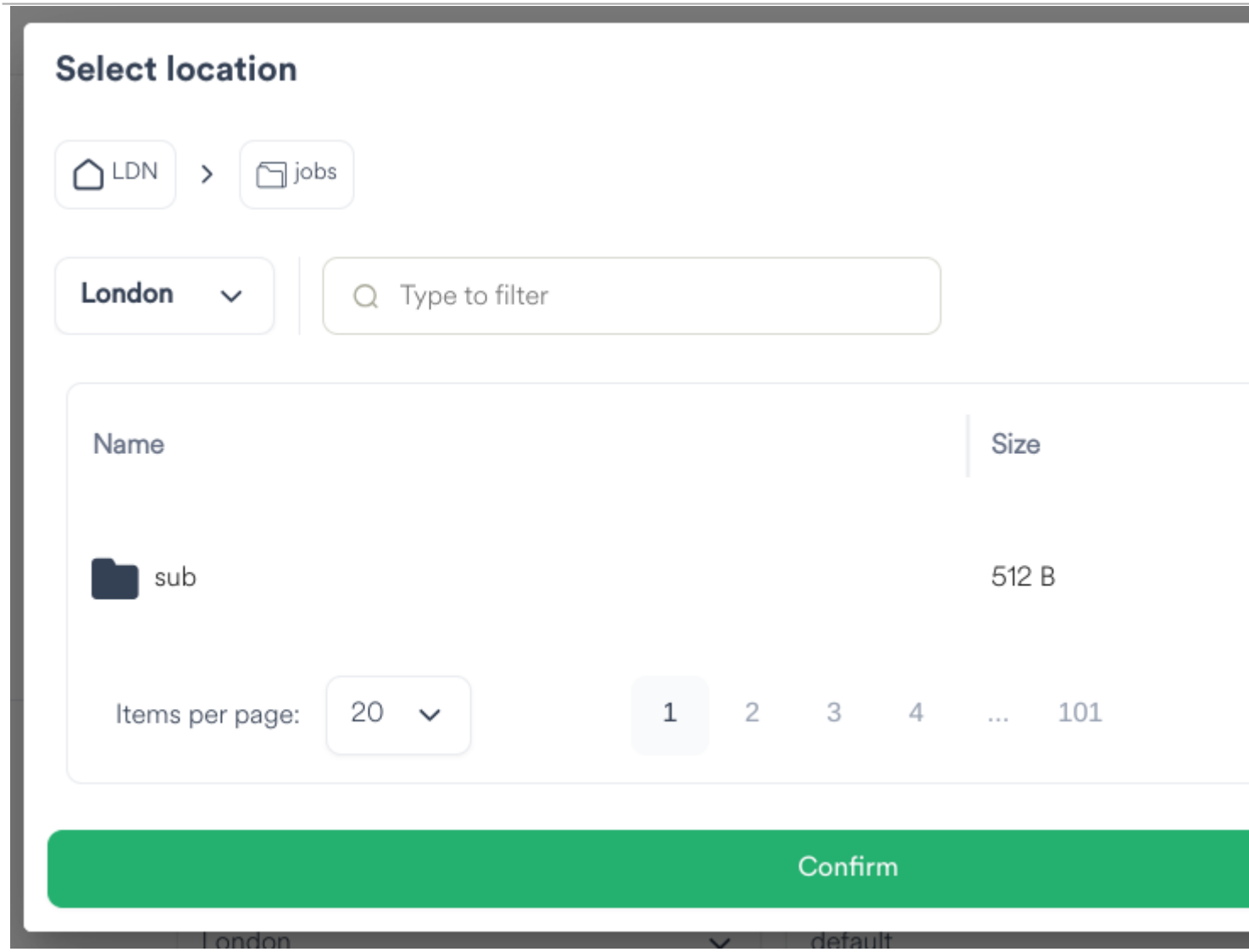
Selections of the entire Target or a directory are each counted as 1 item.

Select the ‘select all’ checkbox next to the Name field in order to select the entire Target.



View selection list	Click the View Selection List button to show the files and directories selected to be processed by the chosen workflow
× Clear selections	Click the Clear selections to remove all files and directories currently selected from the selection list





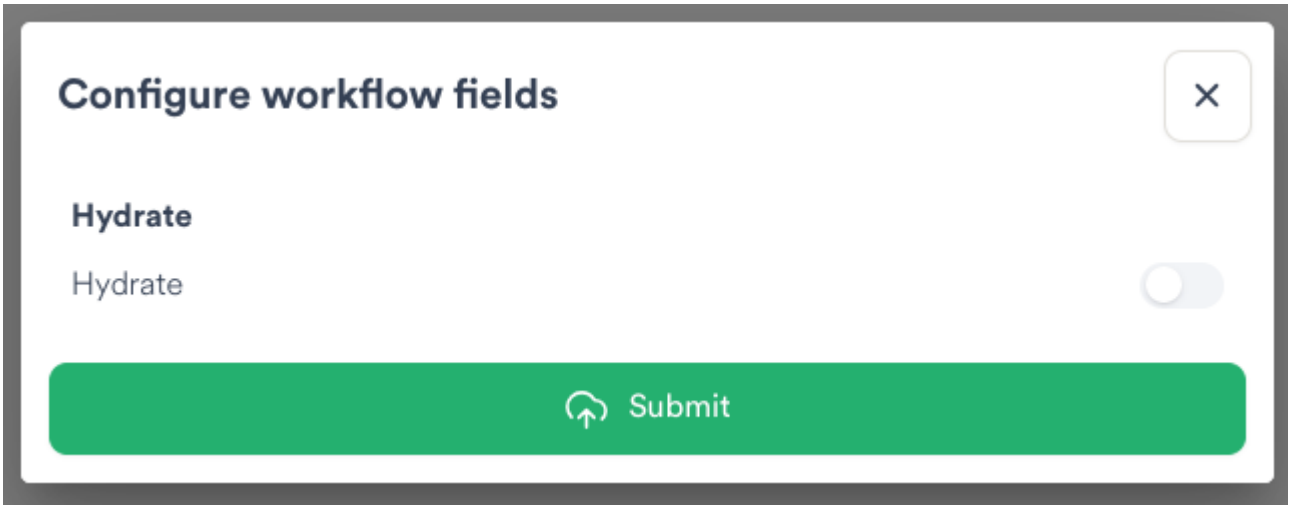
Note: Target files can only be imported to Locations in Spaces which match the target's "File match" pattern.

Note: Target files will be imported into the Space folder with their full remote path

For example, importing `path/in/cloud` to `/mmfs1/data/space/subpath` will result in `/mmfs1/data/space/subpath/path/in/cloud`

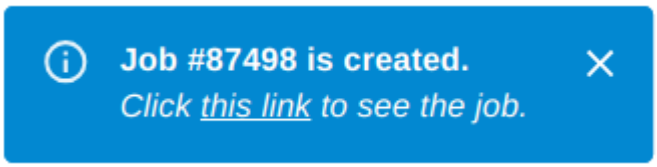
Click the Next button to enact the Job

Jobs with Workflows which require additional user decisions prior to enacting the Job raise the Configure workflow fields dialog:



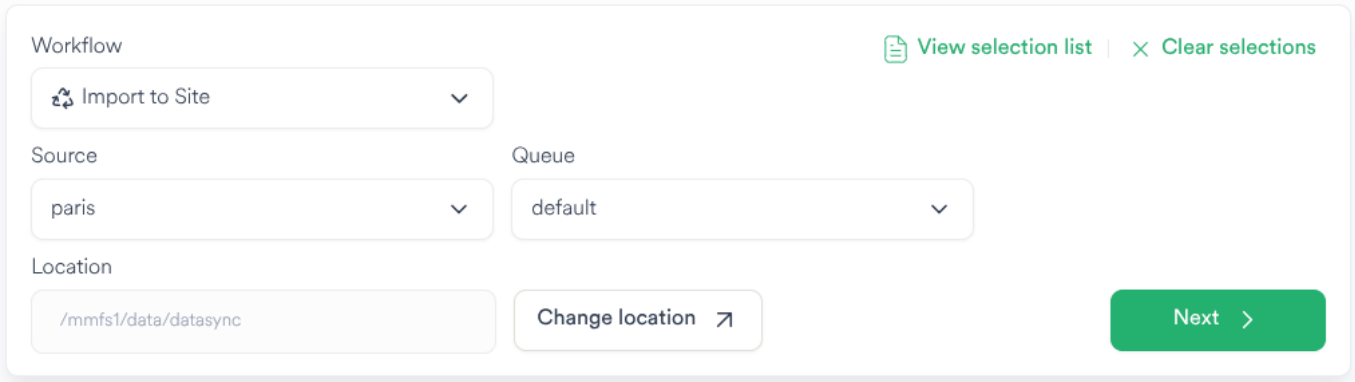
After entering the field information [if required], select the button at the bottom of the dialog to submit the Job.

Click the button at the bottom of the dialog to submit the Job.



Upon submitting the workflow, a notification indicates the job was created with a link to view details of that job

Job Types (Workflows)



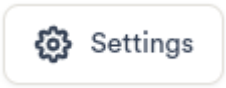
Default Workflows

Import to Site

The Import to Site workflow transfers the file metadata and data from the Ngenea target (E.G. an AWS cloud bucket). After successful transfer the metadata and data content of the file is present on the chosen site, within the chosen Location.

Global Settings

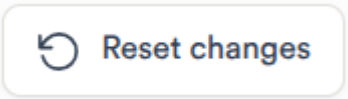
The global settings page controls settings which are applied across all pixstor sites participating in hub management.



Click the settings button to navigate to the Global Settings screen

Navigating the Settings page

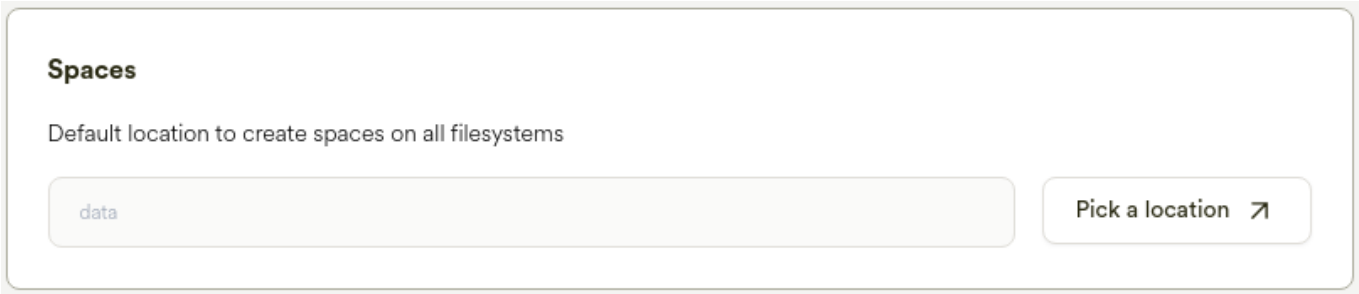
After changing a setting, click the Save button to apply the changes



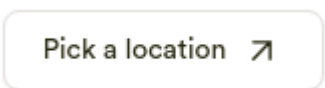
After changing a setting, click Reset Changes to reset any unsaved changes

Spaces

Spaces are created at a set directory path on a PixStor file system.



To set the directory path, click the



Clicking the `Pick a Location` button displays the file system browser opens in a new dialog

- Choose any file system on any Site
- Navigate the file system to a directory where Spaces are to be created
- Choose **Confirm**

All new spaces will be created at the chosen directory location **for all** filesystems.

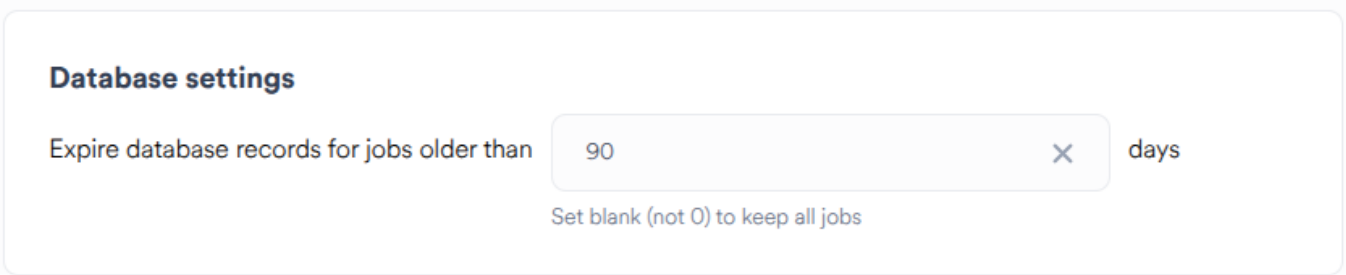
Parent directory paths will be automatically created on any Site where the chosen default location does not exist.

Database

Old job data can be retained in the database for a specific number of days so that the hub database does not fill up the system.

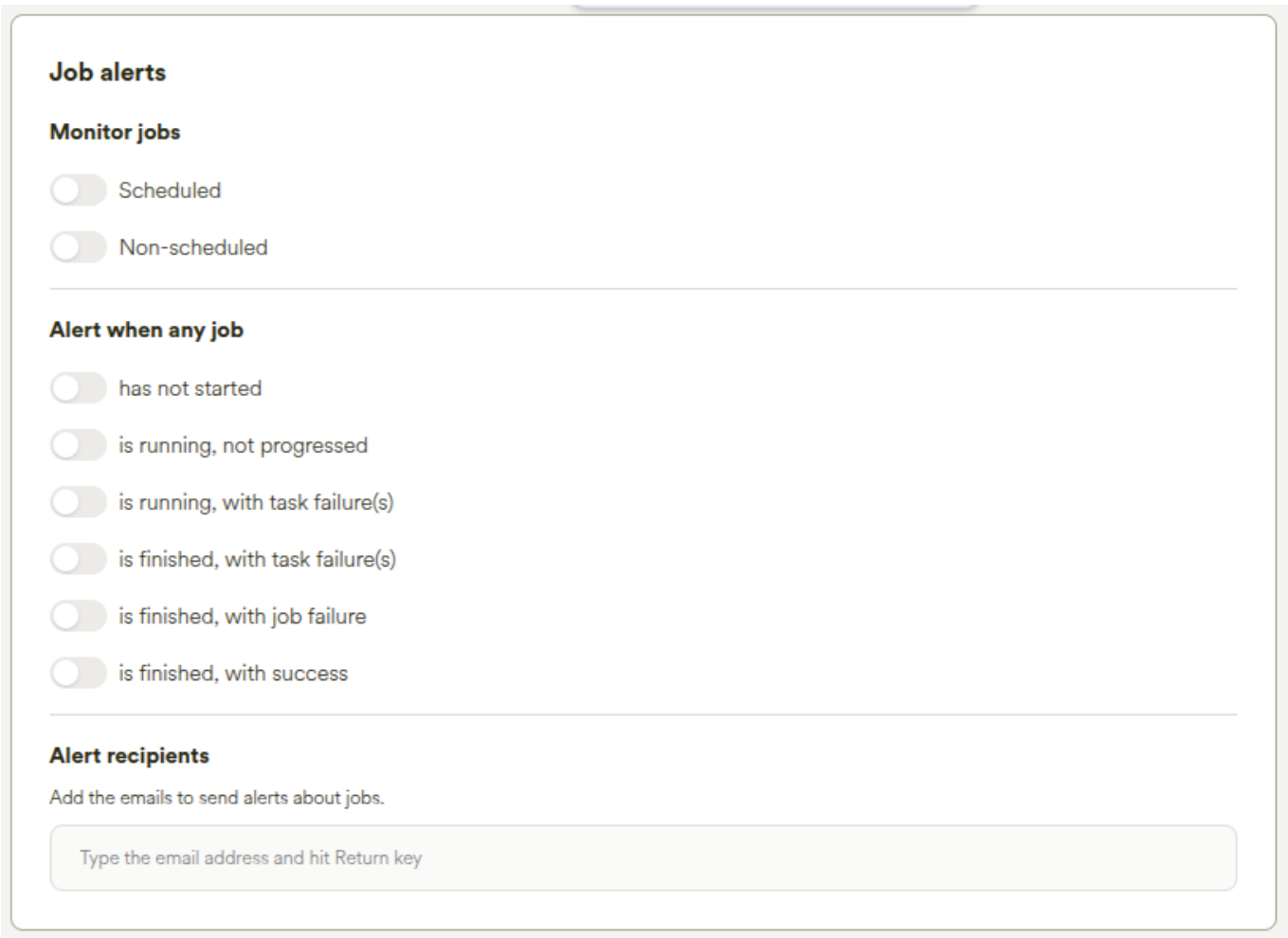
Database expiry can be configured by setting `jobs_ttl` in Global Settings page in UI.

Clicking the Global Settings button displays a form for updating the `jobs_ttl` in days. The `Save` button will be enabled once the user changes the value in the field.



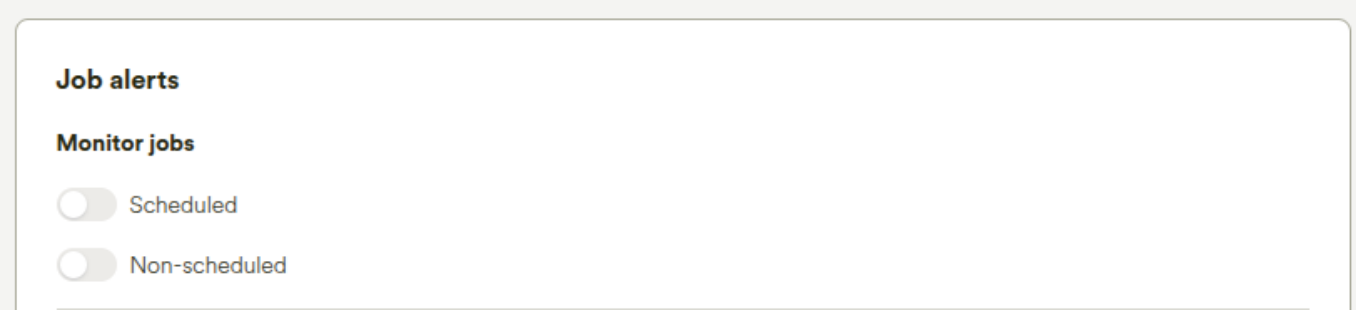
Job Alerts

Ngenea Hub allows us to configure alerts for jobs based on their type and state.



Job Types

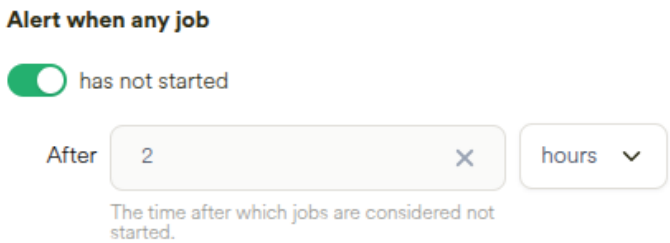
- **Scheduled** - Enables alerts for scheduled jobs. Default is `off`.
- **Non-scheduled** - Enables alerts for non-scheduled jobs. Default is `off`.



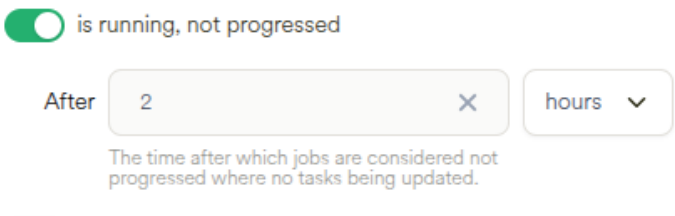
Note: At least one of Scheduled or Non-scheduled must be enabled to configure specific alerts. If both are off, all other alert options are disabled.

Alert Conditions

- **Has not started** - The alert is triggered when a job has not started after a configurable threshold. Default threshold is `2 hours`.



- **Is running, not progressed** - The alert is triggered when a job is running but not progressed after a configurable threshold. Default threshold is `2 hours`.

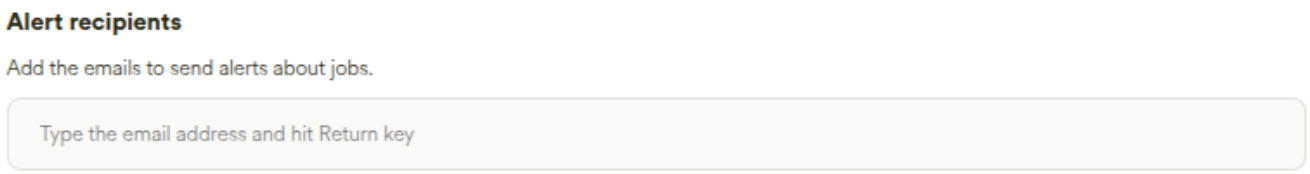


- **Is running, with task failure(s)** - Triggered if one or more tasks fail while the job is running and the job state is `Failing`.
- **Is finished, with task failure(s)** - Triggered if a finished job has one or more task failures and the job state is `Has Failures`.
- **Is finished, with job failure** - Triggered if a finished job fails completely and the job state is `FAILURE`.
- **Is finished, with success** - Triggered if a finished job completes successfully and the job state is `SUCCESS`.

Note: Thresholds can be adjusted for `not started` and `not progressed` jobs to control when alerts are sent.

Alert Recipients

Alerts can be sent to one or more email addresses. Enter an email in the multi-text box and press Enter to add it to the list. You can add multiple recipients this way.

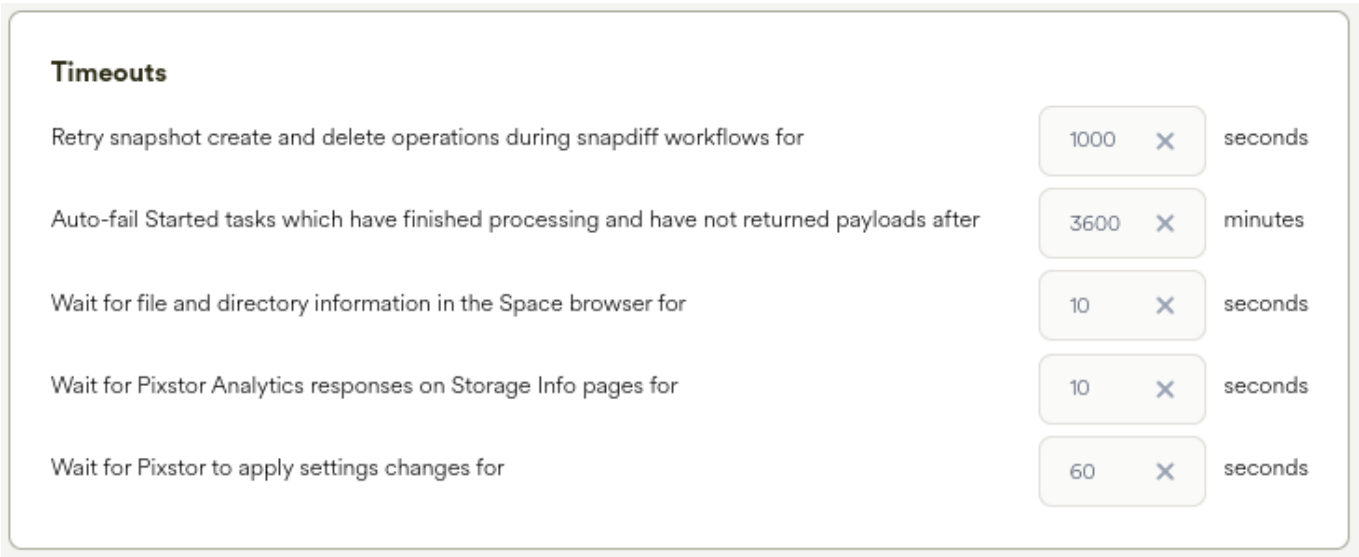


Timeouts

Hub manages many varied operations, at differing scale, across Sites which can be in different geo-locations.

As such, the time to complete operations may require tuning for the differing scenarios.

Additional timeout settings are available in the Administration Guide. See “Hub Configuration”.



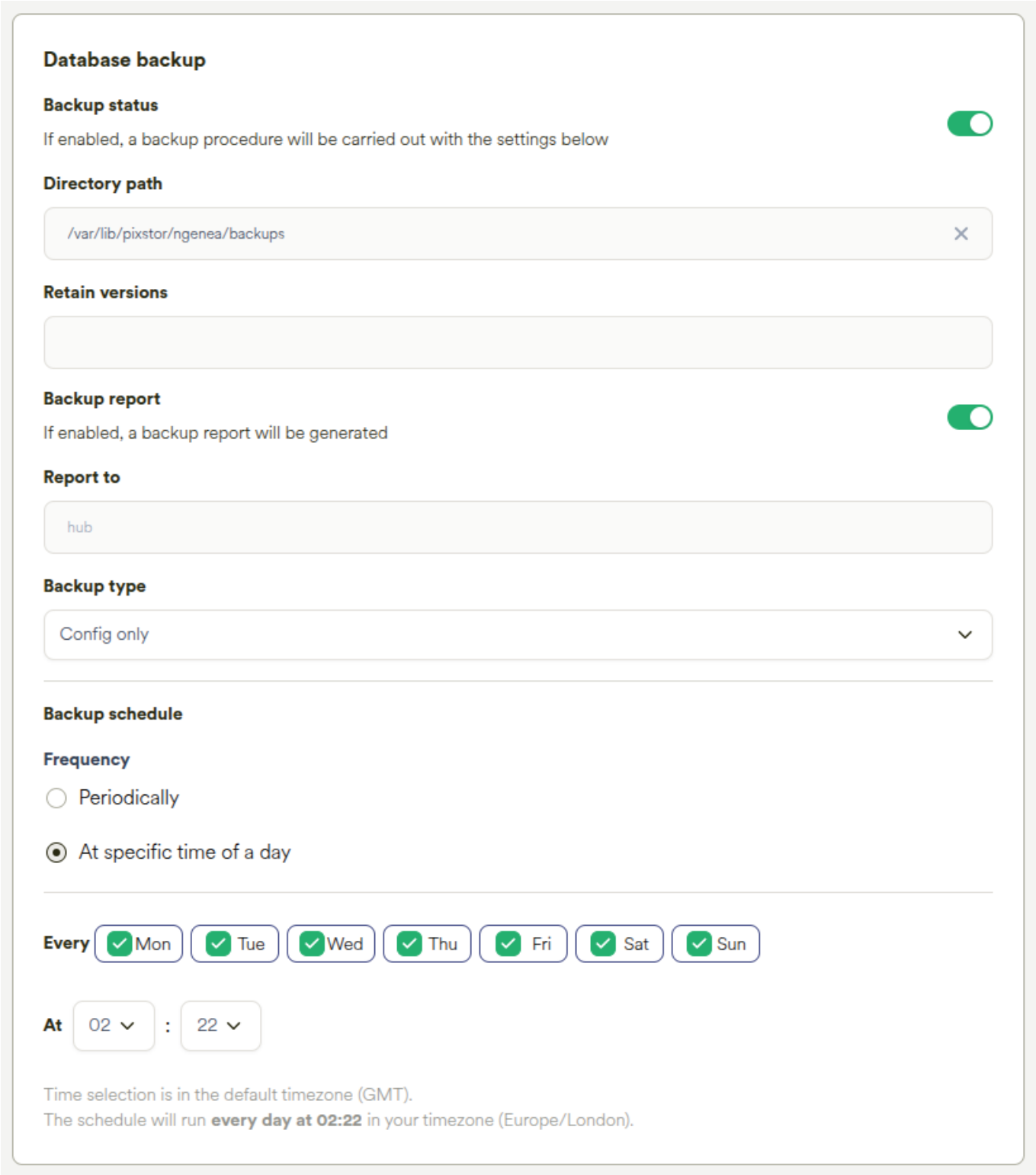
Setting	Description	Default
Retry snapshot create and delete operations during snapdiff workflows for	Where snapshots operations are numerous or blocked by cluster behaviour, continue to retry snapshot creation and deletion within the timeout window	1000 seconds

Setting	Description	Default
Auto-fail Started tasks which have finished processing and have not returned payloads after	If a Job's task has completed, but the results of the task have not been sent back to Hub (perhaps due to network, environmental or payload failure), set the task to state Failure after the timeout window has expired	3600 minutes
Wait for file and directory information in the Space browser for	During browsing of Spaces and similar file browser interfaces do not return disk usage and count information if no such information is received from the Pixstor Analytics after the timeout has expired	10 seconds
Wait for Pixstor Analytics responses on Storage Info pages for	During the browsing of Storage Info for a Site, do not return disk usage and count information if no such information is received from the Pixstor Analytics after the timeout has expired	10 seconds
Wait for Pixstor to apply settings changes for	When applying changes to Sites, wait the specified number of seconds for the Pixstor to respond to the action before failing the Settings task	60 seconds

Database Backup

Ngenea Hub provides automated database backup functionality to protect configuration data and, optionally, job history.

Backups can be scheduled and managed directly from the Global Settings page.



Setting	Description	Default
Backup status	If enabled, a backup procedure will be carried out with the given settings.	Enabled
Directory path	Directory where backup archives are stored.	/var/lib/pixstor/ngenea/backups
Retain versions	Number of backup archives to retain. If not set (None), all backups are kept and pruning does not run.	None

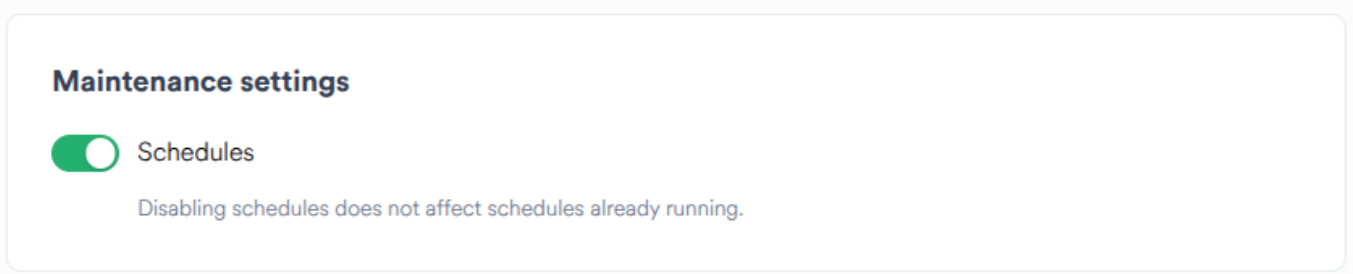
Setting	Description	Default
Backup report	If enabled, a backup report will be generated	Enabled
Report to	Destination for backup reports. When set to “hub”, a Job is created in the Hub UI containing backup and prune details.	hub
Backup type	Defines the data included in the backup: ConfigOnly - Static configuration files and configuration-related data Full - Includes ConfigOnly data plus job history data	ConfigOnly
Frequency	Defines how often backups are executed (periodically or at a specific time of day). The schedule uses the configured site timezone.	At specific time of day

Schedules

On the Schedules page, schedules can be enabled or disabled individually.

The Global Settings page provides the ability to disable all schedules at once.

Clicking the Global Settings button displays a “Schedules” toggle.



Set the toggle to disabled and click `Save` to disable all schedules.

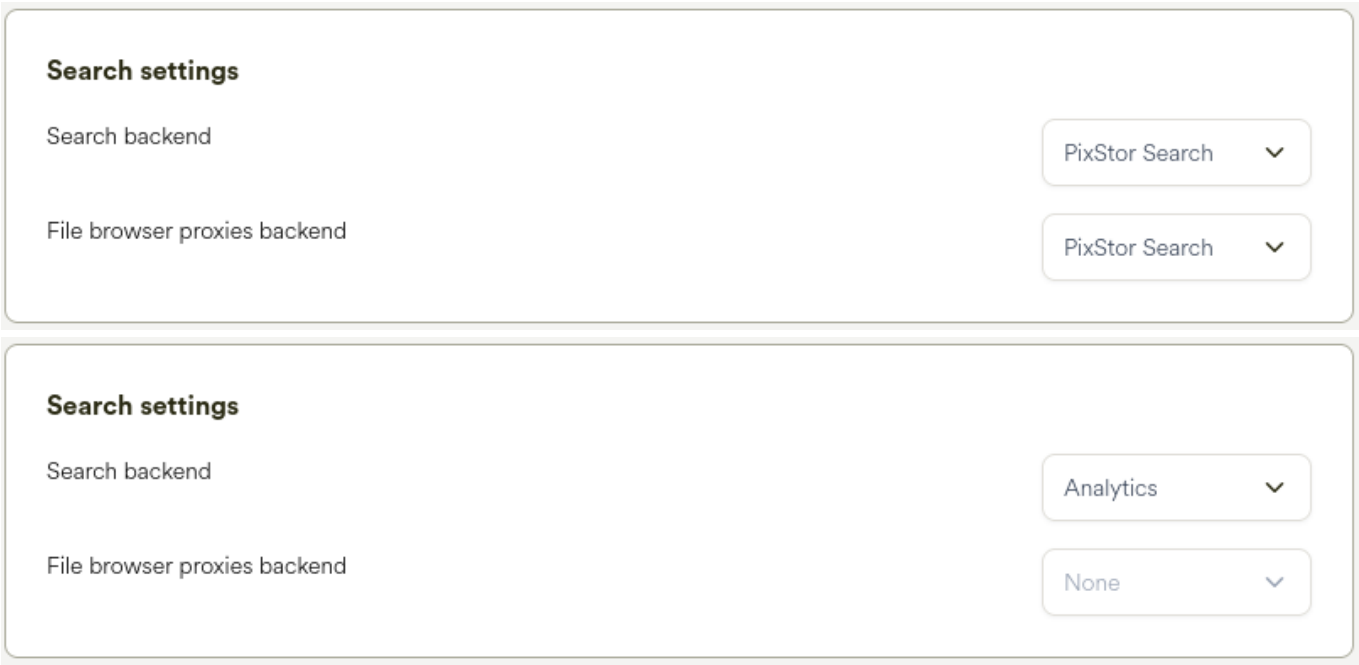
No new scheduled jobs will start for as long as Schedules are disabled globally.

Any scheduled jobs which have already started when schedules are disabled will run to completion, but no new jobs for that schedule will start after that.

Set the toggle back to enabled and click `Save` to re-enable schedules globally. Any schedules which were disabled individually will remain disabled after Schedules are re-enabled globally.

Search

The Global Settings page provides the ability to switch search settings.



The **Search backend** dropdown determines which backend service is used for the Hub search feature.

Two options are available: `Pixstor Search` and `Analytics`.

- Select `Pixstor Search` to perform searches using the Pixstor search service.
- Select `Analytics` to perform searches using the analytics service.

The **File browser proxies backend** dropdown determines which backend service is used to fetch proxy images (thumbnails) displayed in the Space browser.

The options available in the **File browser proxies backend** dropdown depend on the selection made in the **Search backend** dropdown.

When Pixstor Search is selected as the Search backend

The **File browser proxies backend** dropdown provides two options:

- `None` Disables proxy thumbnail retrieval for the file browser.
- `Pixstor Search` Enables fetching proxy thumbnails from the Pixstor search service.

When Analytics is selected as the Search backend

The **File browser proxies backend** dropdown only provides the following option:

- `None` Proxy thumbnails cannot be fetched from the analytics service and are therefore disabled.

These settings allow administrators to choose which backend service is responsible for search functionality and, when supported, for retrieving thumbnails used in the Space browser interface.

For more information about the Hub search feature, see Admin Guide.

Backups

PixStor provides the capability to backup data within a Space on a per-Site basis to specific Ngenea Targets.

Hub enables configuration to be set for the Ngenea Backup service running on pixstor sites.

Important: Best practice is to treat backup data separately from live data. Hub will only configure a backup to an Ngenea target set as a `backup-only` target type.

Configuring Backups

To perform a backup for a Space on a Site, the following actions must be undertaken:

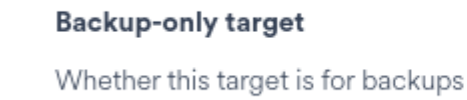
- An Ngenea Target is provisioned with `Backup-Only` enabled
- The Site from which to backup is enabled to participate in backups
- An appropriate backup schedule is set
- The Space is enabled for backup, selecting the Site and the Ngenea Target

Backup-Only Targets

An Ngenea Target can be utilised for backups of a Space.

Where a Space exists across multiple Sites, optionally, each site can perform its own backup of the Space.

Important: Multiple Sites are not permitted to backup to the same Ngenea Target.

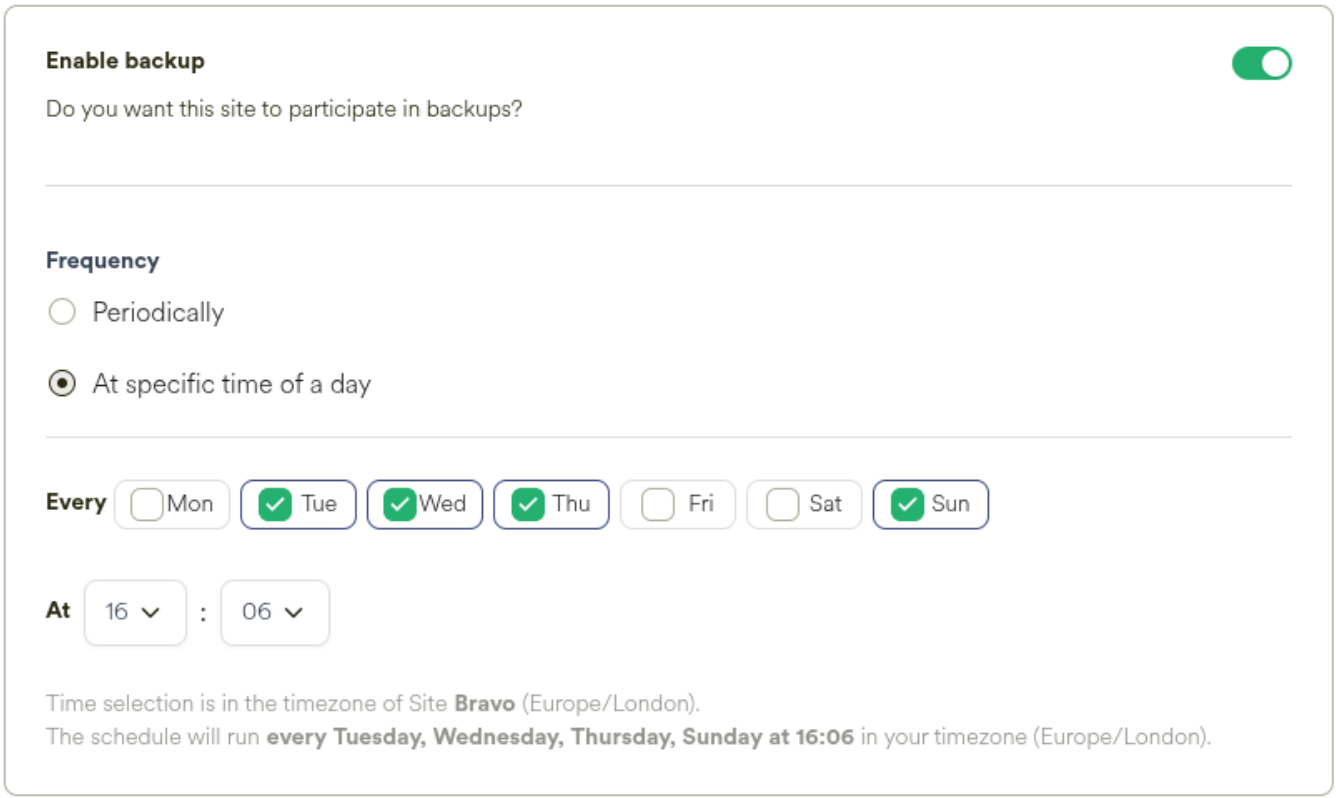


Enabling Site for Backup

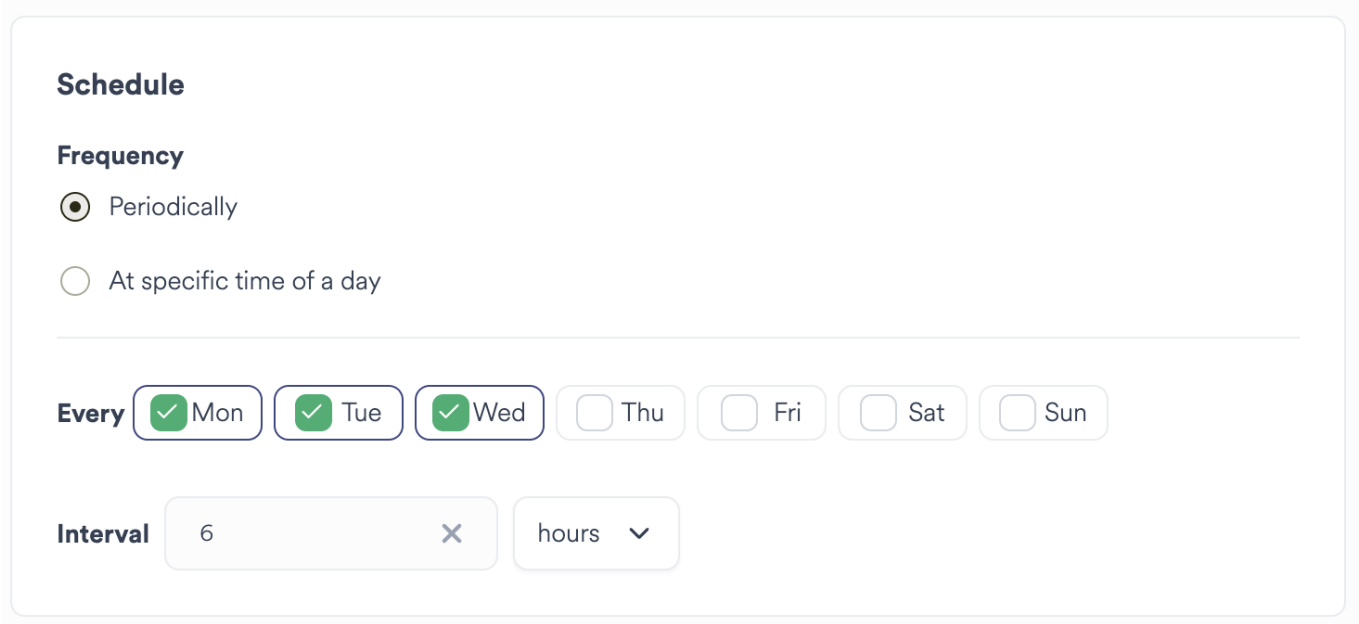
pixstor provides the capability to backup data within a Space on a per-Site basis to specific Ngenea Targets.

Hub enables configuration to be set for the Ngenea Backup service running on pixstor sites.

If the Site is enabled to participate in backups, each Space requires additional configuration to enable the per-Space backup.



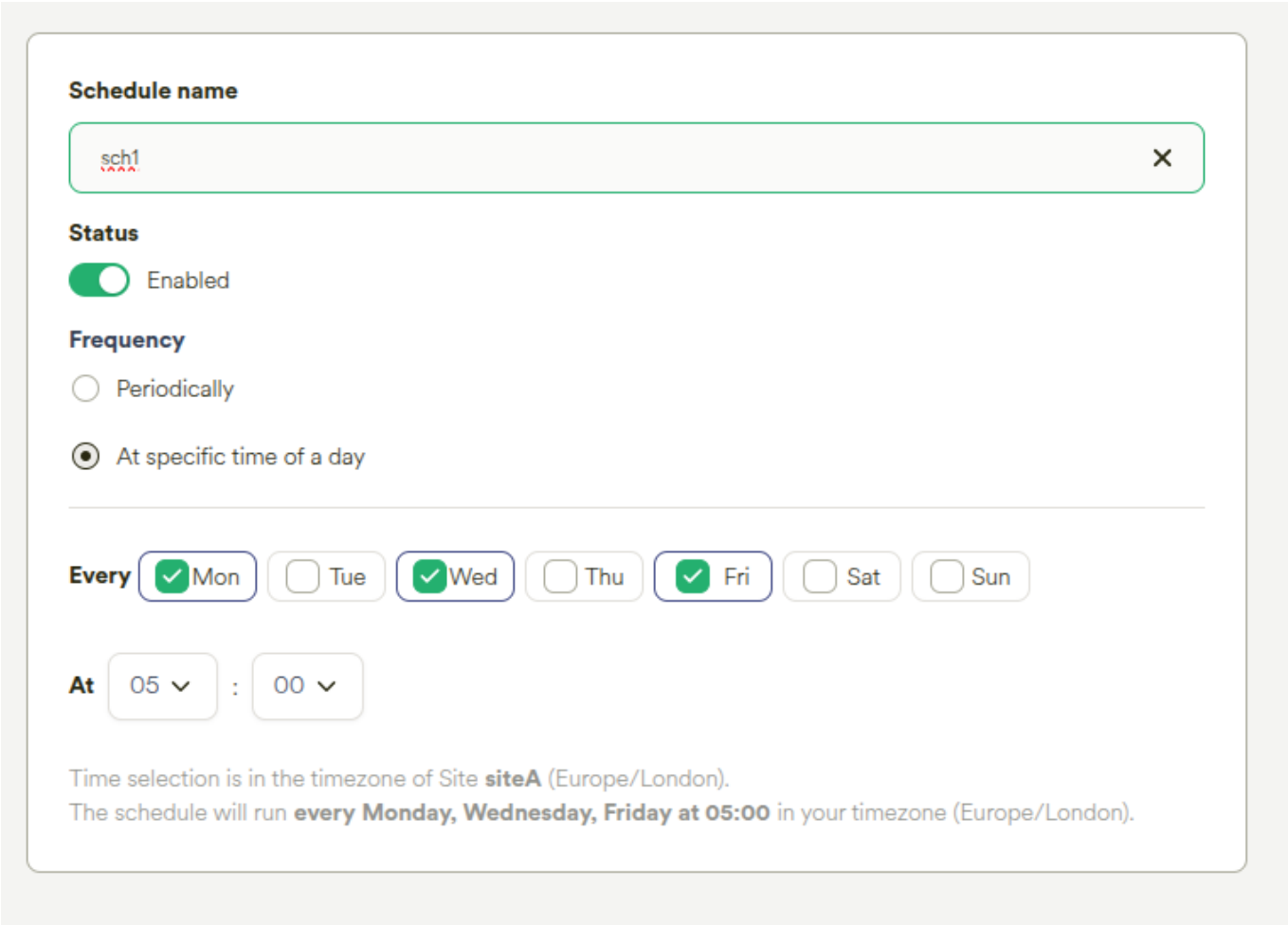
Determine the required frequency of the backup.



Choosing Periodically will ensure that the schedule will run on the next interval set.

E.G:

- 1 hour: The backup will run on the next hour (12.00, 13.00)
- 15 mins: The backup will run on the next 15 minute interval past the hour (15, 30, 45, 00)

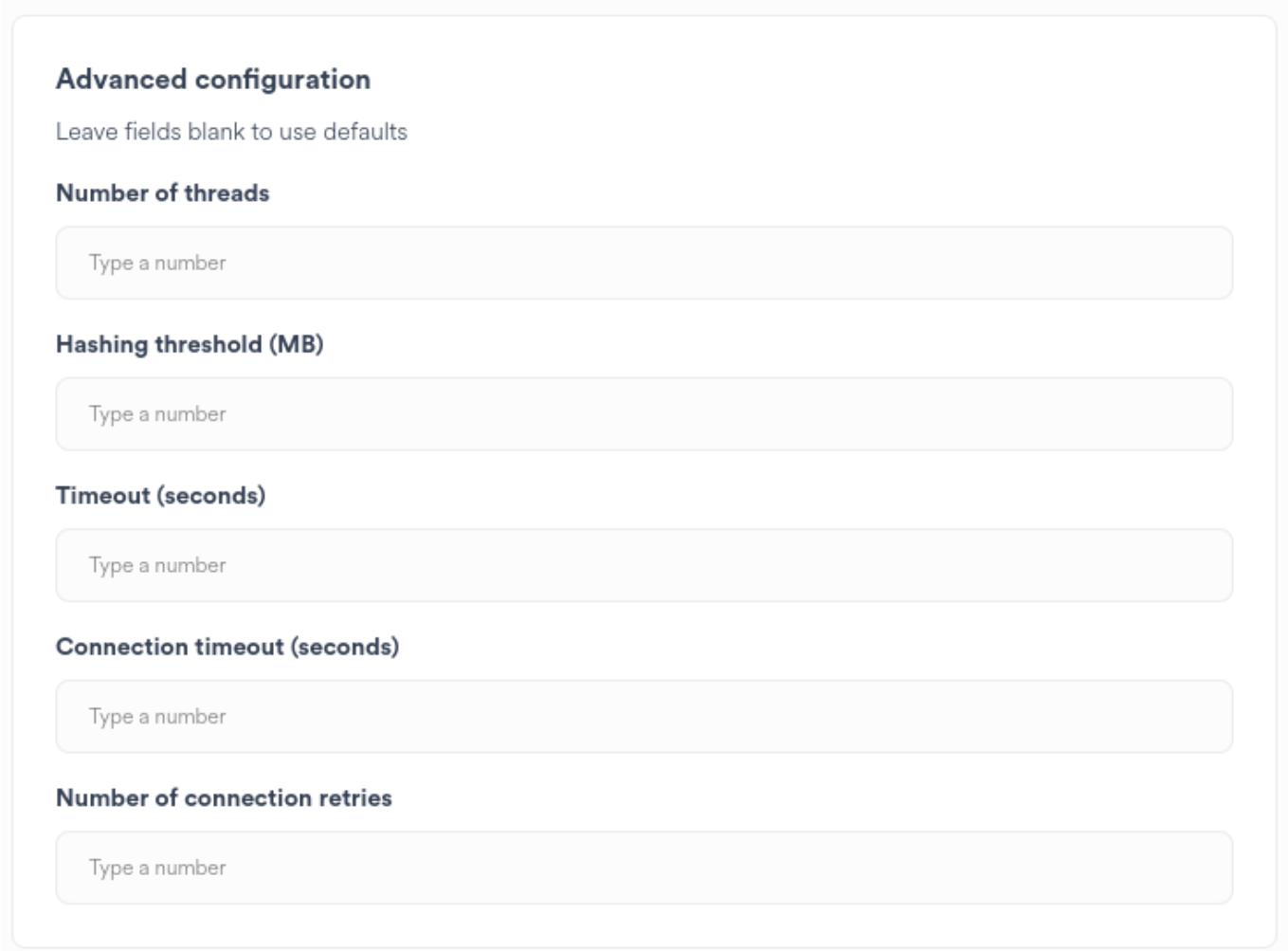


Choosing At Specific Time of A Day allows the Policy to be scheduled once per chosen day at a specific time of day.

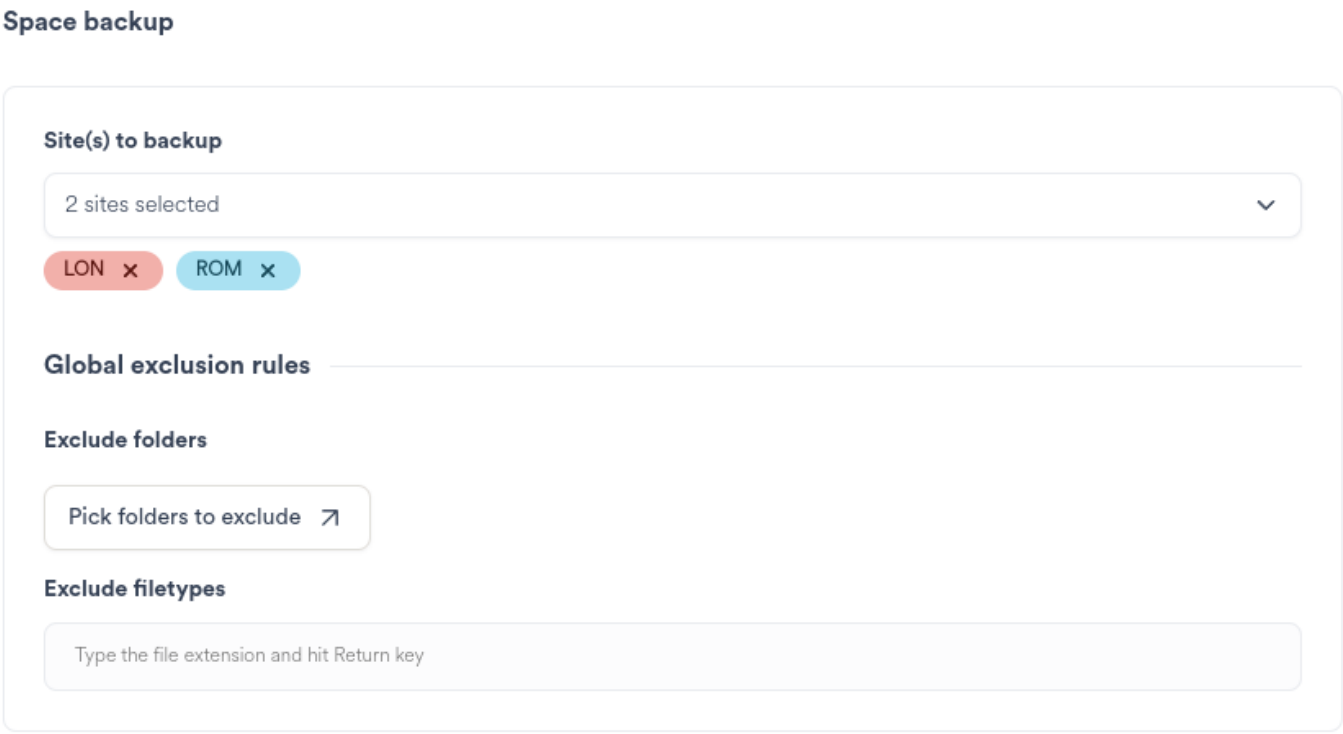
Hint: The schedule is run at the configured time in each site’s timezone

Optional tuning can be set for the backup operation in the Advanced Configuration.

Refer to the Ngenea Backup documention prior to applying any parameters.



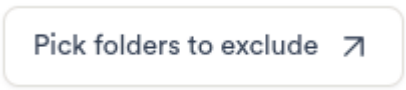
Enabling Space Backup



- Select the Sites which will perform backups for this space
- Define any Global Exclusions

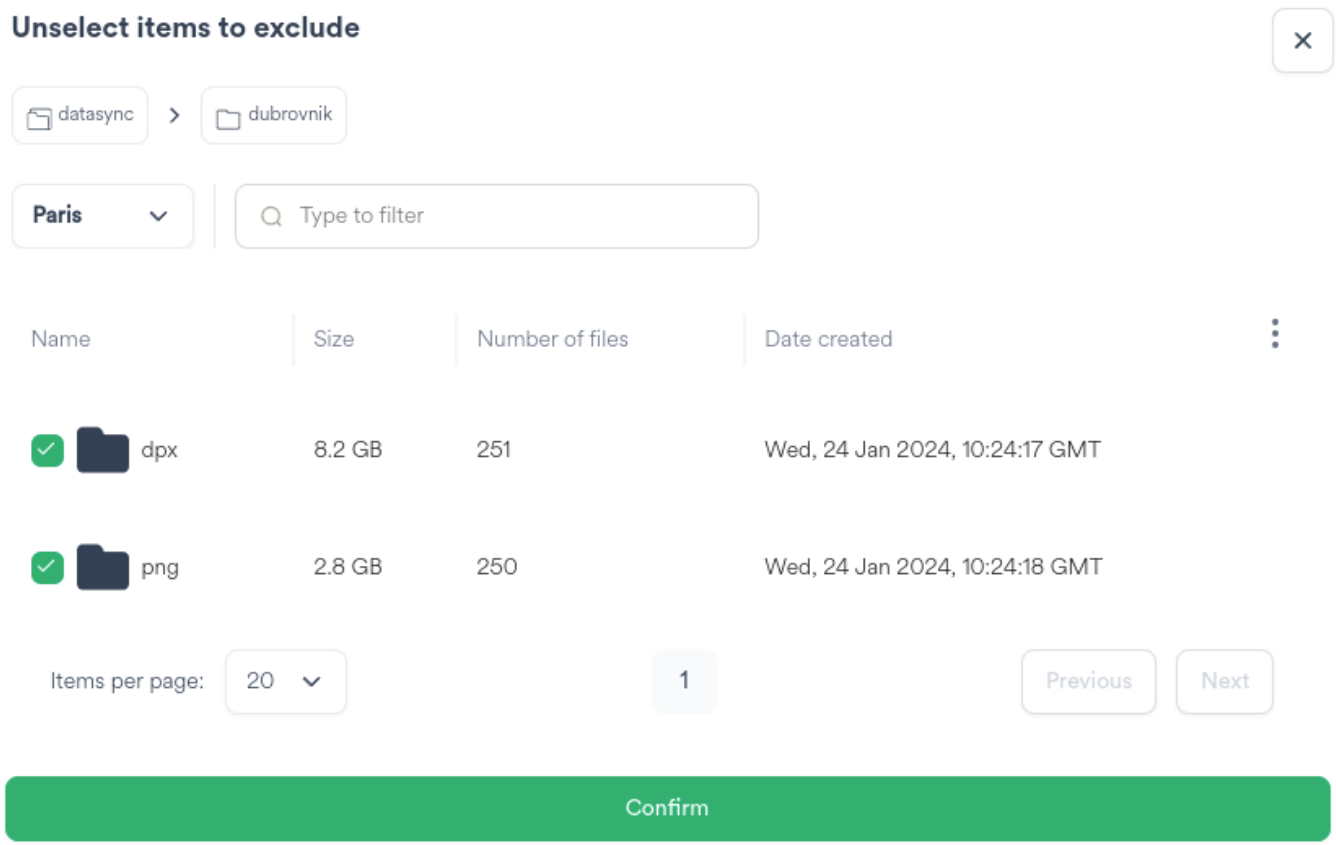
Global Exclusion Rules

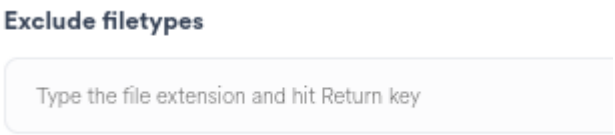
Global exclusions apply to all Site backup configurations for the Space unless overridden on a per-Site basis.



Click the Unselect folders to exclude button to raise the folder browser dialog

Use the folder browser to unselect items which will not be included (hence excluded) from backups.

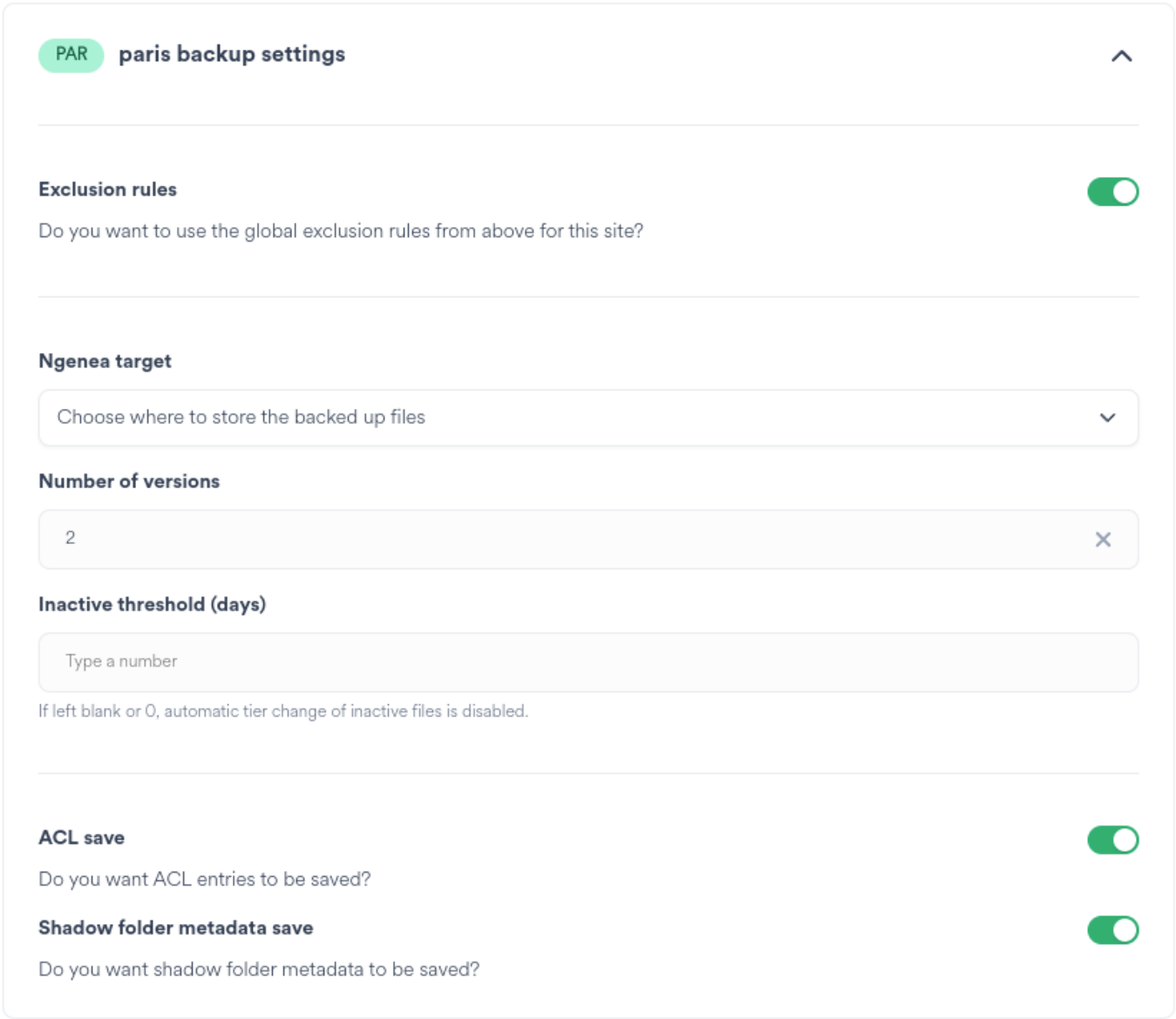




Define file extensions which must be excluded from backups across all Sites. E.G. `*.tmp`

Hint: If multiple Sites are selected, backup configuration is presented per-Site. Optionally each site can specify its own settings.

For each Site participating in Space backup a configuration panel is displayed:



Review the backup settings for each site:

Setting	Description
Exclusion Rules	By default the Site inherits the Global exclusion rules. To override the rules, deselect the slider and define any exclusions using the same methods as described above.
Ngenea Target	Choose an available Ngenea Target. Only targets configured as *Backup-Only target* and not currently in use for other Sites are available for selection.
Number of versions	Defines the maximum number of all file versions within a remote storage location for the targeted Fileset before Ngenea Backup prunes excess versions. If the number of versions of a file has exceeds the defined threshold, the oldest version will be deleted.

Setting	Description
Inactive threshold(days)	Defines the threshold of days for a defined CRITERIA (E.G.: modification) after which each file within the monitored Independent Fileset is validated when INACTIVE_THRESHOLD is enabled. Those files where their defined CRITERIA exceeds this value will be re-copied from the local PixStor filesystem to storage container into an in-active tier. All instances of the file in typical standard storage classes are transitioned to older file versions.
ACL save	If enabled, all of the related NSFv4 ACL entries for an ingested file will be included in the remote object's metadata when a file is backed up. This allows Ngenea to apply those NSFv4 ACL entries when recalled or stubbed.
Shadow folder metdata save	If enabled, additional data objects will be created within the remote storage target as this data is stored as an object within the cloud provider. If enabled, all the directory modification, creation and deletion operations between subsequent backup runs will be stored in remote storage. This allows the restoration or ngrecall of directories with NFSv4 ACL and POSIX permission support. This data will also be version controlled similarly to files. This operation will occur if the permissions or ACL entries are edited on any directories meaning that each change to ACLs will be updated with regular runs of backup.

Iris

Prerequisities

- IRIS feature enabled on Pixstor. For further information refer to the **PixStor Deployment And Configuration Guide**.
- Ngenea targets assigned to Spaces used for Iris
- ngclient configured

Iris Spaces

Enabling the Space for Iris:

- enables the Space data to be accessed via S3 protocol by the suite of Iris applications (Vision, AI+, Ngenea)
- creates an hourly schedule to send notifications of file changes within the Space to the Iris event queue. To change the schedule interval, edit the Schedule within Space settings or the Schedules page.

To enable Iris applications to access a Space, enable the Space for Iris.

Enabling a Space for Iris automatically creates an `iris-notify` schedule for the Space which can be managed from the Space settings or the Schedules page.

The `iris-notify` schedule periodically sends notifications of file events within the Space from the Pixstor file system to the centralised Iris notification queue.

On-Demand Notification

File event notifications can be triggered on-demand.

1. Select files in the Space browser
2. Submit the files using the “Iris Ingest” workflow

For further information refer to Using the Job Creator Panel.

The workflow will extract metadata from the selected files and generate a file ‘created’ notification.

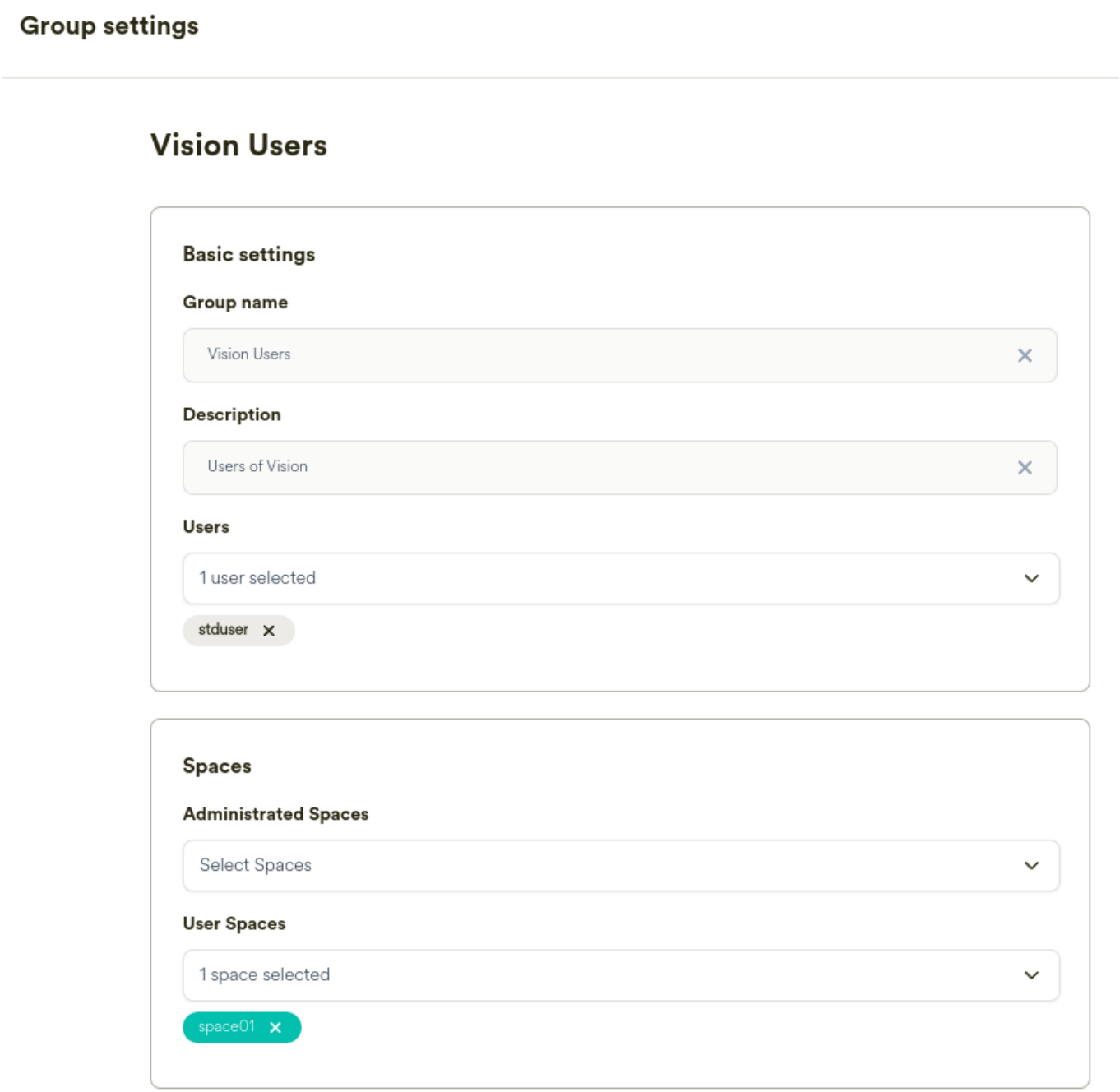
Authentication

Access to Spaces by users of Vision is granted through Hub permissions.

Users can successfully access Spaces (Vision:’Buckets’) where:

- the username in Hub and Vision matches
- access to the Space is granted to the Hub user through an appropriate Hub group

An example of a compliant setup is:



Users requiring to access Spaces in Vision do not require Space Administration privileges.

Administration Guide

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

This section provides a comprehensive guide for installing and configuring Ngenea Hub and Ngenea Worker on CentOS/RedHat systems.

Installing Ngenea Hub :

- **Prerequisites**: Details the required version of Ngenea Hub, network configurations, software compatibility, and necessary steps for offline installation and **Docker authentication**.
- **Online Installation - CentOS/RedHat**: Outlines steps to install Ngenea Hub RPM, create configuration files, start the service, and verify its status.
- **Offline Installation - CentOS/RedHat**: Explains the installation procedure for systems without internet access using local RPM files for both Ngenea Hub and necessary Docker images.

Installing Ngenea Worker: Describes prerequisites, installation of Ngenea Worker RPM, and configuration for communication with Ngenea Hub.

Installing Ngenea Hub

Prerequisites

Before proceeding with the installation of Ngenea Hub, ensure the following conditions are met:

Ngenea Hub Version: Ensure that you are installing the latest Ngenea Hub version.

> **Warning:** There is no direct upgrade path from Hub 1 to Hub 2.
>
> Please contact support for guidance on performing this upgrade.

Network Requirements: Ngenea Hub must be reachable from all Ngenea Workers on the following ports:

- 443/tcp - Ngenea Hub API.
- 8000/tcp - Ngenea Hub API, when exposed over HTTP. Note, this is disabled by default in Pixstor 6.11 and newer.
- 6379/tcp - Hub-internal Redis, when broker and/or result provider is configured as `redis`.
- 6010/tcp, 6011/tcp, 6012/tcp, 6013/tcp, 6014/tcp, 6015/tcp - Redis Cluster, when result provider is configured as `external_redis_cluster`.
- 5672/tcp - Hub-internal RabbitMQ, when broker provider is configured as `rabbitmq`.
- 18671/tcp - RabbitMQ, when broker provider is configured as `external_rabbitmq`.

All Ngenea Workers must be reachable from the Ngenea Hub node on the following ports:

- 22/tcp - SSH, for `pixstor-activate-site-queues`. Only required during bootstrap.

Software Compatibility: The interoperability page discusses compatibility between Ngenea Hub and Ngenea HSM. You can check compatibility using the Product Interoperability Matrix provided in the software documentation.

> **Note:** For compatibility between Ngenea Hub and Ngenea Worker, the version numbers must match up to the minor version. Specifically, if the version format is major.minor.patch, both Hub and Worker must share the same major and minor versions (the patch version may differ).

Post-Installation Features: After installation, you may want to enable additional features like Search. Follow the instructions in the Feature Set-up guide for specific feature configurations.

CentOS / RedHat Systems: Ensure that the target system where you will install Ngenea Hub is running either CentOS or RedHat operating systems.

CentOS / RedHat - Offline Installation (if applicable): For offline installations, ensure that you have both the `ngenea-hub` and `ngenea-hub-images` RPM packages downloaded and available on the target system.

Docker Authentication (if applicable): If your setup requires Docker authentication (not necessary for Pixstor systems), configure it by running:

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

Online Installation : CentOS / RedHat

1. **Transfer the Ngenea Hub RPM Package**: First, ensure that the `ngenea-hub-<version>.rpm` package is available on your target system. You can transfer the RPM file using various methods such as SCP, FTP, or other file transfer protocols.

2. **Install the Ngenea Hub Package**: Once the RPM file has been transferred, navigate to the directory where you stored the package and run the following command:

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

   Be sure to replace `<version>` with the actual version number of the Ngenea Hub RPM file.

> **Note:** For offline installation, please refer to section Offline Installation: CentOS / RedHat.

1. **Create Initial Configuration File (Optional)**: To manually set up the initial configuration for Ngenea Hub, you can edit the file located at `/etc/sysconfig/ngeneahub`. This configuration file includes settings for external queue systems and other deployment parameters. If you'd like to create or adjust the configuration file yourself, run:

```
ngeneahubctl createconfig
```

If you choose not to create it, the file will be automatically generated the first time you start the Ngenea Hub service.

2. **Enable and Start the Ngenea Hub Service**: To ensure the Ngenea Hub service starts automatically on system boot and also starts immediately, execute the following command:

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

3. **Check the Status of the Services**: Verify that both the Ngenea Hub and frontend services are running correctly. Use these commands to check their status:

   For the Ngenea Hub service:

```
ngeneahubctl status
```

   For the frontend service:

```
systemctl status ngeneahub-frontend.path
```

   These commands will display the current status of the services, indicating whether they are active and running or if any issues need to be addressed.

Offline Installation : CentOS / RedHat

When installing the Ngenea Hub on systems without internet access (due to network restrictions or other reasons), the service typically attempts to pull the necessary Docker images directly from Ngenea's software repository. However, if this is not possible, you can perform the installation offline by using RPM packages.

Here's how to proceed with the offline installation:

1. **Obtain the Required RPM Packages**: You need to download two RPM packages:

   - `ngenea-hub-<version>.rpm`: This RPM contains the main application for Ngenea Hub.
   - `ngenea-hub-images-<version>.rpm`: This additional RPM contains the Docker container images required for Ngenea Hub to operate, avoiding the need to pull them from the internet.

   Both of these RPMs should be available in the same location where you obtained the main Ngenea Hub software.

2. **Transfer RPMs to the Target System**: Copy or transfer the downloaded RPM files (`ngenea-hub-<version>.rpm` and `ngenea-hub-images-<version>.rpm`) to the system where you wish to install Ngenea Hub. This can be done using USB, external storage, or any other secure transfer method.

3. **Install the RPMs**: Once the RPM files are available on the target system, you can install them using the *rpm* command. Open a terminal on the target system and run the following command, replacing `<version>` with the appropriate version number of the files you downloaded:

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

- `yum install` is the command used to install RPM packages.

1. **Verify Installation**: After the RPM installation is complete, you can verify that Ngenea Hub is properly installed by checking that the service is running and the required Docker containers are available locally.

Installing Ngenea Worker

This section provides a step-by-step process for installing and configuring the Ngenea Worker. Following these instructions will enable you to install the worker on a system and have it interact with the Ngenea Hub.

Prerequisites

Ngenea Server Software

Ensure that the Ngenea Server software is installed and configured before proceeding with the worker installation.

File Systems Configuration with Auto-Inode-Limit

- The file systems on the nodes where you will install the ngenea-worker package must be configured with the auto-inode-limit.
- This setting is essential for the proper functioning of the worker.
- To configure auto-inode-limit on the file system, use the following command:

```
mmchfs <file_system_name> --auto-inode-limit
```

Replace `<file_system_name>` with the name of the file system.

Software Compatibility: The interoperability page discusses compatibility between Ngenea Hub and Ngenea HSM. You can check compatibility using the Product Interoperability Matrix provided in the software documentation.

Installing the Ngenea Worker Package

To install the ngenea-worker package, use the YUM package manager. This process assumes that your system is configured to use YUM repositories.

Run the following command to install the package:

```
yum install ngenea-worker
```

The `yum` command will download and install the ngenea-worker package along with its necessary dependencies. Wait for the installation to complete.

Worker Configuration

After the package installation, the next step involves configuring the worker so that it can communicate with the Ngenea Server.

Create or Edit the Worker Configuration File

The Ngenea Worker configuration is stored in a file located at: `/etc/ngenea/ngenea-worker.conf`.

If this file does not already exist, please create it. The configuration file utilizes the *.ini* format. You will need to include the following content within it:

```
[settings]
site = site1
api_host = localhost
api_port = 8000
api_secure = true
redis_port = 6379
gpfs_nodes = ["localhost"]
```

Here's how to adjust the example configuration based on your setup:

- **site**: Replace *site1* with the name of your specific site.
- **api_host**: Enter the hostname or IP address of your API server. If the API is hosted on the same machine as the worker, leave this as *localhost*.
- **api_port**: Enter the port on which your API server is running. The default value is *8000*.
- **redis_port**: This is the port Redis is running on. The default value is *6379*, and you should only change it if Redis is using a different port.
- **gpfs_nodes**: If you are using a General Parallel File System (GPFS), list your nodes here. If GPFS is installed on the same machine, *["localhost"]* is sufficient. You can specify multiple nodes if needed, using a comma-separated list within the brackets, E.G. *["node1","node2","node3"]*.

Handling SSL Configuration

SSL (Secure Sockets Layer) ensures encrypted communication between the Ngenea Worker and the API. Here's how to handle the SSL settings:

- **If you have a valid SSL certificate**, set `api_secure=true`.
- When `api_secure=true`, the system requires a valid SSL certificate, and self-signed certificates are not allowed.
- Ensure that a valid SSL certificate is installed on the server if using SSL (`api_secure=true`). For assistance with obtaining or installing an SSL certificate, refer to the documentation on External SSL provisioning.
- **If you do not have a valid SSL certificate** and want to disable SSL, set `api_secure=false`. This will allow the system to communicate without using SSL.
- **If you do not specify the ``api_secure`` option** in the configuration file, the system will automatically default to `api_secure=true`. If you do not have SSL, you must explicitly set `api_secure=false` to avoid issues with the connection.

Worker Configuration Settings - Introduction

The Ngenea Worker configuration involves setting up and managing workers that process tasks for the Ngenea Hub.

It includes options for API connectivity, task concurrency, Redis integration, and advanced logging settings. The system allows customization through custom configurations, multiple site setups, and debug-level adjustments to ensure efficient task handling and monitoring.

> **Note:** Click the **link** to learn more about Ngenea Worker settings and configurations!

> **Note:** Please refer to the link to understand more about Hub Configuration: {ref} `Hub Configuration Documentation <hub_config>`.

Joining the Worker to the Hub

To establish secure communications and register the worker with the hub, you will need to run the join command with a valid hub admin username and password.

Use the following command:

```
ngenea-worker join --user <USERNAME>
```

Example: Let's assume the username for the hub admin is hubadmin. To run the join command:

```
ngenea-worker join --user hubadmin
```

The system will then prompt you to enter the password for the hubadmin user.

After providing the password, the worker will authenticate with the hub, and you should see output similar to this:

```
hubadmin's password:
Attempting to auth the hub using the provided credentials...
Authenticated as user onprem-worker.
Storing the client certificates for the worker...
Client certificate stored at /etc/pki/tls/certs/onprem.crt
Client private key stored at /etc/pki/tls/private/onprem.key
Storing the root CA for NgeneaHub...
Root CA stored at /etc/pki/tls/certs/ng-hub-ca.crt
```

Enabling and Starting the Ngenea Worker Service

```
The steps for authentication and certificate storage are
automatically performed as part of the `ngenea-worker join`
command.
```

Once the worker is authenticated and certificates are stored, the next step is to ensure that the Ngenea Worker service is enabled and running. You will use the systemd system and service manager for this.

Enable and start the Ngenea Worker service using this command:

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

Verification: You can verify the service status by running:

```
systemctl status ngenea-worker
```

If everything is configured correctly, you should see an active and running status.

Product Interoperability Matrix

Ngenea Hub requires specific versions of Ngenea.

- All Ngenea Hub managed sites must use the same version of Ngenea
- Mixed versions are unsupported

Ngenea

The following table defines the supported Ngenea versions for each version of Ngenea Hub.

Ngenea Hub version	Ngenea version minimum	Ngenea version maximum
2.11.1	1.33.0-1	1.35.0-1
2.11.0	1.33.0-1	1.35.0-1
2.10.2	1.33.0-1	1.34.0-1

Ngenea Hub version	Ngenea minimum version	Ngenea maximum version
2.10.1	1.33.0-1	1.34.0-1
2.9.5	1.33.0-1	1.33.0-1
2.9.4	1.32.0-1	1.33.0-1
2.9.3	1.32.0-1	1.33.0-1
2.9.2	1.32.0-1	1.32.0-1
2.9.1	1.32.0-1	1.32.0-1
2.9.0	1.32.0-1	1.32.0-1
2.8.3	1.31.0-1	1.32.0-1
2.8.2	1.31.0-1	1.32.0-1
2.8.1	1.31.0-1	1.32.0-1
2.8.0	1.31.0-1	1.32.0-1
2.7.2	1.31.0-1	1.31.0-1
2.7.1	1.31.0-1	1.31.0-1
2.7.0	1.31.0-1	1.31.0-1
2.6.1	1.31.0-1	1.31.0-1
2.6.0	1.31.0-1	1.31.0-1
2.5.5	1.30.1-1	1.30.1-1
2.5.4	1.30.1-1	1.30.1-1
2.5.3	1.30.1-1	1.30.1-1
2.5.2	1.30.1-1	1.30.1-1
2.5.1	1.30.1-1	1.30.1-1
2.5.0	1.30.1-1	1.30.1-1
2.4.3	1.30.1-1	1.30.1-1
2.4.2	1.30.1-1	1.30.1-1
2.4.1	1.30.1-1	1.30.1-1
2.4.0	1.30.1-1	1.30.1-1
2.3.2	1.29.0-2	1.29.0-2
2.3.1	1.29.0-2	1.29.0-2
2.3.0	1.29.0-2	1.29.0-2
2.2.5	1.28.0	1.28.0
2.2.4	1.28.0	1.28.0
2.2.3	1.28.0	1.28.0
2.2.2	1.28.0	1.28.0
2.2.1	1.28.0	1.28.0
2.2.0	1.28.0	1.28.0
2.1.0	1.27.0	1.27.0
2.0.2	1.25.0	1.25.0
1.30.0	1.27.0	1.27.0
1.29.0	1.27.0	1.27.0
1.28.0	1.26.1	1.26.1
1.27.0	1.25.0	1.25.0
1.26.0	1.24.1	1.24.1

Ngenea Hub version	Ngenea minimum version	Ngenea maximum version
1.25.0	1.24.1	1.24.1
1.24.0	1.21.0	1.22.0
1.23.0	1.21.0	1.22.0
1.22.0	1.21.0	1.22.0
1.21.0	1.21.0	1.21.0
1.20.0	1.21.0	1.21.0
1.19.0	1.21.0	1.21.0
1.17.3	1.20.1	1.20.1
1.17.0	1.19.0	1.19.0
1.13.0-1.16.0	1.16.0	1.19.0
1.10.0-1.12.0	1.15.0	1.16.0
1.9.0	1.14.0	1.14.0
1.8.0	1.13.0	1.14.0
1.7.0	1.12.0	1.12.0
1.6.0	1.12.0	1.12.0
1.5.0	1.12.0	1.12.0
<=0.6.0	1.9.0	1.11.0

Event Based Orchestration

The following table defines the supported Event Based Orchestration versions for each version of Ngenea Hub.

Ngenea Hub version	Event Based Orchestration version
1.30.0	1.0.0

Upgrade

Warning: There is no direct upgrade path from Hub 1 to Hub 2.

Please contact support for guidance on performing the upgrade.

Note: Hub versions 2.3.0+ have additional requirements.

Ensure to review the Product Interoperability Matrix and align software versions accordingly.

If an upgrade is being performed from a version < 2.3.0 please note changes in the Installation and Upgrade guides.

Warning: Upgrading to 2.10

Upgrading to Hub 2.10 is a one-way process.

You **must** create a backup of your Ngenea Hub environment before you begin the upgrade process, which will allow you to revert to a prior version if required.

See Backing Up HUB Static Configurations for more information.

Once your backup is complete, you can safely proceed with the upgrade.

Warning: Upgrades from Hub < 2.4 must be performed in two steps, e.g. 2.3 -> 2.4 -> 2.10+.

Ensure to backup the Hub database prior to proceeding.

Before you start

Before upgrading, you must wait for any pending or active jobs to complete, otherwise they may be lost or present an incorrect future state.

Scheduled workflows must be temporarily disabled prior to upgrading 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 or by using Ngenea Hub backup management command Backing Up HUB Static Configurations

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

See NGCLIENT-WORKFLOWS for more information.

Stop Services

Note: Hub versions 2.1.0+ have additional requirements

When upgrading Hub 2.1.0+, the additional service ngenea-worker-frontend.path is required to be actioned

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
systemctl stop ngeneahub-frontend.path
```

Warning: Not stopping the workers before upgrading may result in jobs being stuck as `PENDING`. If this happens, restarting the workers using `systemctl restart ngenea-worker` will allow jobs to start running again.

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
```

Upgrade Configurations

Warning: If upgrading from Ngenea Hub version 1.17 or older, you must convert worker configuration to 1.18+ format. Please refer to the *appropriate worker configuration* and *worker join* steps as described in Installing Ngenea Worker. Installation of the worker is not required as this has been achieved in the prior Upgrade Packages step.

Ngenea Hub versions 2.3+ have additional requirements. If an upgrade is being performed from a version < 2.3.0 please note changes in the Installation and Upgrade guides require prior LDAP connections require to be re-authenticated. Ensure to review the Hub Authentication sections applying the preferred setup accordingly.

Restart Services

First, startup Ngenea Hub

```
systemctl start ngeneahub
systemctl start ngeneahub-frontend.path
```

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
systemctl status ngeneahub-frontend.path
```

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 First Steps

You can now log into the Ngenea Hub User Interface (UI).

- Open a web browser and navigate to : `http://server.address:8000`
- Replace `server.address` with the actual IP address or domain of your server.
- Enter your admin user credentials

If you don’t have a hub admin user set up yet, create one by following the steps:

- Open a terminal on the server where Ngenea Hub is installed. Run the following command to create a new admin user:

```
ngeneahubctl adduser
```

- You will be prompted to provide a username and password. These credentials will be used to log into the Ngenea Hub UI later.
- Replace `SITENAME` with the required name of your site and run the following command:

```
ngeneahubctl addsite SITENAME
```

- Once you've created the admin user and registered the site, you can now log into the Ngenea Hub User Interface (UI).

Hub Configuration

Settings

Ngenea Hub requires some important settings for correct operation. These settings tell the system how to connect to its database, process tasks, and secure data.

The main configuration file is located at:

`/etc/sysconfig/ngeneahub`

This file contains all the necessary information for Ngenea Hub to function. If you do not set these values, the system will use default settings.

Mandatory Settings

Here's a simple explanation of each important setting:

Setting	Description
DJANGO_SECRET	A secret key that protects important system data, like user sessions. Think of it as a password used internally by the system.
POSTGRES_DB	The name of the database where Ngenea Hub stores its information.
POSTGRES_USER	The username that allows the system to connect to the database.
POSTGRES_PASSWORD	The password used to access the database securely.
WORKER_THREADS	Controls how many tasks can be handled at the same time. Default: **2**.
DAG_THREADS	Helps process multiple tasks at the same time. Default: **7**.
REFRESH_THREADS	Controls how many jobs can refresh at the same time. DefaultL **2**
BROKER_PROVIDER	Defines the task queue broker service (**rabbitmq**, **redis**, **external_rabbitmq** or **external_redis**). Default: **redis**.

Setting	Description
RESULT_PROVIDER	Defines the task result storage (**redis**, **external_redis**, or **external_redis_cluster**). Default: **redis**.
RABBITMQ_USER	The username for RabbitMQ (if used). Default: **ngeneahub**.
RABBITMQ_PASSWORD	The password for RabbitMQ (if used). Default: **ngeneahub**.
RABBITMQ_VHOST	A virtual host inside RabbitMQ for organizing tasks. Default: **nghub**.
TASK_DAEMON_BATCH_SIZE	Number of tasks that update their status when a job starts. Default: **100**.
DAG_REFRESH_INTERVAL	Controls how often in seconds the job refresh daemon polls for new jobs to refresh. Default: **1**
DAG_REFRESH_MAX_AGE	Controls how long in seconds the job refresh daemon will wait for a refresh to complete before retrying it. Default: **600**
DAG_REFRESH_BATCH_SIZE	Controls the maximum number of job refreshes that will be queued per refresh interval. Default: **5**
JWT_PRIVATE_KEY	A key used to sign user authentication tokens. Generated on first startup.
JWT_PUBLIC_KEY	A key used to verify authentication tokens. Generated on first startup.
JWT_EXPOSED_JSON	Stores extra details related to authentication security.
GF_SECURITY_ADMIN_PASSWORD	Password for accessing **Grafana**, the system monitoring tool.
HUB_PORT	The port where Ngenea Hub runs. Default: **8000**.
WEB_BIND_IP	The IP address used for connecting to the system. Default: **0.0.0.0**.
SHARED_SECRET	A secret key for secure communication within the system.
DJANGO_ENCRYPTED_FIELDS_KEY	Encryption key used to secure sensitive database information.
DJANGO_EJF_CRYPTER_KEY	Additional security key for encrypting Django responses.

Optional settings

Settings and Description

- **REDIS_HOST**: The address of the Redis broker or results store, can be external to the hub services. Defaults to the container service address *redis*.

- **REDIS_PORT**: The port of the Redis broker or results store, can be external to the hub services. Defaults to *6379*.
- **REDIS_PASSWORD**: The password for an external Redis broker or results store, this setting is exclusively for external services.
- **REDIS_TLS**: Enables TLS support for external Redis services, defaults to *False*.
- **RABBITMQ_HOST**: The address of a RabbitMQ service. Can be external to the hub services. Can be a comma-separated list. Defaults to the container service address *rabbitmq*.
- **RABBITMQ_PORT**: The port of the RabbitMQ broker service, defaults to *5671*.
- **RABBITMQ_TLS**: Enables TLS support for external RabbitMQ services, defaults to *False*.
- **RABBITMQ_QUEUE_TYPE**: Queue type to use when creating queues in RabbitMQ, when there is only one *RABBITMQ_HOST*. Either *classic* or *quorum*. Defaults to *classic*. The type of a queue can't be changed after it has been created. Configure as *quorum* if you plan to scale out RabbitMQ.
- **POSTGRES_HOST**: The address of the PostgresSQL database instance. Defaults to container service address.
- **POSTGRES_PORT**: The port of the PostgresSQL database instance. Defaults to container service port.
- **POSTGRES_TARGET_SESSION_ATTRS**: The sesssion properties to apply to the connection to a PostgresSQL database instance. Defaults to any.
- **WORKERS**: The number of workers for API requests. More workers allow the system to handle more requests simultaneously. Default: **8**. This is deprecated. Please use **GUNICORN_WORKERS** instead.
- **GUNICORN_THREADS**: The number of worker threads for handling requests. Typically, use 2–4 × CPU cores; adjust as needed for your workload. Default: **2**
- **GUNICORN_WORKERS**: The number of gunicorn workers for API requests. More gunicorn workers allow the system to handle more requests simultaneously. This setting will override any **WORKERS** settings provided. Default: **8**
- **API_TIMEOUT**: The timeout in seconds for API requests. Default: **600** seconds.
- **GATEWAY_TIMEOUT**: The timeout for requests passing through nginx. Should be greater than or equal to API_TIMEOUT. Default: **600** seconds.
- **CONSUMER_TIMEOUT**: The timeout for RabbitMQ consumer delivery acknowledgment in seconds. Default: **10800000** seconds (3 hours).
- **PUBLIC_URL**: Configurable base URL for the hub stack. Must not end in a trailing slash.
- **CELERY_THREADS**: The number of concurrency threads for handling multiple background tasks. More threads mean faster processing of tasks.
- **EVENT_THREADS**: The number of concurrency threads for handling event reporting tasks. More threads help report events faster.
- **RESULT_THREADS**: The number of concurrency threads for handling result reporting tasks. More threads mean quicker reporting of task results.
- **HEARTBEAT**: Enables or disables celery heartbeats to keep connections alive. Default: **True** (enabled).
- **GOSSIP**: Enables or disables celery gossip communication. Default: **False** (disabled).
- **MINGLE**: Enables or disables celery mingle for coordination. Default: **False** (disabled).

- **REDIS_CLUSTER_STARTUP_NODES**: Comma-separated list of Redis Cluster nodes in the form *host:port*. For use when Hub is configured to use Redis Cluster as its result provider.
- **REDIS_HEALTH_CHECK_INTERVAL**: The interval in seconds between health checks for Redis backend. Default: **60** seconds.
- **REDIS_TCP_BACKLOG**: The number of pending requests to Redis. Higher values help avoid slow connections in high request environments. Default: **511**.
- **REDIS_SOCKET_TIMEOUT**: The timeout in seconds for resetting idle Redis backend connection sockets. Default: **60** seconds.
- **CELERY_SOCKET_TIMEOUT**: The timeout in seconds for resetting idle Celery broker connection sockets. Default: **60** seconds.
- **CELERY_CONNECTION_TIMEOUT**: The timeout in seconds for resetting idle connections via Redis broker. Default: **60** seconds.
- **BROKER_CONNECTION_RETRY_ON_STARTUP**: This enables all internal workers to retry to connect to the broker on startup only. Defaults: **True**.
- **BROKER_CONNECTION_RETRY**: This enables all internal workers to retry to connect to the broker. Defaults: **True**.
- **BROKER_CONNECTION_MAX_RETRIES**: This controls how many attempts the internal works will attempt once connection is lost. Defaults: **10**.
- **BROKER_CANCEL_ON_CONNECTION_LOSS**: This kills long running tasks in the internal workers if connections has been lost. Defaults: **True**.
- **CELERY_BROKER_HEARTBEAT**: Used to detect if a connection was lost between the broker and clients. Default: **10** seconds. (RabbitMQ only)
- **CELERY_BROKER_HEARTBEAT_CHECKRATE**: How often to check for missed broker heartbeats. Default: 2 per interval.(RabbitMQ only)
- **CELERY_SOCKET_KEEPALIVE**: Enable TCP socket keepalive. Default: true. (Redis only)
- **CELERY_SOCKET_KEEPALIVE_IDLE**: Start probing for liveness after the socket has been idle for this long. Default: 60s. (Redis only)
- **CELERY_SOCKET_KEEPALIVE_COUNT**: Number of probe attempts before an idle socket is considered dead. Default: 5. (Redis only)
- **CELERY_SOCKET_KEEPALIVE_INTERVAL**: Interval between idle socket probe attempts. Default: 10 seconds(Redis only)
- **EXPIRE_OLD_JOBS_INTERVAL**: Cron schedule for expiring old jobs. Default: `0 0 * * *` (daily).
- **REMOVE_OLD_SEARCH_RESULTS_INTERVAL**: Cron schedule for removing old search results. Default: `0 0 * * *` (daily).
- **INVALIDATE_CANCELLED_JOB_TASKS_INTERVAL**: Cron schedule for revoking cancelled job tasks. Default: `0 * * * *` (every hour).
- **CLEANUP_OLD_EVENTS_INTERVAL**: Cron schedule for cleaning up old snapdiff events. Default: `0 * * * *` (every hour).
- **INACTIVE_TASKS_INTERVAL**: Cron schedule for invalidating inactive tasks. Default: `0 * * * *` (every hour).
- **SYNC_SITE_SETTINGS_INTERVAL**: Cron schedule for syncing site settings in the database. Default: `0 * * * *` (every hour).
- **SYNC_GLOBAL_SETTINGS_INTERVAL**: Cron schedule for syncing global settings across all sites. Default: `0 * * * *` (every hour).
- **REFRESH_SITE_ANALYTICS_INTERVAL**: Cron schedule for refreshing site analytics. Default: `37 */12 * * *` (every 12 hours).

- **SYNC_STORAGE_POOLS_INTERVAL**: Cron schedule for syncing storage pools. Default: `*/30 * * * *` (every 30 minutes).
- **SYNC_REMOTE_SERVERS_INTERVAL**: Cron schedule for syncing remote servers. Default: `0 0 * * *` (once a day).
- **SYNC_REMOTE_NODECLASSES_INTERVAL**: Cron schedule for syncing remote nodeclasses. Default: `0 0 * * *` (once a day).
- **SYNC_SPACES_QUOTA_INTERVAL**: Cron schedule for syncing spaces' quotas. Default: `*/30 * * * *` (every 30 minutes).
- **SYNC_SPACES_INTERVAL**: Cron schedule for syncing spaces. Default: `0 * * * *` (every hour).
- **SYNC_ALERTS_INTERVAL**: Cron schedule for syncing alerts from all sites. Default: `* * * * *` (every minute).
- **EXPIRE_OLD_FSOBJECTS_INTERVAL**: Cron schedule for expiring old file system objects. Default: `0 0 * * *` (daily).
- **REMOVED_QUEUE_CLEANUP_INTERVAL**: Cron schedule for cleaning up removed queues. Default: `24 * 60` (every day).
- **CERTIFICATE_LIFE_TIME_DAYS**: Lifetime of SSL certificates in days. Default: **36500** (100 years).
- **QUEUE_ONLINE_CHECK_INTERVAL**: Interval in seconds to check if a queue is online. Default: **10** seconds.
- **SCHEDULE_LOCK_EXPIRY**: Time after which the schedule lock should expire (in seconds) if the holding process dies without releasing the lock.
- **OVERWRITE_COMPOSE**: Enables or disables ngeneahubctl overwriting changes to the compose file when running any service operations

Broker Settings

The messaging queue configuration for Ngenea Hub is a crucial component in managing communication between workers and the Hub.

This system supports two main brokers: **Redis** and **RabbitMQ**, offering flexibility depending on your specific requirements and infrastructure. Configuring the messaging queue is handled through various settings in the `/etc/sysconfig/ngeneahub` file, along with specific configuration files for Redis and RabbitMQ.

In the sections that follow, we will delve into the details of selecting and configuring the messaging queue, including how to manage Redis and RabbitMQ settings, enable the RabbitMQ admin interface, and understand the limitations of these configurations within the Hub environment.

For more details, refer to Hub Messaging Queue Configuration.

RabbitMQ HA

Ngenea Hub supports the use of High Availability (HA) RabbitMQ, using Quorum Queues, as the broker provider.

RabbitMQ HA replicates queues to additional hosts to support failover in the event that one or more hosts go down.

To use RabbitMQ HA as the broker provider, configure

```
BROKER_PROVIDER=external_rabbitmq
RABBITMQ_HOST=my-pixstor-mn01,my-pixstor-ng01,my-pixstor-ng02
RABBITMQ_PORT=5671
# other RABBITMQ_* settings
```

Configure all the hosts in the RabbitMQ cluster as a comma-separated list in `RABBITMQ_HOST`. The same `RABBITMQ_PORT` will used for all hosts by default. If each host uses a different port, the ports can be included in `RABBITMQ_HOST`, for example `host1:1234,host2:5678,...`

If multiple hosts are configured, Hub will assume that queues should be `quorum` queues. If only one host is configured, Hub will assume the queues are non-quorum `classic` queues by default.

Note: Queues cannot change type once created. If you wish to move from a single-host to HA, existing 'classic' queues will need to be deleted and re-created as quorum queues.

If you have a new system with only one RabbitMQ host, but plan to switch to HA in future, configure `RABBITMQ_QUEUE_TYPE=quorum`. This ensures that you will not need to delete and re-create your queues when switching to HA.

Redis Cluster

Ngenea Hub supports the use of Redis Cluster as a result provider.

Redis Cluster provides high availability (HA) and improved performance and throughput compared to single instance Redis.

To use Redis Cluster as the result provider, configure

```
RESULT_PROVIDER=external_redis_cluster
REDIS_HOST=my-pixstor-mn01
REDIS_PORT=6001
# other REDIS_* settings
```

Where `REDIS_HOST` and `REDIS_PORT` point to *any* node within the Redis Cluster. The Redis client will automatically discover the other nodes in the cluster.

The `redis_cluster_startup_nodes` configuration can be used to declare all the nodes within the Redis Cluster. This is used during Hub start up, in the event that the node configured as `REDIS_HOST` is offline or uncontactable. The Hub will identify at least one of the 'startup nodes' which is online and contactable, and use it to discover the rest of the available nodes.

Redis Cluster **cannot** be used as the broker provider.

Server Configurations

In **Ngenea Hub**, certain settings are stored in the Ngenea Hub database and can be easily viewed and modified via the **Ngenea Hub REST API** at the `/api/configurations/` endpoint.

The Ngenea Hub REST API allows you to manage configuration settings both globally and on a per-site basis. These configurations control various aspects of the system, such as job time-to-live (TTL), search backend setup, snapshot retries, task invalidation times, and more. By using simple API requests, you can optimize your system's performance and behavior.

For a comprehensive overview of available configurations, including how to view and change them through the API, as well as additional features like dynamic file batching, refer to the full documentation - Configuration.

Docker Configuration

Ngenea Hub is run via a collection of services defined in a Docker Compose file, this manages and controls all actions, settings and operations within Ngenea Hub using the `ngeneahubctl` CLI tool.

It uses the configuration defined in `/etc/sysconfig/ngeneahub` to populate the environment files

With every call using `ngeneahubctl` it will generate both the docker compose files and the required environment files each time. This is to ensure that any changes between setting changes correctly update the service files and environment variables.

Directory structure

After the first run of either `ngeneahubctl start` or `ngeneahubctl build` the following file structure is generated within `/etc/ngenea/docker`. This directory contains all needed files for use with the hub and can be edited but to ensure these edits are not overwritten by `ngeneahubctl` the setting `OVERWRITE_COMPOSE` must be set to *False*.

With overwrite disabled, the compose can be edited and executed with like any other docker compose stack.

> **Note:** Using the docker compose outside of `ngeneahubctl` is only advised if you are a system administrator that requires extra features not present in `ngeneahubctl`. Using docker configurations not generated by `ngeneahubctl` may not be supported.

```
/var/lib/pixstor/ngenea/
├── docker-compose.yml
└── environment
    ├── backend.env
    ├── celery-beats.env
```

```
    ├── celery-dags.env
    ├── celery.env
    ├── celery-events.env
    ├── celery-monitor.env
    ├── celery-refresh.env
    ├── celery-results.env
    ├── db.env
    ├── frontend.env
    ├── grafana.env
    ├── rabbitmq.env
    ├── redis.env
    ├── refresh-daemon.env
    ├── task-daemon.env
    └── websockets.env
```

All containers within the hub use an environment file per service that are populated using the sysconfig file for `ngeneahubctl`. When any operation is run through `ngeneahubctl` it will repopulate these environment files unless `OVERWRITE_COMPOSE` is disabled.

Operating the hub outside ngeneahubctl

With the docker components exposed, it can be interacted with the same as any docker compose file.

Example use case

Using the above compose, there may be requirements for the hosting system, one example could be limiting the memory consumption of the API. To limit its memory you can adjust the following section:

```yaml
backend:
  image: eurepo.arcapix.com/arcapix/ngeneahub:ngeneahub-dev
  depends_on:
    - db
  env_file: /etc/ngenea/docker/environment/backend.env
  volumes:
    - ${BITBUCKET_CLONE_DIR:-/etc}/ngenea:/etc/ngenea
  restart: on-failure
```

Could be edited into to limit the max memory to 512mb with a reserved usage of 256mb:

```yaml
backend:
  image: eurepo.arcapix.com/arcapix/ngeneahub:ngeneahub-1.0.0
  deploy:
    resources:
      limits:
        memory: 512M
      reservations:
        memory: 256M
  depends_on:
```

```
    - db
  env_file: /etc/ngenea/docker/environment/backend.env
  volumes:
    - ${BITBUCKET_CLONE_DIR:-/etc}/ngenea:/etc/ngenea
  restart: on-failure
```

After changing this container, you can interact with it using the following:

Start up the hub

```
docker compose --file /var/lib/pixstor/ngenea/docker-compose.yml u
p -d
```

Shut down the hub

```
docker compose --file /var/lib/pixstor/ngenea/docker-compose.yml d
own
```

Shell into the backend container within the hub

```
docker compose --file /var/lib/pixstor/ngenea/docker-compose.yml e
xec -it backend bash
```

Hub Authentication

Ngenea Hub provides several authentication modes for users of the Ngenea Hub user interface.

Standalone

- In this mode all users are authenticated locally by the Hub
- No external services (E.G. Microsoft Active Directory) are used for authentication
- This is the default mode of Hub when deployed standalone from PixStor, or on PixStor <= v6.10.x

Direct: LDAP/Active Directory

- In this mode authentication is provided by an external service, typically OpenLDAP or Microsoft Active Directory
- This mode can be used when Hub is deployed standalone from PixStor

To configure this authenticaion mode refer to LDAP / Active Directory Login

Single Sign On

- Single Sign On mode cannot be used when Hub is deployed standalone from PixStor

- New installations of PixStor >= v6.11.x are deployed in Single Sign On mode by default
- Upgrades from PixStor versions <= v6.10.x retain their prior authentication mode

Important: In order to utilise additional products, such as Iris, Hub's Single Sign On service on is required

Hub Only

- In this mode users created in Hub are added to Hub's Single Sign On provision
- Users added to the Single Sign On provision are readily available for future use across PixStor products (E.G. Iris)
- No external services (E.G. Microsoft Active Directory) are used for authentication

Active Directory

- In this mode Microsoft Active Directory is used to authenticate user's logins to Hub using their Active Directory credentials
- In case of loss of connection to Microsoft Active Directory, user accounts created prior to SSO enablement can continue to authenticate to Hub using their Hub managed credentials

To configure Single Sign on authentication refer to Single Sign-On / External IdP

Worker Configuration

Note: In the installation guide we have explained how to configure the Ngenea Worker, create or edit the `/etc/ngenea/ngenea-worker.conf` file with the required settings for your site, API server, Redis, and GPFS nodes. You can also configure SSL for secure communication with the API.

For more details, please refer to the link.

Settings

The following is a list of available settings:

Option	Type	Default	Required	Description
site	*string*		Yes	The name of the queue to listen to
api_host	*string*		Yes	The base url for Ngenea Hub, this will be the url without the https protocol or port.

Option	Type	Default	Required	Description
api_port	*string*		Yes	The port that the Ngenea Hub is being hosted on, by default within Ngenea Hub it is 8000.
api_secure	*string*		No	If the API is behind a secure HTTPS connection, by default this is true. Refer to: Installing SSL Certificates for provisioning SSL certificates if `api_secure=true`.
api_secure_verify	*string*		No	If the API requests should verify certificates, by default this is true.
broker_provider	*string*	redis	No	The service that will provide worker connectivity with the hub, must be one of the following: *redis*| *rabbitmq*| *external_rabbitmq*| *external_redis*
result_provider	*string*	redis	No	The result storage that will store completed worker tasks for the hub to access, must be one of the following: *redis*| *external_redis*| *external_redis_cluster*
threads	*int*	*10*	No	The number of concurrent tasks that can be run.
heartbeat	*bool*	*true*	No	Key for Disabling/ Enabling celery heartbeats, by default Enabled.
gossip	*bool*	*false*	No	Key for Disabling/ Enabling celery gossip, by default Disabled.

Option	Type	Default	Required	Description
mingle	*bool*	*false*	No	Key for Disabling/ Enabling celery mingle, by default Disabled.
redis_health_check_interval	*int*	*60*	No	The Redis backend supports health checks. This value must be set as an integer whose value is the number of seconds between health checks.
redis_socket_timeout	*int*	*60*	No	The Redis results backend supports a socket connection timeout, this value must be set as an integer whose value is the number of seconds.
redis_host	*string*		No	The host for the Redis service. - For **Internal Redis**: if provided, this host will be used instead of *api_host*. - For **External Redis**: required; must be a valid IP or hostname.
redis_port	*string*	*6379*	No	The host port for a Redis service that is not within the hub's services, this value must be a string of a valid port.
redis_password	*string*		No	The password for a Redis service that is not within the hub's services, this value must be a string.
redis_tls	*bool*	*false*	No	Enables TLS for external Redis service configurations. This does not disable TLS when not configured as external, disabled by default.

Option	Type	Default	Required	Description
redis_tls_ca	*string*		No	The absolute path to the root certificate for an external Redis service, this value must be a string.
redis_tls_cert	*string*		No	The absolute path to the client certificate for an external Redis service, this value must be a string.
redis_tls_key	*string*		No	The absolute path to the certificate key for an external Redis service, this value must be a string.
redis_cluster_hosts_json	*string*		No	The absolute path to a JSON file which maps Redis Cluster nodes' internal addresses to external addresses. For use when Redis Cluster is configured as the result provider.
rabbitmq_host	*string*		No	The host for a RabbitMQ service that is not within the hub's services, this value must be a valid IP or hostname.
rabbitmq_port	*string*	*5671*	No	The port for a RabbitMQ service both external and within the hub's services, this value must be string of a valid port.
rabbitmq_vhost	*string*	*nghub*	No	The virtual host for a RabbitMQ service that both external and within the hub's services, this value must a string.

Option	Type	Default	Required	Description
rabbitmq_user	*string*	*ngeneahub*	No	The username for a RabbitMQ service that both external and within the hub's services, this value must a string.
rabbitmq_password	*string*	*ngeneahub*	No	The password for a RabbitMQ service that both external and within the hub's services, this value must a string.
rabbitmq_tls	*bool*	*false*	No	Enables TLS for external RabbitMQ service configurations. This does not disable TLS when not configured as external, disabled by default.
rabbitmq_tls_ca	*string*		No	The absolute path to the root certificate for an external RabbitMQ service, this value must be a string.
rabbitmq_tls_cert	*string*		No	The absolute path to the client certificate for an external RabbitMQ service, this value must be a string.
rabbitmq_tls_key	*string*		No	The absolute path to the certificate key for an external RabbitMQ service, this value must be a string.
rabbitmq_queue_type	*string*	*classic*	No	Queue type to use when creating queues in RabbitMQ, when there is only one *rabbitmq_host*. Either *classic* or *quorum*. Configure as *quorum* if you plan to scale out RabbitMQ.

Option	Type	Default	Required	Description
celery_socket_timeout	*int*	*60*	No	The Redis celery broker supports a socket timeout, this value must be set as an integer whose value is the number of seconds.
celery_connection_timeout	*int*	*60*	No	The Redis celery broker supports a socket connection timeout, this value must be set as an integer whose value is the number of seconds.
gpfs_nodes	*list*		No	A list of nodes to run the snapdiff policy scan on as a list of hostnames
enable_plugins	*bool*	*false*	No	Key for Disabling/ Enabling support for custom worker plugins, by default Disabled.
loglevel	*str*	*INFO*	No	Key for setting the worker loglevel (more specific to worker main process). valid choices are INFO, DEBUG, WARNING, CRITICAL, ERROR (by default INFO).

Controlling Functions

Ngenea Worker uses internal function queues to process tasks. These queues can be adjusted based on your needs.

What Are Function Queues ?

Function queues are like task managers that organize and run specific tasks. You can turn these on or off and adjust their settings.

How to Configure Function Queues ?

- Open the configuration file (`/etc/ngenea/ngenea-worker.conf.`) for Ngenea Worker.
- Modify settings to enable or disable queues.

- Refer to the Workflow Steps documentation to see what each queue does.

Creating Custom Queues

If the default queues don't meet your needs, you can create custom job queues. These allow you to prioritize specific tasks or allocate more resources.

Example Configuration for a Custom Queue

Steps to Add a Custom Queue

1. Open the worker's configuration file (`/etc/ngenea/ngenea-worker.conf.`).
2. Add a new section with the name of the queue in brackets, like `[Queue highpriority]`.
3. Specify the number of threads for that queue. (Optional)
4. Save and reload the worker service to apply changes.

For detailed information, check Queues Documentation.

Using a Custom Configuration File

Configuration files are text files used to set options and preferences for a system or application. Sometimes, you may need to override the default configuration with a custom one. Here's how to set it up:

By default, the system uses the file located at:

```
/etc/ngenea/ngenea_config_arg.conf
```

How to Use a Custom Config File ?

- **Uncomment the CONFIG Line:** Open the file `/etc/ngenea/ngenea_config_arg.conf` and remove the # symbol in front of this line:

```
CONFIG=/etc/ngenea/ngenea-worker-custom.conf
```

- **Create a Custom Worker Config File:** Save a new file with a name like ngenea-worker-custom.conf. Use this file to define your settings.
- **Register Your Site:** Run these commands to register the custom site:

```
ngeneahubctl addsite <sitename>
ngenea-worker join --user <username> --site <sitename>
```

RabbitMQ HA

Ngenea Hub supports the use of High Availability (HA) RabbitMQ, using Quorum Queues, as the broker provider.

RabbitMQ HA replicates queues to additional hosts to support failover in the event that one or more hosts go down.

To use RabbitMQ HA as the broker provider, configure

```
broker_provider=external_rabbitmq
rabbitmq_host=my-pixstor-mn01,my-pixstor-ng01,my-pixstor-ng02
rabbitmq_port=5671
# other rabbitmq_* settings
```

Configure all the hosts in the RabbitMQ cluster as a comma-separated list in `rabbitmq_host`. The same `rabbitmq_port` will used for all hosts by default. If each host uses a different port, the ports can be included in `rabbitmq_host`, for example `host1:1234,host2:5678,...`

If multiple hosts are configured, Hub will assume that queues should be `quorum` queues. If only one host is configured, Hub will assume the queues are non-quorum `classic` queues by default.

Note: Queues cannot change type once created. If you wish to move from a single-host to HA, existing 'classic' queues will need to be deleted and re-created as quorum queues.

If you have a new system with only one RabbitMQ host, but plan to switch to HA in future, configure `rabbitmq_queue_type=quorum`. This ensures that you will not need to delete and re-create your queues when switching to HA.

Redis Cluster

Ngenea Hub supports the use of Redis Cluster as a result provider.

Redis Cluster provides high availability (HA) and improved performance and throughput compared to single instance Redis.

When Redis Cluster is used as the result provider, configure

```
result_provider = external_redis_cluster
redis_host = my-pixstor-mn01
redis_port = 6001
# other redis settings
```

Where `redis_host` and `redis_port` point to *any* node within the Redis Cluster. The Redis client will automatically discover the other nodes in the cluster.

Redis Cluster Host Mapping

In a Pixstor environment, the nodes within a Redis Cluster will communicate with each other on the 'backplane' (internal) network. However, sites which aren't within the same cluster network will not be able to access Redis Cluster via its internal addresses.

Pixstor provides external access to Redis Cluster via proxying. However, the Redis client needs to be told how to map its nodes' internal addresses to the corresponding external proxy addresses.

The host mapping is a JSON formatted file, which defines the mapping of each node's internal address to its external address

```
{
    "10.100.0.1:6000": "my-pixstor-mn01:6010",
    "10.100.0.2:6001": "my-pixstor-mn02:6011",
    "10.100.0.3:6002": "my-pixstor-sn01:6012"
}
```

The worker is then configured using `redis_cluster_hosts_json`, to tell the worker where the JSON file is located.

```
redis_cluster_hosts_json = /etc/ngenea/redis_cluster_hosts.json
```

When Redis Cluster can be reached on its internal addresses, the `redis_cluster_hosts_json` configuration can be omitted. Alternatively, the addresses can be mapped to their selves, or to `null` to indicate that no mapping is necessary.

The host mapping is also used to define 'startup nodes'. Start up nodes are used as fallbacks, in the event that the configured `redis_host` is offline or not contactable when the worker starts up.

Running Ngenea Worker with Multiple Sites

Note: Running multiple sites on a node has several drawbacks, so using custom queues is preferred where possible.

To run services with two sites:

- Create new service file same as `/usr/lib/systemd/system/ngenea-worker.service` with different filename (Eg: ngenea-worker-site1.service).
- Create new custom config file same as `/etc/ngenea/ngenea_config_arg.conf` with different filename (Eg: ngenea_config_arg1.conf).
- Create new worker config file same as `/etc/ngenea/ngenea-worker.conf` with different filename and give the filename in ngenea_config_arg1.conf.
- In `ngenea-worker-site1.service`, change the value of `EnvironmentFile=...` to new custom config file created (Eg: EnvironmentFile=/etc/ngenea/ngenea_config_arg1.conf).
- Now run systemctl start `ngenea-worker.service` and `systemctl start ngenea-worker-site1.service`.

Setting Ngenea Worker Debug Level in systemd Service Script

Environment Directive: In a system, there are background tasks that run automatically (called services). The **`Environment` directive** in the service's setup tells it about certain settings it should use while running.

What Does `DYNAMO_DEBUG=true` Do?

`Environment=DYNAMO_DEBUG=true` sets a setting called `**DYNAMO_DEBUG**` to `**true**` for the service called **ngenea-worker**.

What Happens with This Setting?

- Normally, the **ngenea-worker** service only logs basic information about what it's doing (like regular updates).
- When you set `**DYNAMO_DEBUG=true**`, the service will log **extra, detailed information** about its activities.

Hub Messaging Queue Configuration

When setting up communication between various components of the Ngenea Hub, such as workers and the hub itself, a messaging queue plays a crucial role. These messaging queues act as intermediaries, efficiently managing and delivering messages between components.

The Ngenea Hub supports two messaging queue systems: **Redis** and **RabbitMQ**. Below is a step-by-step guide to understanding and configuring these systems.

Choosing the Messaging Queue

The Ngenea Hub provides two primary options for a messaging queue:

- **Redis**: Known for its speed and simplicity, Redis is an excellent choice for systems requiring high performance with straightforward configuration.
- **RabbitMQ** : RabbitMQ offers advanced features compared to Redis, including more sophisticated tools for managing queues, making it a better choice for complex messaging requirements.

The messaging between workers and Ngenea Hub is configurable between these two supported services. The broker can be configured to use either Hub's own internal services, or externally hosted services.

Broker selection is controlled with variables in `/etc/sysconfig/ngeneahub` along with additional configuration files.

The following are all broker related configuration values:

Setting	Default	Description
BROKER_PROVIDER	redis	This controls the messagi `redis|rabbitmq|external_redis|external_`
RABBITMQ_USER	ngeneahub	Username of the rabbitmq user for use with
RABBITMQ_PASSWORD	Randomly generated password	Password used for the rabbitmq user for u Hub
RABBITMQ_VHOST	nghub	The virtual host used within the rabbitmq Ngenea Hub

Setting	Default	Description
RABBITMQ_HOST	rabbitmq	The address of a RabbitMQ service. Can be hub services. Can be a comma-separated the Hub-internal container service address
RABBITMQ_PORT	5671	The port of the RabbitMQ broker service.
RABBITMQ_TLS	False	Enables TLS support for external RabbitMQ
RABBITMQ_QUEUE_TYPE	classic	Queue type to use when creating queue when there is only one *RABBITMQ_HOST*. Eit *uorum*. Configure as *quorum* if you plan RabbitMQ.
REDIS_HOST	redis	Address of a Redis server. Defaults to th container service address.
REDIS_PORT	6379	Port of the Redis server.
REDIS_PASSWORD	Not set	The password for an external Redis broker this setting is exclusively for external service
REDIS_TLS	False	Enables TLS support for external Redis servi
REDIS_HEALTH_CHECK_INTERVAL	60	The Redis backend supports health checks. be set as an integer whose value is the num between health checks.
REDIS_TCP_BACKLOG	511	In high requests-per-second environments y backlog in order to avoid slow client conne redis.
REDIS_SOCKET_TIMEOUT	60	When there are network issues redis back sockets can become stale, this timeout se the socket connection after this value in becoming idle and resume operation.

For more details on general configuration see Hub Configuration for more details.

Redis Setting Files

When setting up Redis within Ngenea Hub, you're provided with a default configuration file located at: `/etc/ngenea/redis/redis.conf`, any edits made to this file will persist with software upgrades.

This file is pre-configured to help get you started with Redis, but it can be customized to better meet your specific needs. Common adjustments include tuning memory usage, adjusting data persistence settings, or optimizing performance based on the expected workload.

Customizing Redis

To make changes to the Redis configuration, follow these steps:

- **Open the Configuration File:** Use a text editor to open the redis.conf file: `sudo nano /etc/ngenea/redis/redis.conf`
- **Edit the File:** Make the necessary changes to the configuration. The file allows you to adjust various Redis settings, including memory limits, persistence

options, and more. For a full list of configurable options, you can refer to Redis configuration.

- **Save the Changes:** After making the required changes, save the file and close the text editor.

Restarting Redis to Apply Changes

For your changes to take effect, Redis must be restarted. There are two ways to restart the Redis service:

Preferred Method: Restarting Ngenea Hub

The easiest and preferred method is to restart the entire Ngenea Hub system. This can be done with the following command:

```
sudo systemctl restart ngeneahub
```

This will restart all services, including Redis, and apply your changes without any risk of inadvertently disrupting the Redis container.

Alternative: Restarting Only the Redis Container

If you prefer to restart just the Redis container without affecting other services, you can restart it using Docker. **However, this method may have unintended consequences**, as it only restarts Redis and not the other dependent services in Ngenea Hub.

To restart the Redis container, use the following command:

```
docker restart ngeneahub-redis-1
```

While this can be useful for testing small changes, it's generally safer to restart the entire system to avoid potential issues with container dependencies.

RabbitMQ Setting Files

If Ngenea Hub is configured to use RabbitMQ, additional configuration files can be provided within the directory: `/etc/ngenea/rabbitmq/conf.d`

This directory can be populated with custom configuration files for broader RabbitMQ configuration.

Default Configuration Files: The provided default configuration files within the `/etc/ngenea/rabbitmq/conf.d` directory are defined and updated in: `/usr/share/ngeneahub/rabbitmq`.

These default configuration files will be replaced during software upgrades. Therefore, it is important to apply any changes within custom configuration files to preserve your customizations.

Customizing RabbitMQ: RabbitMQ allows extensive customization, such as setting limits on the number of messages processed concurrently.

Note: Be mindful that some default configuration files may be overwritten during software updates. To avoid losing customizations, save your configuration changes in separate files within the `/etc/ngenea/rabbitmq/conf.d` directory.

Applying Changes: After modifying the RabbitMQ configuration files, you must restart the RabbitMQ service to apply the changes. This can be done with the following command: `sudo systemctl restart rabbitmq-server`.

Using RabbitMQ's Admin Interface

Unlike Redis, RabbitMQ does not come with built-in command line tools to monitor or manage its operations directly.

However, a tool known as the "admin panel" is available for RabbitMQ. By default, it is not turned on, but it can be enabled by running the following command in the Ngenea Hub host's command line:

```
ngeneahubctl exec -c rabbitmq rabbitmq-plugins enable rabbitmq_ma
nagement
```

Once it is turned on, access can be gained by visiting the address (**port 15672**) on the Ngenea Hub.

```
http://<hub-host>:15672
```

To access this, a login is required using a username and password. These login details are stored in a specific file on the server, called `/etc/sysconfig/ngeneahub`, where two important pieces of information can be found:

- `RABBITMQ_USER` : The username to log in.
- `RABBITMQ_PASSWORD` : The password for the username.

Limitations

When using RabbitMQ, it's important to note that its data—such as queue metrics, worker statuses, and throughput—will not appear in the Ngenea Hub's built-in monitoring tool, Grafana, which is accessible via:

`/hubmetrics`

Instead, you must rely on the RabbitMQ admin panel to view and manage such metrics.

Iris Configuration

This page describes how to configure Ngenea Hub to integrate with the Iris product.

Iris Tasks and RabbitMQ

Ngenea Hub provides a set of tasks — also known as 'publishers' — for generating and extracting file metadata. These tasks integrate with Iris by publishing messages to the Iris RabbitMQ.

The configurations for these Iris tasks to connect with RabbitMQ are defined in `/etc/ngenea/iris-rabbitmq.yaml`.

Important: In a Pixstor environment, when Iris is enabled, this file is pre-configured and should not be edited.

The config file is divided into several sections, described below.

RabbitMQ

The `rabbitmq` section defines general connection settings for RabbitMQ. These settings apply to all publishers.

- **host** — Hostname or IP address of the RabbitMQ instance to connect to.
- **port** — Port of the RabbitMQ instance, typically `5671`.
- **username** — Username to authenticate the connection to RabbitMQ (default: `guest`).
- **password** — Password for the `username` used to connect to RabbitMQ (default: `guest`).
- **use_ssl** — Whether to connect to RabbitMQ using SSL (default: `false`).
- **ca_cert_path** — Path to the certificate authority file for SSL connections.
- **client_cert_path** — Path to the client certificate file, signed by the configured certificate authority.
- **client_key_path** — Path to the client key file matching the configured client certificate.
- **virtual_host** — Virtual host to use when connecting to RabbitMQ (default: `/`).
- **max_retries** — Number of connection attempts if a connection cannot be achieved (default: `3`).
- **retry_interval** — Time to wait between connection attempts, in seconds (default: `5`).
- **log_level** — Controls the level of RabbitMQ connection logging in Ngenea Worker logs. Values range from `0` (least verbose; default) to `2` (most verbose).

Warning: `password` is stored in plaintext. Ensure permissions on `iris-rabbitmq.yaml` prevent unwanted access.

Example `rabbitmq` section:

```
rabbitmq:
    host: localhost
    port: 5671

    username: guest
    password: guest
```

```
    use_ssl: true
    ca_cert_path: /path/to/rabbitmq/tls/ca.crt
    client_cert_path: /path/to/rabbitmq/tls/tls.crt
    client_key_path: /path/to/rabbitmq/tls/tls.key

    virtual_host: /

    max_retries: 3
    retry_interval: 5

    log_level: 0
```

Publishers

The `publishers` section defines which RabbitMQ exchange the 'publishers' — Iris tasks — should publish their messages to.

There is one subsection per publisher, and each subsection defines the following:

- **exchange_name** — Name of the RabbitMQ exchange to publish messages to.

Currently, there are three publishers:

- **filenotify** — Generates file system change events, such as when a file is created or a directory is moved.
- **mediainfo** — Extracts media file metadata using the `mediainfo` tool.
- **exiftool** — Extracts file metadata using the `exiftool` tool.

Example `publishers` section:

```
publishers:
    filenotify:
        exchange_name: fs-events-xchg
    mediainfo:
        exchange_name: metadata-xchg
    exiftool:
        exchange_name: metadata-xchg
```

Messages

The `messages` section defines per-publisher metadata.

There is one subsection per publisher, and each subsection defines the following key:

- **endpoint_url** — URL of the Versity endpoint which exposes files processed by this publisher.

Typically, the `endpoint_url` has the form `https://VERSITY_GATEWAY:7070`. It will normally be the same for all publishers.

The existing publishers are listed in the **Publishers** section above.

Example `messages` section:

```
messages:
    filenotify:
        endpoint_url: https://10.100.0.100:7070
    mediainfo:
        endpoint_url: https://10.100.0.100:7070
    exiftool:
        endpoint_url: https://10.100.0.100:7070
```

Error Handling

The `error_handling` section defines how tasks should handle errors when processing files.

There is one subsection per task (publisher), and each subsection defines the following key:

- **on_error_set_aborted** — Whether to mark a file as `aborted` (true) or `failure` (false) when the file cannot be processed.

When files are marked as `failed`, the whole job is also marked as failed. When files are marked as `aborted`, the job will be marked as successful (unless other, unrelated failures occur in the job).

The existing publishers are listed in the **Publishers** section above.

Additionally, `on_error_set_aborted` can be set at the top level of the `error_handling` section to apply the same setting to all tasks.

Example `error_handling` section:

```
error_handling:
    on_error_set_aborted: true

    exiftool:
        on_error_set_aborted: true
    mediainfo:
        on_error_set_aborted: false
```

In the above example, the `exiftool` task (and any others not explicitly included) will inherit the top-level `on_error_set_aborted: true`.

Iris Workflow Queue

When a space is enabled for Iris in Ngenea Hub, a scheduled workflow is created which runs the Iris tasks mentioned in the previous section.

This scheduled workflow is configured to run on the dedicated `iris_notify` queue. This means that the notify workflow will not block or be blocked by any other tasks running at the same time in Ngenea Hub, such as file sync.

The Iris queue **must** be defined in the Ngenea Worker configuration file, as described in Queues.

For example:

```
[Queue iris_notify]
threads = 10
```

If the queue is not defined in the worker config or is removed, the Iris tasks will not execute.

> **Note:** In a Pixstor deployment, this queue is defined automatically. Configuration changes, such as altering the number of threads, must be done via Pixstor

Feature Set-up

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

Search

The search feature in Ngenea Hub allows you to easily find files and folders across different sites. Setting it up correctly will ensure that you can use this powerful tool to quickly locate files when needed.

Requirements

The search feature works with systems running on PixStor, which is the platform that powers Ngenea Hub. There are two search options available: **PixStor Search** and **PixStor Analytics**.

This guide assumes that one of these options has already been set up as part of your PixStor installation.

Configuration Steps

There are two main configurations you need to consider to set up search properly.

Choose Your Search Backend

Depending on the version of Ngenea Hub you're using, the system may utilize either the Analytics Backend or the PixStor Search Backend for handling search functionality. Both backends provide the same core features, but you might need to switch between them based on your system's specific configuration.

If you want to switch from the default Analytics Backend to the PixStor Search backend, it's simple to do, and no coding is required. You can change the setting by sending a request through a command. Here's how:

Option 1: Using curl

`curl` is a tool that allows you to interact with online services from the command line. To switch to the PixStor Search backend, use the following command:

```
curl -s -X PATCH 'http://example.com/api/configurations/' -H 'Acc
ept: application/json' -H "Authorization: Bearer
$JWT_ACCESS_TOKEN" -H 'Content-Type: application/json' -d '{"sear
ch_backend": "pixstor_search"}'
```

- Replace http://example.com with your system's actual address.
- $JWT_ACCESS_TOKEN should be replaced with your valid access token, which is required for authentication.

This command will update the configuration of your system to use PixStor Search instead of the default Analytics Backend.

Option 2: Using ngcurl

If you have `ngcurl` installed and configured on your system, you can use this simpler command:

```
ngcurl patch configurations '{"search_backend": "pixstor_search"}'
```

This will also switch the search backend to PixStor Search. Both options achieve the same result; you can choose the one that fits your setup.

Note: To understand in detail please refer to the Global Configuration documentation.

Set the Elasticsearch URL

If you're using the Analytics backend and your system is running on PixStor 6, you'll need to adjust the Elasticsearch URL to ensure the system can find the right data. The new address will be `http://localhost:19200`.

If you're using PixStor 5, the default address `localhost:9200` will work fine, and you don't need to change anything.

Here's an example of how to update this setting for PixStor 6:

```
$ 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"}'
```

Note: To understand in detail please refer to the Site-specific Configurations documentation.

Using the Search Feature

Once everything is set up, you can begin using the search feature in Ngenea Hub.

Search via REST API:

The search functionality can be accessed through the Ngenea Hub's REST API. This is a great option if you're comfortable with using API tools. For details on how to use search via the REST API, refer to Search

Search via the Ngenea Hub User Interface (UI):

The Ngenea Hub interface provides a user-friendly way to search for files across sites directly from the global search page in UI.

Cloud Functions

Introduction

"Cloud to Hub" is a function that works in the cloud and reacts to changes in cloud storage, like when files are added or deleted. When something changes, it sends an update to Ngenea Hub, which then syncs the changes with other systems (like PixStor). This ensures that all the systems stay up-to-date and match the information in cloud storage.

It works with any supported platform and can handle any type of event (like creating or deleting files). To set it up for a new platform or event, you only need to change the settings in the `config.json` file.

Note: A function is a specific task or action that a system or program can perform. In this case, the Cloud to Hub function reacts to changes in cloud storage and triggers actions to keep other systems in sync.

Cloud Functions supports **Python 3.12 and newer**.

Recommended Version

We recommend using:

- Python 3.12+
- Latest stable patch release where possible

Why Use Cloud Functions ?

Automation: Cloud Functions ensure that once something happens in the cloud (like a file upload), the correct action is taken automatically, without manual intervention.

Synchronization: Cloud Functions help keep different systems in sync with each other, making sure all systems know when a file has changed.

Scalability: The cloud-based nature of the function means it can handle large amounts of data and automatically scale as needed.

Supported Platforms:

Google Cloud Platform (GCP) is a set of cloud tools by Google. It helps you run apps, store data, and automate tasks without using your own servers. One of these tools is Cloud Functions. A Cloud Function runs code when something happens—like when a file is uploaded to storage.

In this section of the guide, you'll create a function that runs when a file is added or changed in a **Cloud Storage bucket**. This is useful for things like data processing or connecting to other systems.

Before you start, here are some important things to know:

- A **Cloud Storage bucket** is a place in the cloud to store your files.
- The function will **watch the bucket** and run automatically when a file is added or changed.
- A **service account** is needed so the function can access the bucket.
- The service account needs the `storage.objectViewer` role to read files from the bucket.
- The function will use **Python 3.12** as its programming language.
- Your code will be uploaded to GCP as a **ZIP file**.
- You'll edit a file called `config.json` to set specific settings for how the function works.
- If the function needs to talk to a private system (with no public IP), you may need a **VPC Connector**.

This guide will show you each step in order. You'll learn how to create the function, set the right permissions, choose the correct settings, upload your code, and deploy it. When you're done, your function will automatically run whenever a file is added or changed in your storage bucket. This introduction helps you understand the basics so you can follow the setup process more easily.

Step 1: Unpack the GCP Cloud Function Zip

```
mkdir ngenea_hub_gcp_cloud_function
cd ngenea_hub_gcp_cloud_function
unzip -x cloud_bucket_event_gcp-<version>.zip
```

Step 2: Modify the config.json

Required known settings:

- IP address of Hub or Cloud Load Balancer in front of Hub
- API key defined in a user account which will be responsible for adding Jobs to Hub
- The workflow being used when the Cloud Function is triggered
- The workflow parameters (flags) to apply
- The site(s) on which to create a job of the workflow type for each file triggering the Cloud Function
- Review the remaining settings accordingly

Warning: Resource management is the responsibility of the end user. Ensure to obtain sufficient understanding from the cloud vendor documentation and Hub workloads such that costs are predictable.

Example:

```
{
    "version": 1.0,
    "hub_access": {
        "hub_ip": "10.10.10.10",
        "hub_port": 443,
        "hub_protocol": "https",
        "api_key": "ABC123456",
        "workflow": "reverse_stub",
        "workflow_flags": {
            "hydrate": false,
            "overwrite": true
        }
    },
    "sites": [
        {
            "site": "site1",
            "default": "stub",
            "skip_from_ngenea": true
        }
    ],
...
```

Step 3: Set Up Service Account for Access

Cloud Functions often need permissions to interact with other Google Cloud services, like Cloud Storage. We'll create a service account to grant these permissions.

What is a Service Account? A service account is like a virtual identity that a program or a service can use to interact with Google Cloud resources. It has specific permissions assigned to it.

- To create a service account using the `gcloud` command-line tool, run the following commands. Replace `GCP-PROJECT-1` with your actual **project ID** and **ngeneahub-function** with the name of your service account.
- **Assigning the `storage.objectViewer` role:** This grants the service account the ability to view objects (files) in your Cloud Storage bucket.

```
PROJECT_ID='GCP-PROJECT-1'
SERVICE_ACCOUNT_ID='ngeneahub-function'
ROLE_NAME='roles/storage.objectViewer'

gcloud iam service-accounts create $SERVICE_ACCOUNT_ID \
    --description='A service account to give the {{ brand_name }}
function read access to GCS buckets' \
    --display-name=$SERVICE_ACCOUNT_ID
```

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member="serviceAccount:$SERVICE_ACCOUNT_ID@$PROJECT_ID.iam.
gserviceaccount.com" \
    --role=$ROLE_NAME
```

Step 4: VPC Connector (Optional) – If your function needs to access a private network (such as a private IP address or a database), you'll need to set up a VPC Connector. However, if your function doesn't require a private connection, you can skip this part.

What is a VPC Connector? A VPC (Virtual Private Cloud) Connector enables your Cloud Function to access resources located in your Google Cloud private network, or other routed networks. This is necessary if your function needs to communicate with services that do not have a public IP address, such as a private database or internal system like the Ngenea Hub.

When is it Needed? If the Ngenea Hub does not have an external (public) IP address, you will need to use a VPC Connector to allow the function to connect to it via its private IP.

Note: Creating a new VPC Connector is outside the scope of this documentation. Ensure that a connector is already set up if your function requires access to private network resources.

Step 5: Deploy the Cloud Function to GCP

From the directory above `ngenea_hub_gcp_cloud_function` issue a gcloud command similar to the following example and appropriate to your GCP environment.

Example:

```
gcloud functions deploy ngeneahub-function \
--no-gen2 \
--runtime python312 \
--no-allow-unauthenticated \
--region=europe-west1 \
--trigger-location=europe-west1 \
--trigger-bucket=myproject-space01 \
--service-account=myserviceuser-account \
--ingress-settings=internal-only \
--entry-point=lambda_handler \
--min-instances=0 \
--max-instances=10 \
--memory=128MB \
--cpu=0.5 \
--timeout=300s \
--source ./ngenea_hub_gcp_cloud_function \
--egress-settings=all \
--vpc-connector=myvpcconnector
```

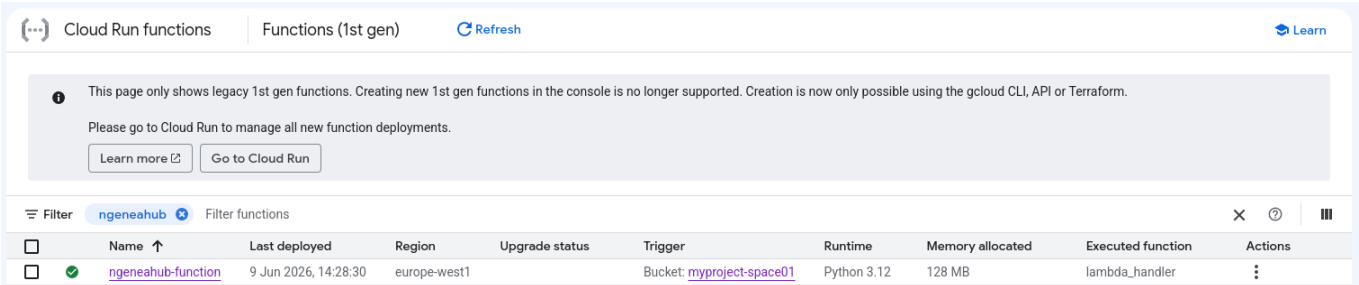
Sample result:

```
Deploying function (may take a while - up to 2
minutes)...⠧

For Cloud Build Logs, visit: https://console.cloud.google.com/
cloud-build/builds;region=europe-west1/....
Deploying function (may take a while - up to 2
minutes)...done.

automaticUpdatePolicy: {}
availableMemoryMb: 128
buildId: 12345678-1d73-4204-bbaf-376bb1715d5a
buildName: projects/my-gcp-project/locations/europe-west1/
builds/....
dockerRegistry: ARTIFACT_REGISTRY
entryPoint: lambda_handler
eventTrigger:
  eventType: google.storage.object.finalize
  failurePolicy: {}
  resource: projects/_/buckets/myproject-space01
  service: storage.googleapis.com
ingressSettings: ALLOW_INTERNAL_ONLY
labels:
  deployment-tool: cli-gcloud
maxInstances: 10
name: projects/my-gcp-project/locations/europe-west1/functions/
ngeneahub-function
runtime: python312
satisfiesPzi: true
serviceAccountEmail: my-service-user
sourceUploadUrl: https://storage.googleapis.com/uploads-....zip
status: ACTIVE
timeout: 300s
updateTime: '2026-06-09T13:28:30.801312358Z'
versionId: '1'
vpcConnector: projects/my-gcp-project/locations/europe-west1/
connectors/myvpcconnector
vpcConnectorEgressSettings: ALL_TRAFFIC
```

The deployed Cloud Function (1st Gen) is now live and operational.

Step 6: Navigate to Cloud Functions (1st Gen) in Google Cloud Console

- Open the Google Cloud Console to view the Cloud Functions (1st Gen).



Review the settings of the deployed Cloud Function.

Additional changes can now be made via the UI, or deletion and redeployment.

AWS Lambda Function

What's AWS Lambda?

AWS Lambda is a service provided by Amazon Web Services (AWS) that allows you to run code without needing to manage servers. Imagine it as a machine that runs your code only when something specific happens, like a file being uploaded to an online storage system (like S3, which is AWS's storage service). You just upload your code, and AWS Lambda automatically handles the running of that code when it's needed.

Warning: Resource management is the responsibility of the end user. Ensure to obtain sufficient understanding from the cloud vendor documentation and Hub workloads such that costs are predictable.

What's IAM (Identity and Access Management)?

IAM is a service in AWS that helps you manage access to AWS resources. It allows you to define who can access what and what they can do with those resources. Think of IAM as a security guard that checks if someone has permission to use a certain resource or perform a certain action.

Create an IAM Policy and Role

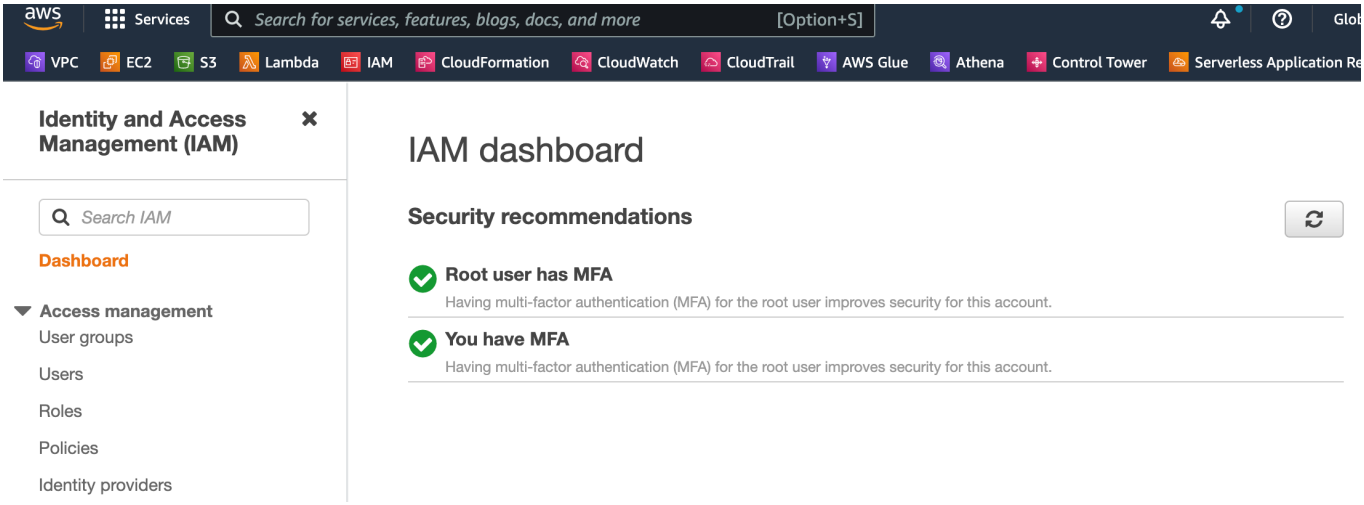
IAM Policy: This is a set of permissions that define what a service (like Lambda) can or cannot do. For example, in this guide, the policy gives Lambda permission to:

- Create log groups (where logs are stored).
- Write logs into CloudWatch (a service that monitors AWS resources).
- Get information about IAM users (a part of the AWS permissions system).

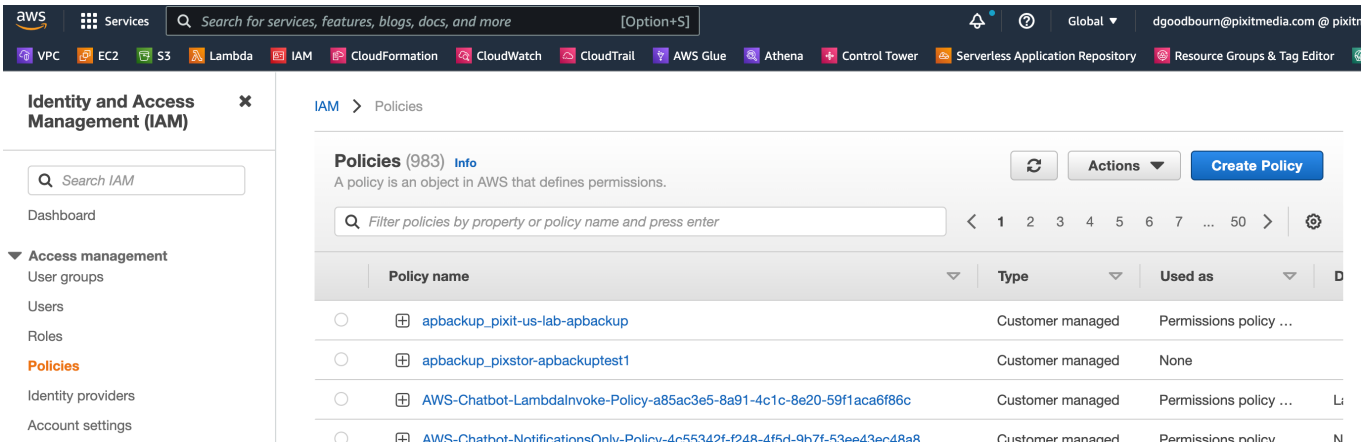
IAM Role: A role is like a `job` given to a service, like AWS Lambda, with specific permissions attached. The role lets Lambda use the permissions defined in the IAM policy you just created. It's like assigning a job to an employee (Lambda) and giving them a set of instructions (the policy) for what they're allowed to do.

To set up an IAM policy and role for AWS Lambda, follow these steps in the AWS console:

1. **Navigate to the IAM Service**: In the AWS Management Console, go to the **I AM** service. You can search for "IAM" in the search bar if it's hard to find.



1. **Create a New Policy:** In the left-hand menu, choose Policies and click the **Cr eate policy** button.



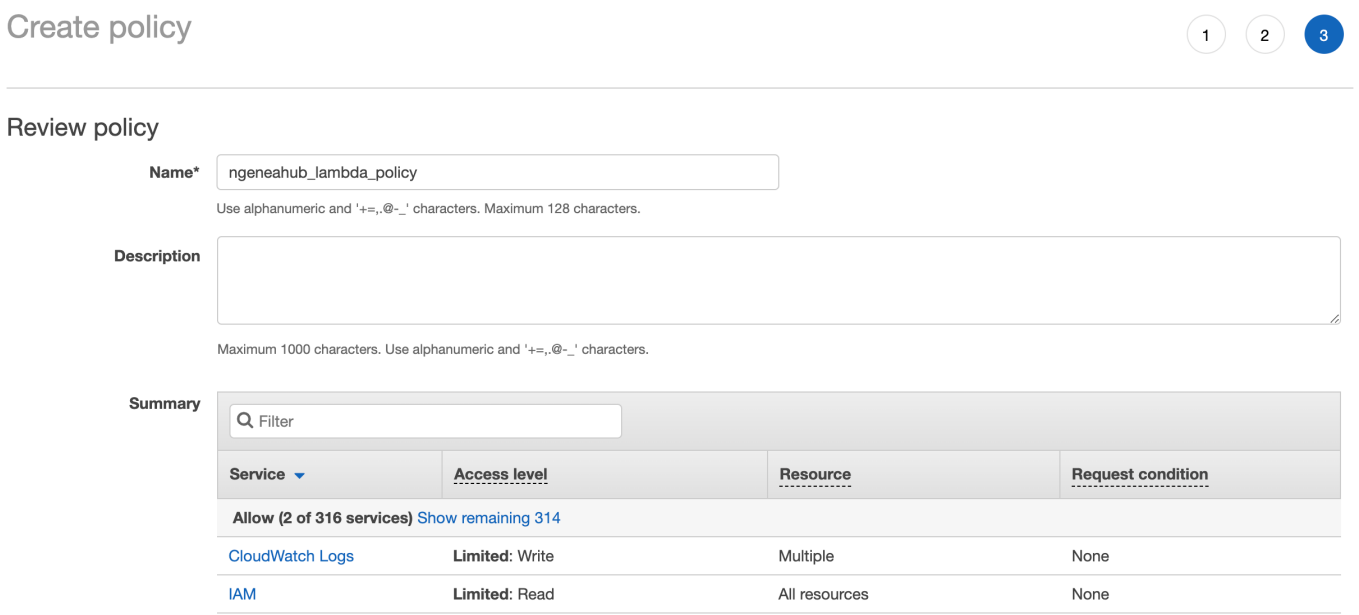
1. **Switch to the JSON Tab**: After clicking **Create policy**, you'll be taken to a page where you can define the policy. Select the JSON tab.
2. **Replace the Default Policy JSON**: Replace the existing text in the **JSON** editor with the following policy document:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:*:<<AWS_ACCOUNT_ID>>:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:<<AWS_ACCOUNT_ID>>:log-group:/aws/
lambda/<<LAMBDA_NAME>>:*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": "iam:GetUser",
            "Resource": "*"
        }
    ]
}
```

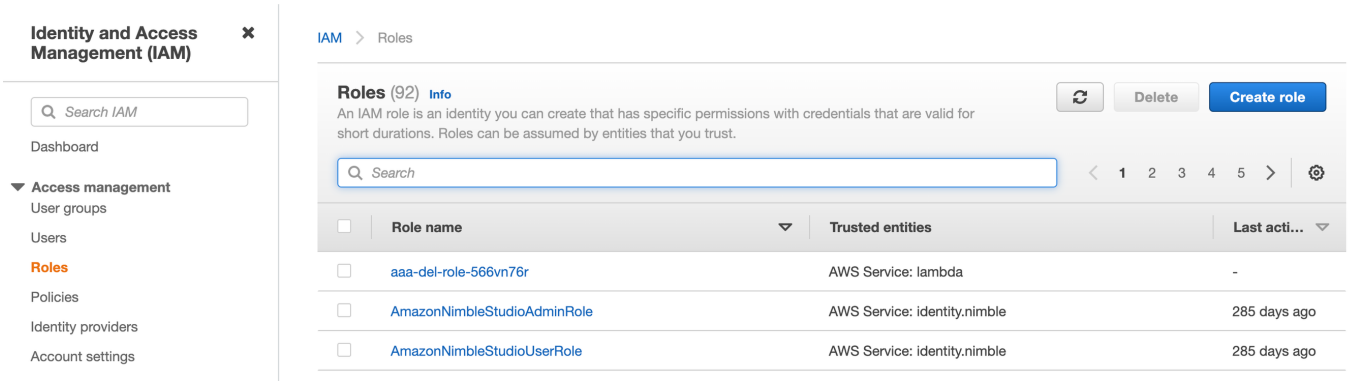
- Replace `<<AWS_ACCOUNT_ID>>`: Substitute `<<AWS_ACCOUNT_ID>>` with your actual AWS account ID. Your AWS account ID is a 12-digit number (e.g., `123456789012`) . **Do not include any hyphens**.
- Replace `<<LAMBDA_NAME>>`: Replace `<<LAMBDA_NAME>>` with the name of your Lambda function (e.g., my-lambda-function).

1. **Review and Create the Policy**:
    - After replacing the placeholders with the correct values, click **Next: Tags**.
        - If you need to add tags (optional), you can do so here, otherwise click **Next: Review**.
    - Give the policy a meaningful name, such as `ngeneahub_lambda_policy`.
    - Finally, click **Create Policy**.

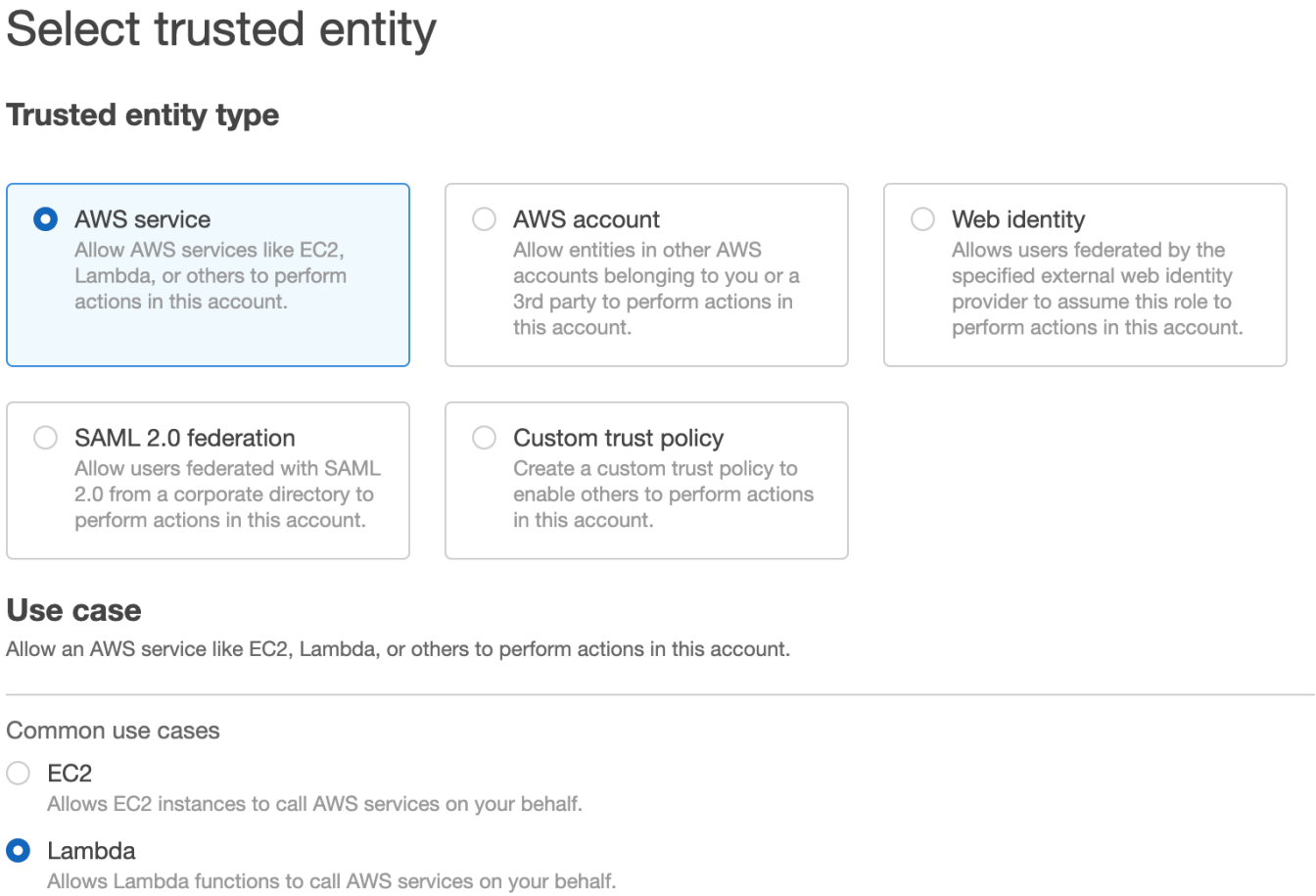


Create the IAM Role

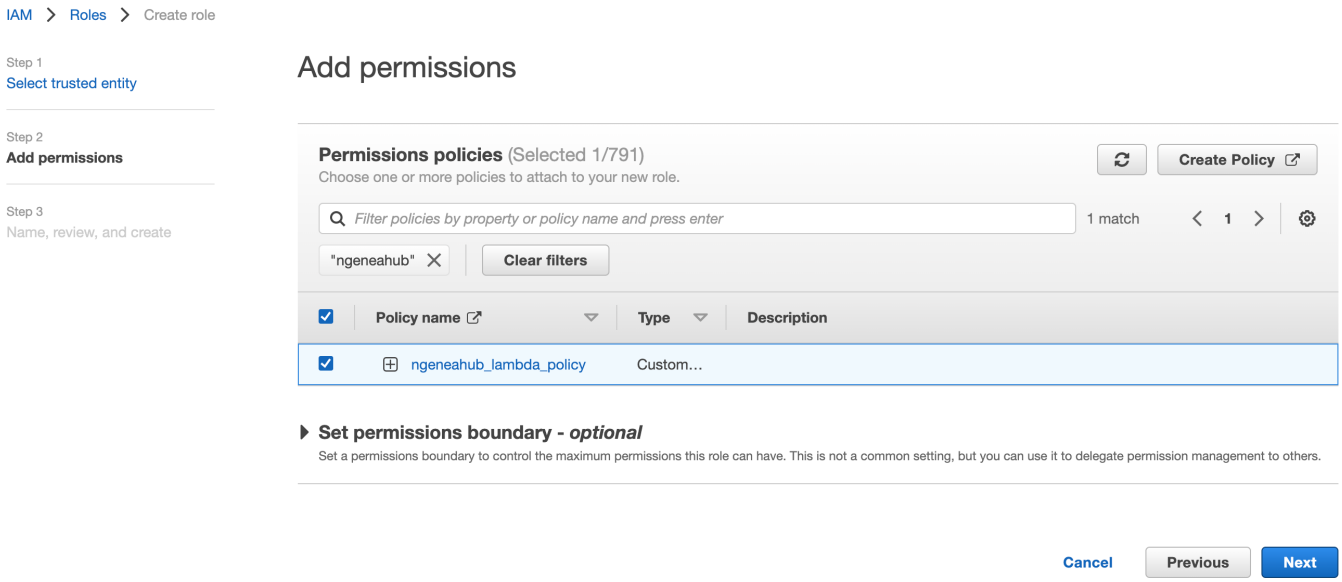
1. To **Create the IAM Role**, In the left-hand menu, select **Roles** and Click the **Cr eate role** button.



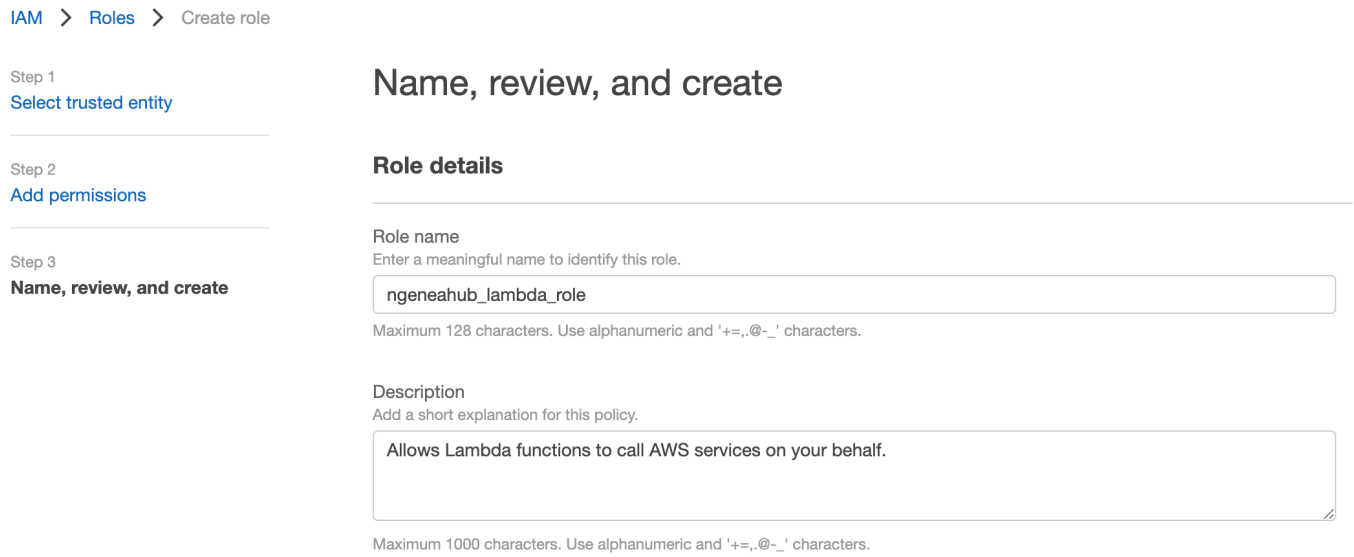
1. For the **trusted entity type**, choose **AWS service**. In the use case, select **L ambda**.



1. On the next screen, you'll be asked to attach permissions to the role. Search for the policy you just created, Select the policy, then click **Next: Tags**.
    - Again, if you need to add tags, you can do so here, otherwise click **Next: Review**.



1. Name the role `ngeneahub_lambda_role` to keep it descriptive and clear. Finally, click **Create Role**.



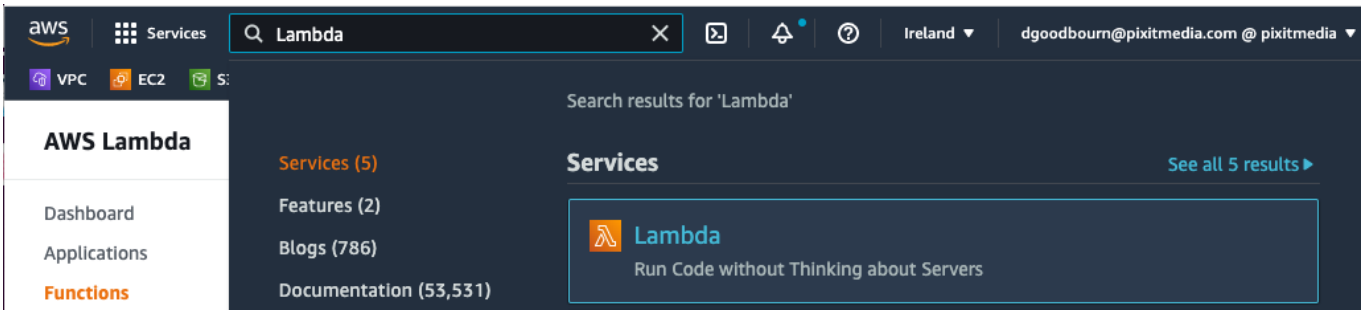
Create the Lambda function

A function in AWS Lambda is a small piece of code that does something specific when triggered. For example, you could write a Lambda function that resizes images whenever a new image is uploaded to S3. The Lambda function runs automatically when an event occurs, like uploading a file.

Creating the Function: To create a Lambda function, you start by selecting `Python 3.12` or above as the runtime and choosing the appropriate IAM role that defines the permissions for the function. If the function needs to connect to a private network, you configure the VPC, Subnet, and Security Group. You then upload the function code in a ZIP file, which contains the logic for Lambda to execute when triggered. Additionally, you provide a config.json file with necessary settings, such as API keys or database connections, for the function to operate correctly.

Follow the steps to create an AWS Lambda function.

1. Go to the AWS Management Console at Cloud Computing Services - Amazon Web Services (AWS) and log in with your account.
2. In the AWS Console, type “Lambda” in the search bar at the top and select Lambda from the dropdown list. This will take you to the Lambda service.

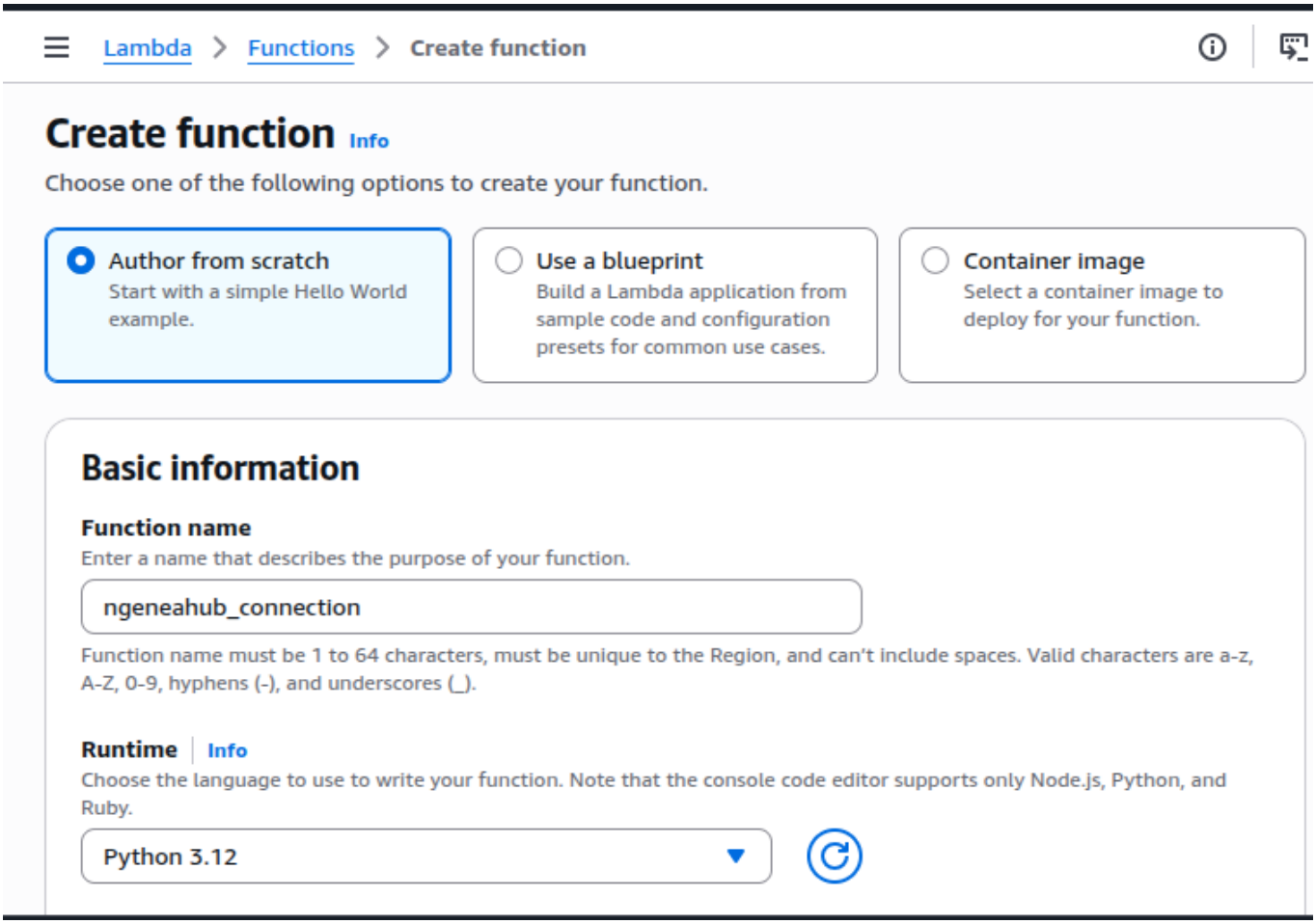


1. On the Lambda dashboard, you’ll see a button that says Create function. Click on it.
2. Select **“Author from scratch”**: This is the option that allows you to build a function from the ground up.
    - **Function name**: Enter a name for your Lambda function (e.g., “MyLambdaFunction”).
    - **Runtime**: From the Runtime dropdown, choose **Python 3.12+** (or another version of Python if required for your use case).

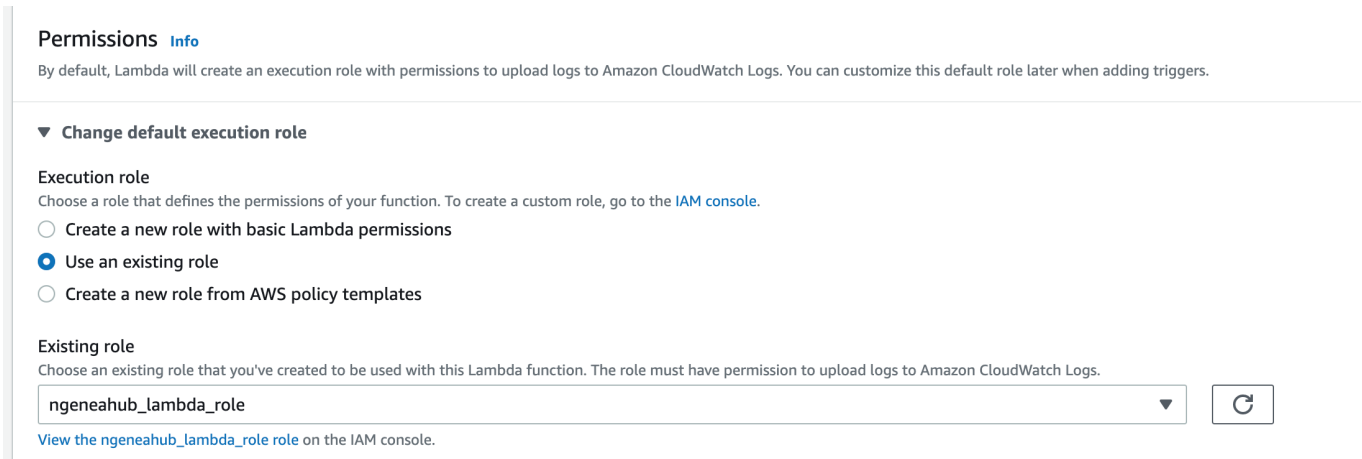
    Note: Re-enabling the Requests Module in AWS Lambda (Python 3.12 Runtime)

    Since AWS has removed Python 3.7 from the Lambda runtimes, it has also removed the **Requests module**. To continue using the Requests module in your Lambda functions running on Python 3.12, you will need to add it back by creating a custom Layer.

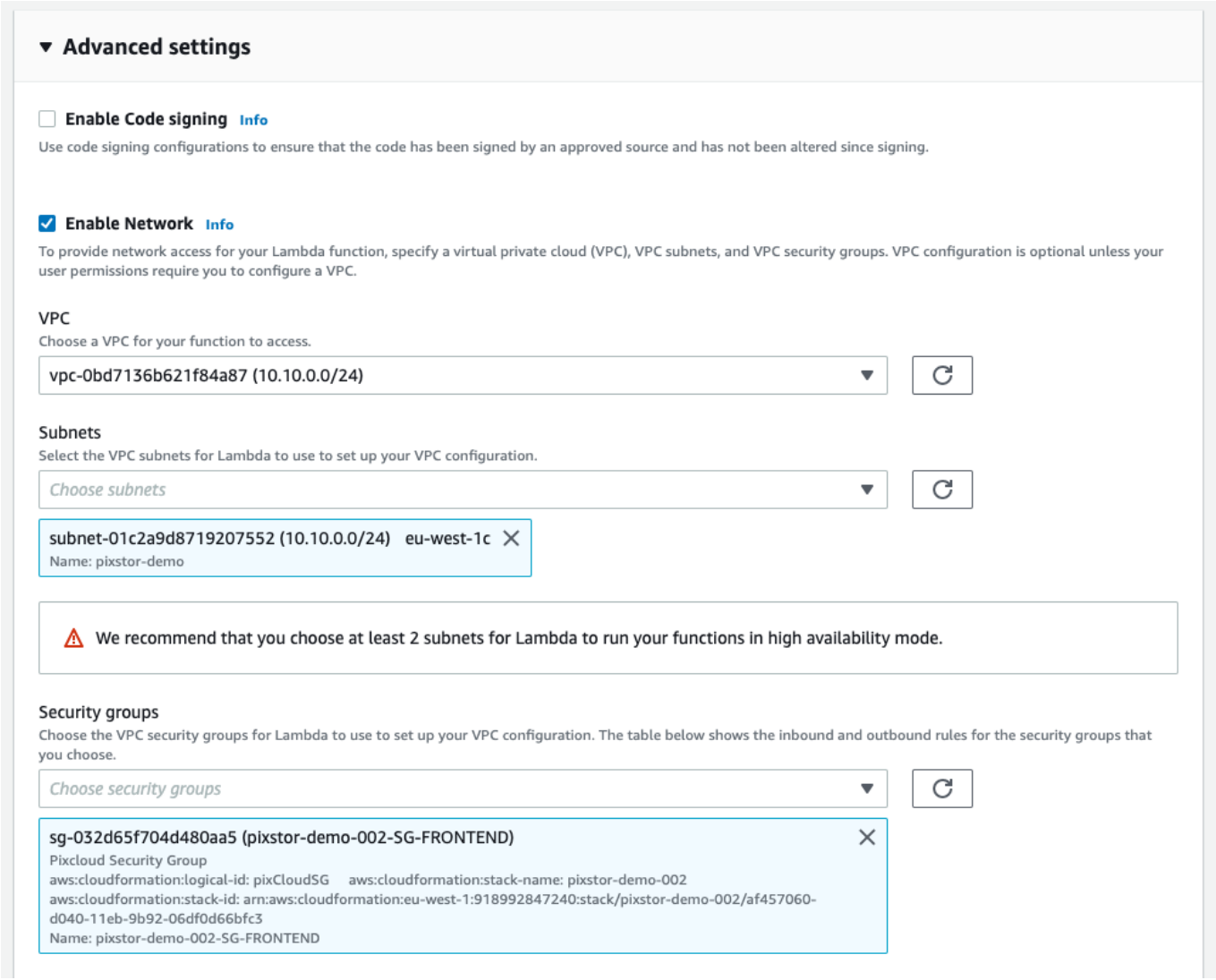
    Follow the steps provided the in section: **5.2.2.5. AWS Lambda with Python 3.12: Adding Requests Module** to add a custom Layer to your Lambda function, which will include the Requests module for your code to utilize.



1. Under **Execution role**, choose Use an existing role. In the **Existing role** dropdown, select the IAM role that you have previously created for this function (the one that has the appropriate permissions for your Lambda).



1. **Configure VPC and Networking (If Necessary)**: If your Ngenea Hub doesn't have an external IP to connect to, follow these steps:
    - Scroll down to Advanced settings and select **Enable Network**.
    - From the dropdown, select the **VPC** where the Ngenea Hub is located.
    - Select the **Subnet** within the VPC where the Ngenea Hub resides.
    - Choose a **Security Group** that allows access to the Ngenea Hub over port 8000 (Only if hub is being served on port 8000).



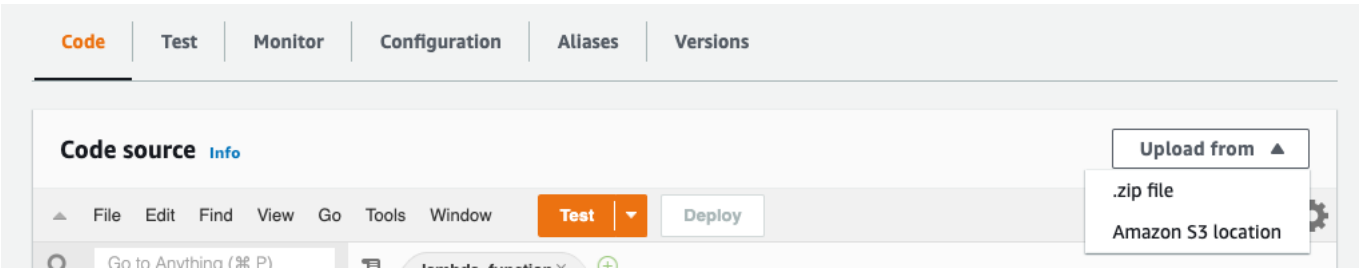
1. **Update the IAM Policy**: Your Lambda function will need permission to traverse the VPC's networking. Here's how to do it:
    - Go to the **IAM console** and find the policy you previously created. Add the following section to the policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:*:<<AWS_ACCOUNT_ID>>:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:<<AWS_ACCOUNT_ID>>:log-group:/aws/lambda/<<LAMBDA_NAME>>:*"
            ]
        },
```

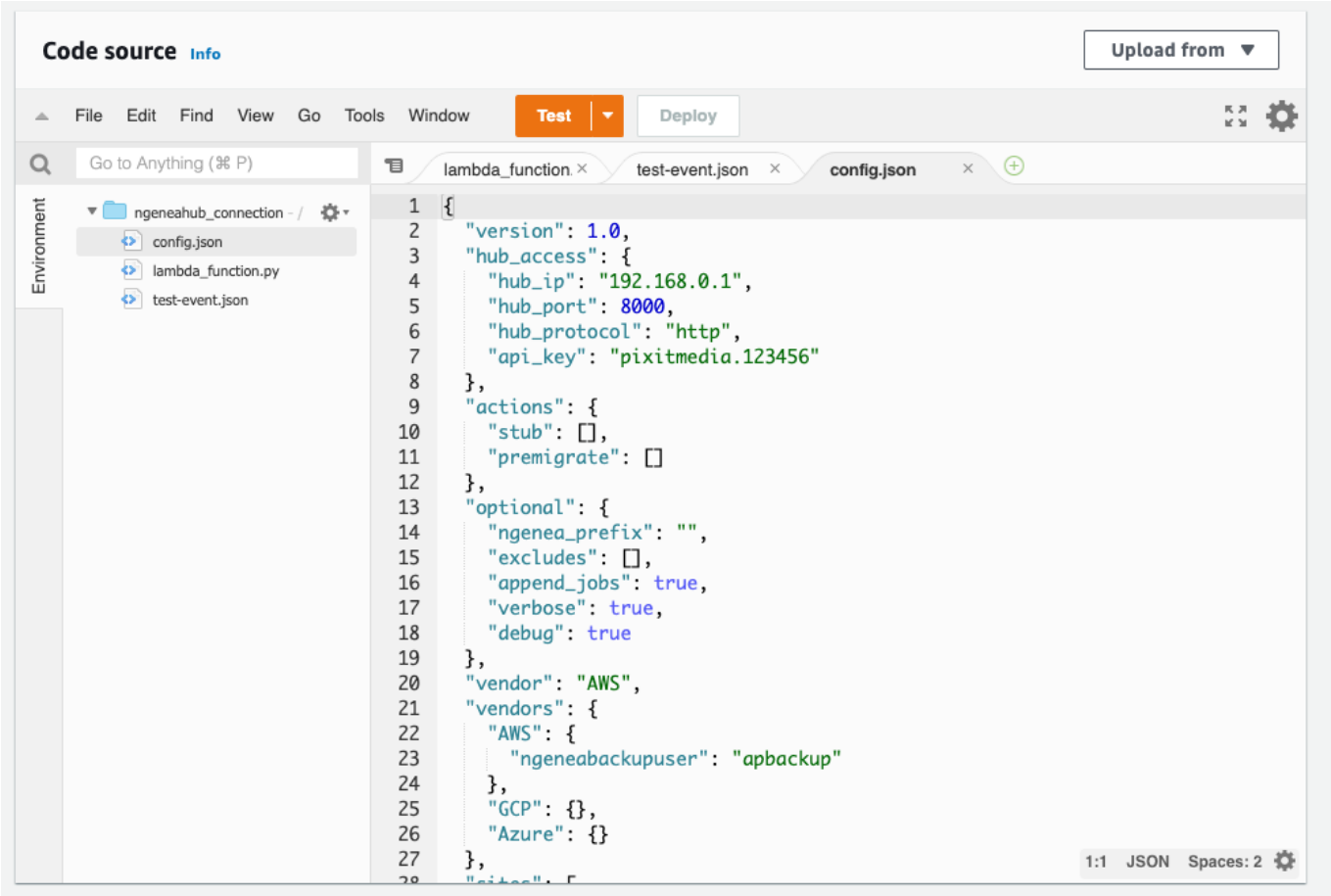
```
        {
            "Effect": "Allow",
            "Action": "iam:GetUser",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetUser",
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": "*"
        }
    ]
}
```

- Replace `<<AWS_ACCOUNT_ID>>` with your actual AWS account ID.
- Replace `<<LAMBDA_NAME>>` with the name of your Lambda function.
- Once you've updated the policy, **save** it.

1. **Create the Lambda Function**: With the role and network settings configured, click the **Create function** button. This will take a few minutes.
2. After your function is created, you will be redirected to the **function's configuration page**. Click on the **Code tab**.
3. Click on **Upload a .zip file**.
4. Click Upload, and then browse to select the **AWS zip file** you previously downloaded (from the ../../download page). Select the zip file and click **Save** to upload your code.



1. **Update the config.json File**: In the Code section, locate and edit the `config.json` file. You will need to update this file with all the relevant details based on the docs from Cloud Functions



1. Once you have updated the file, click **Deploy** to save your changes.

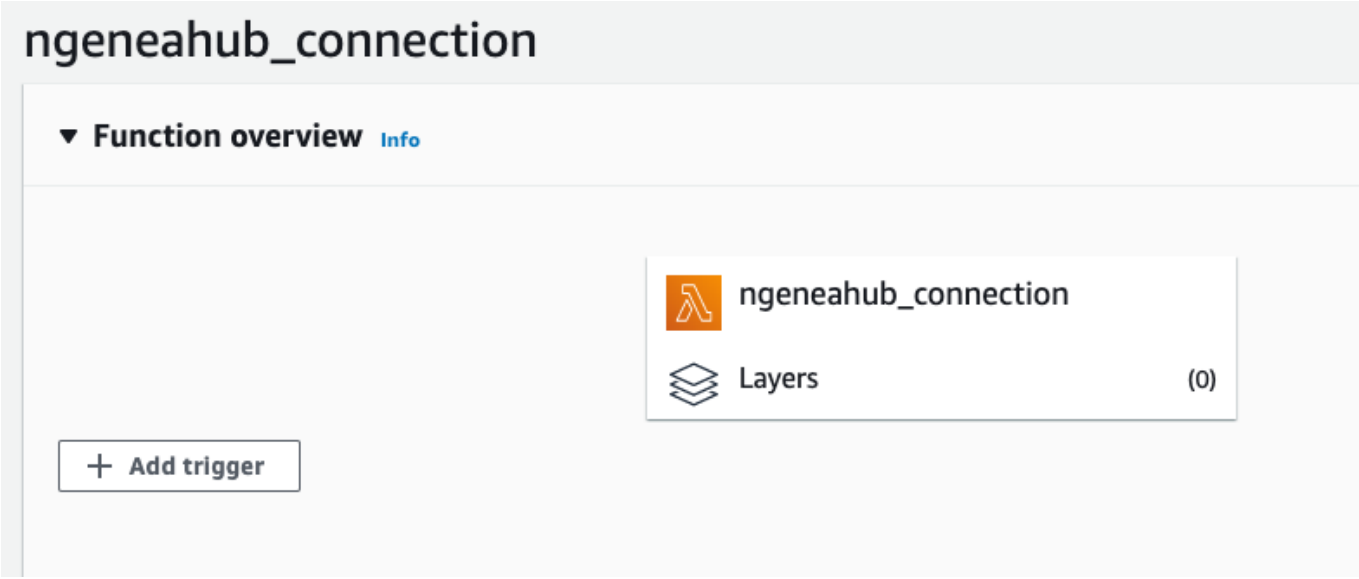
Set Up the Trigger (S3 Bucket Event)

Event Trigger: A trigger is an event that automatically activates a specific action, and in this case, you can set up an S3 trigger to invoke a Lambda function whenever a file is uploaded to an S3 bucket. By selecting the “All object create events” option, you ensure that Lambda responds to any new file uploaded to the bucket, regardless of the file type. This allows Lambda to automatically execute whenever a new file is created in the S3 bucket.

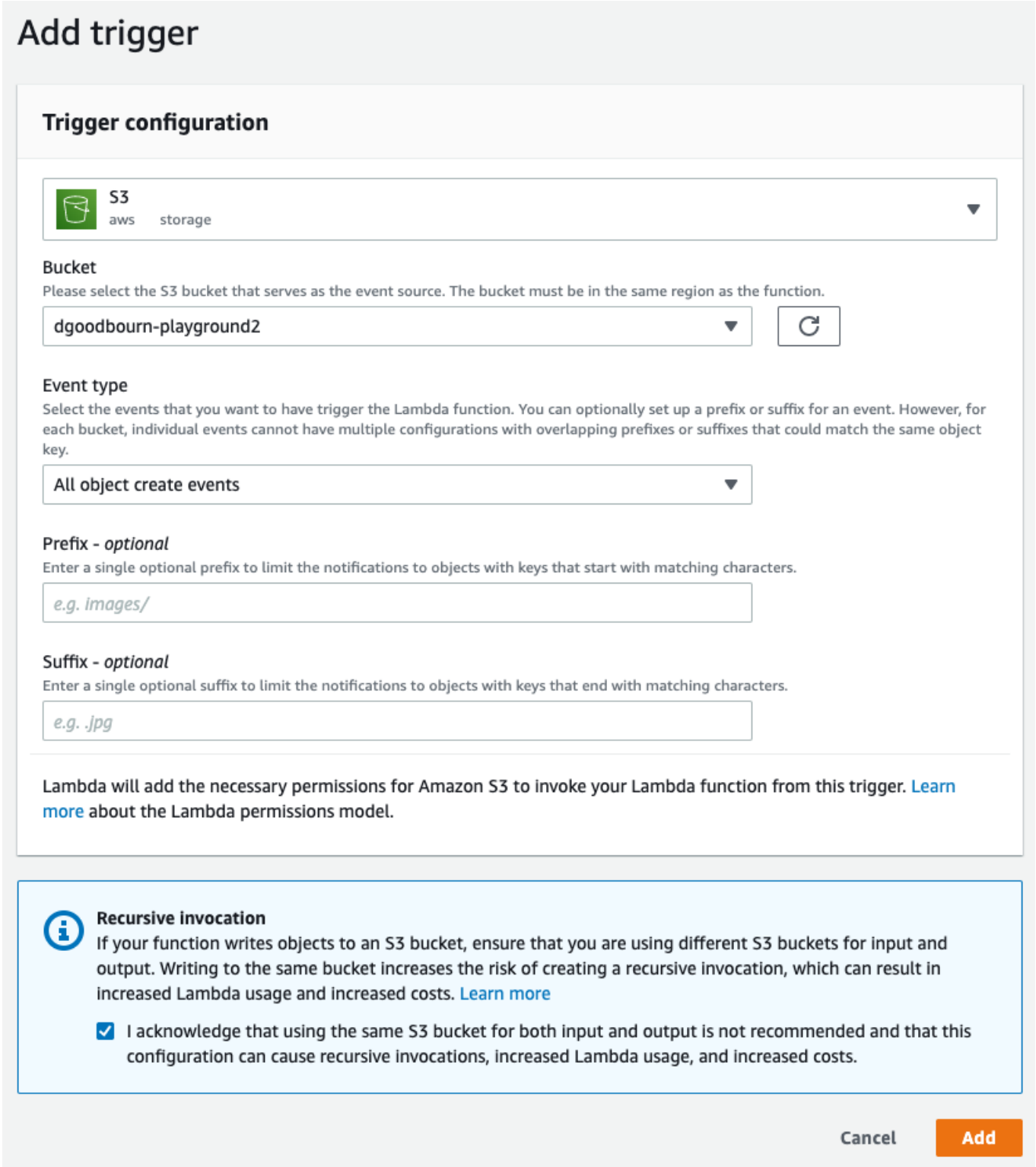
Recursive Invocation: This setting ensures that if Lambda’s own activities (like writing logs) somehow trigger the function again, it won’t end up running in an endless loop.

Assigning a trigger to your Lambda function using an S3 bucket event:

1. Open the **AWS Lambda Console**. Select the **Lambda function** you created earlier.
2. Under the **Function overview** section, click on **Add trigger**.



1. From the list of available triggers, choose **S3**.
2. In the **Bucket** dropdown, select the S3 bucket where you want the Lambda function to be triggered.
3. For **Event type**, select **All object create events**. This ensures that Lambda is triggered whenever any new object (file) is created in the S3 bucket.
4. Check the box labeled **Recursive Invocation checkbox warning**. This prevents Lambda from continuously triggering itself if it processes files within the bucket.
5. After selecting the options, click the **Add** button to assign the trigger to your Lambda function.



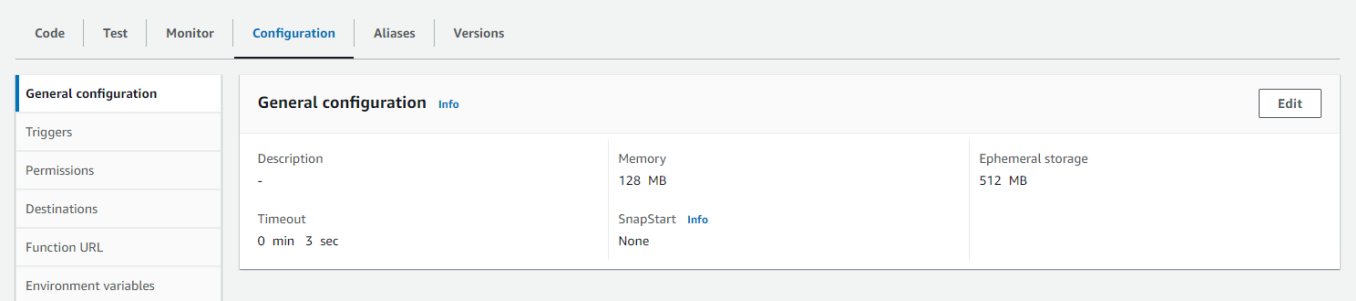
1. Your Lambda function is now configured with the S3 bucket event trigger, meaning it will automatically execute whenever a new file is uploaded to the selected bucket.

Configuring Timeout

Lambda runs your code for a set amount of time before timing out. Timeout is the maximum amount of time in seconds that a Lambda function can run. The default value for this setting is 3 seconds, but you can adjust this in increments of 1 second up to a maximum value of 900 seconds (15 minutes).

You can configure function timeout in the Lambda console. When your Lambda function is configured,

1. Select the function
2. Choose the **Configuration tab** and select **General Configuration**
3. Under **General Configuration**, choose **Edit**
4. For **Timeout**, set a value between 1 and 900 seconds (15 minutes)
5. Click on **Save**



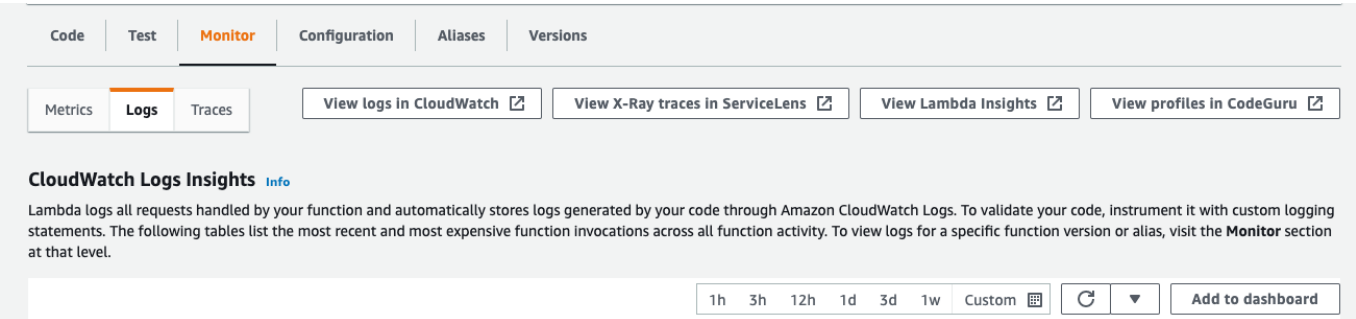
Monitoring Lambda Functions

CloudWatch: Amazon CloudWatch is an essential tool provided by AWS for monitoring various aspects of your AWS resources, including Lambda functions. It tracks vital metrics such as execution duration and logs that capture detailed information about the function's behavior.

By leveraging CloudWatch, you can determine if your Lambda function is performing correctly and identify any potential errors or issues.

Once a trigger is assigned to your Lambda function, you can monitor its performance in the **Monitor tab**. This tab provides a variety of monitoring options, giving you a comprehensive view of your function's performance.

For a more in-depth analysis, you can use View **Logs in CloudWatch**. This is the best place to access detailed logs and troubleshoot any issues your Lambda function might be facing.

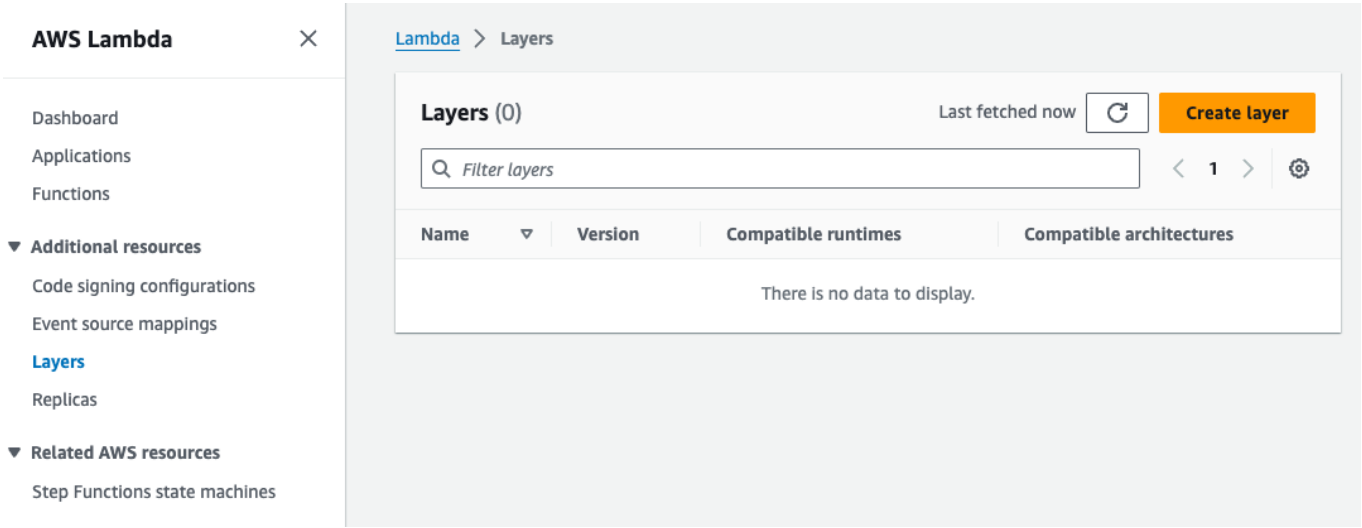


AWS Lambda with Python 3.12: Adding Requests Module

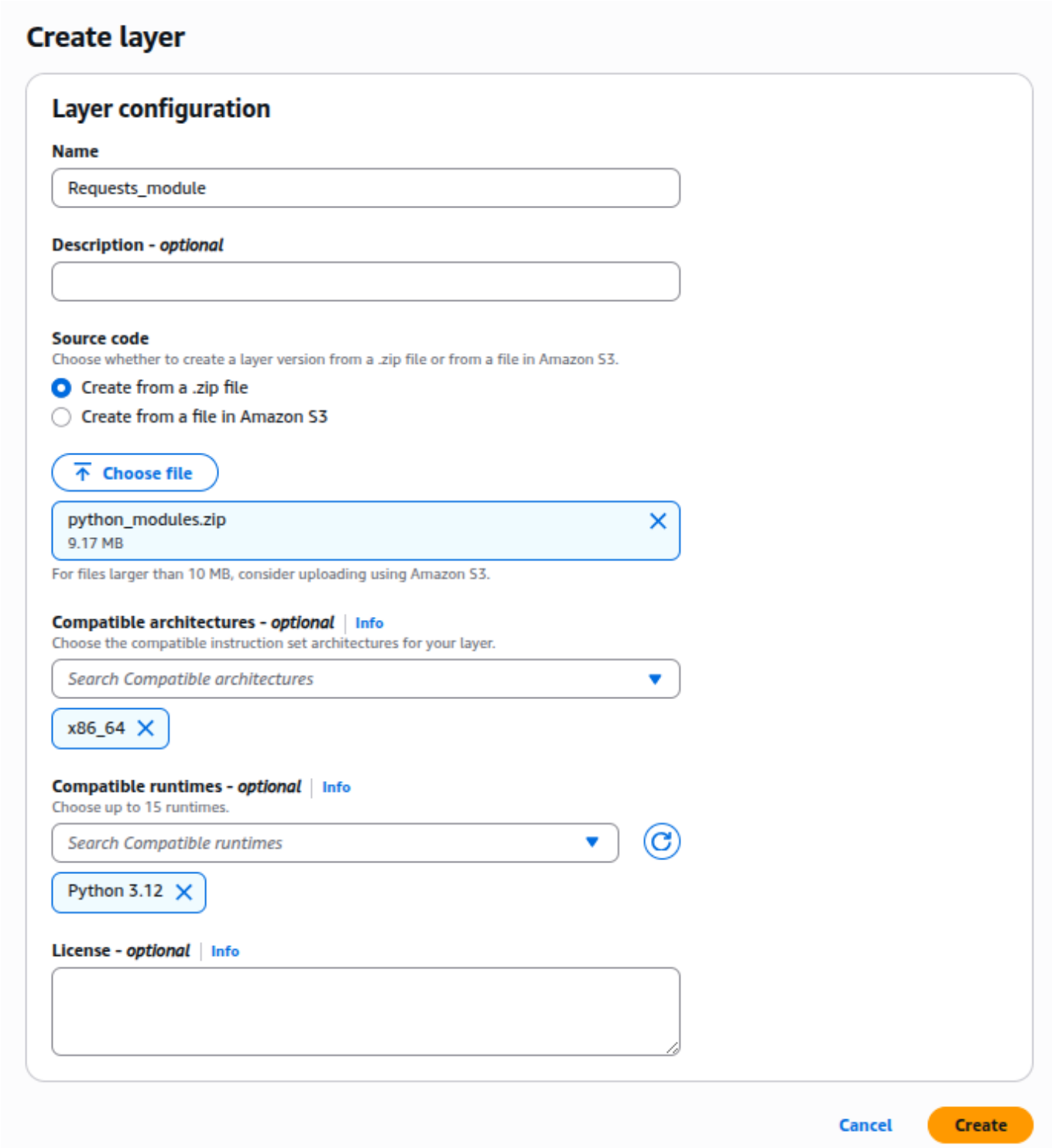
Create a Layer and attach it to the Lambda function.

> In AWS Lambda, a Layer is a way to package and share additional code, such as libraries, dependencies, or even custom runtime components, that your Lambda function can use. In this case, you can create a Lambda Layer that includes the Requests library and attach it to your function.

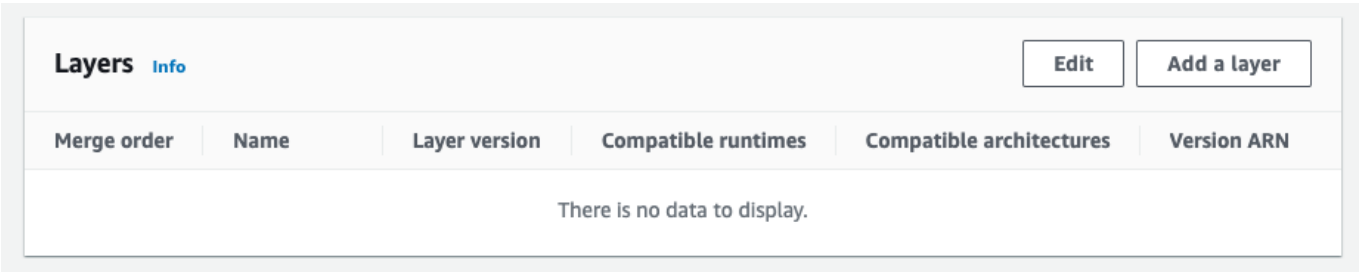
1. **To create a new Lambda Layer**, In the AWS Console, go to **Lambda** and click **Layers** in the left menu. Click **Create Layer** to continue.



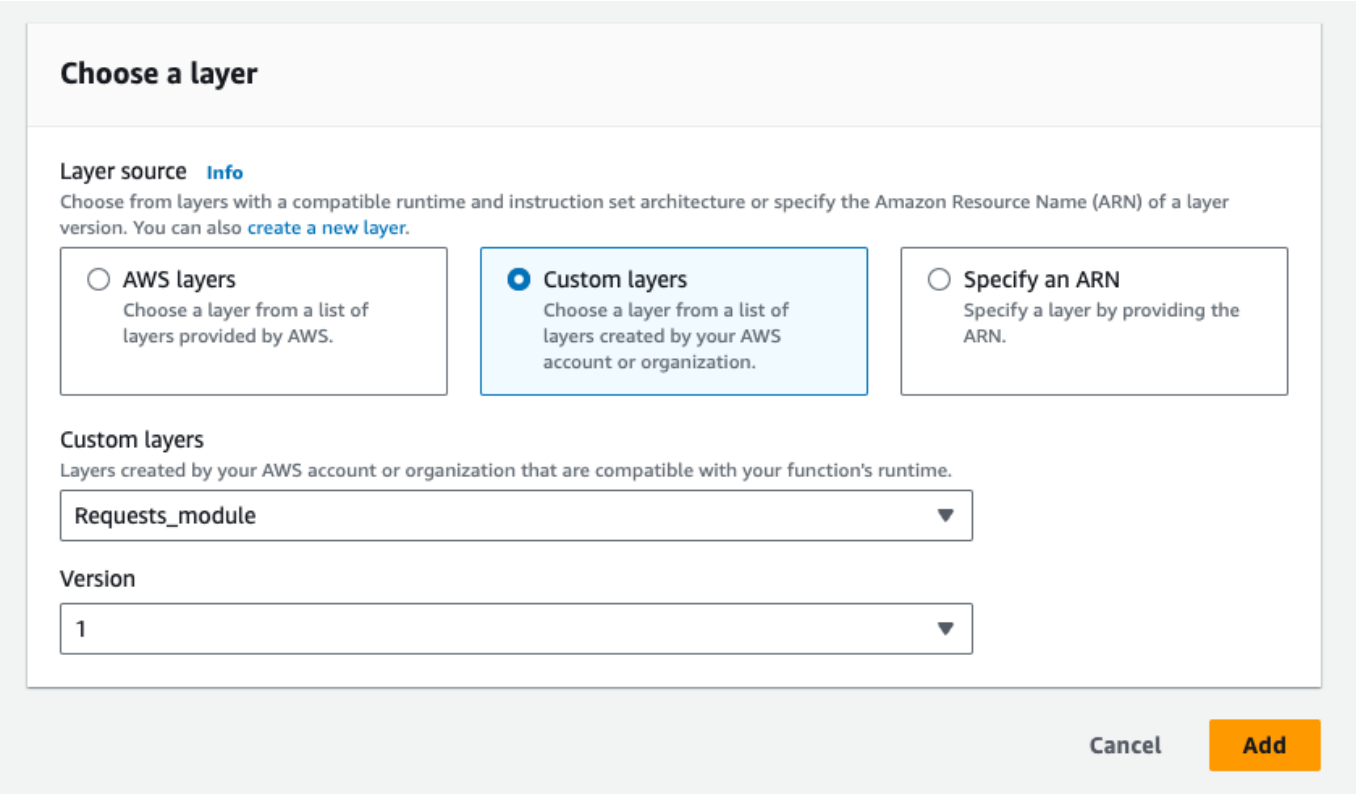
1. Enter a name for your layer (e.g., `RequestsLayer`).
2. Upload the `python_modules.zip` zip file.
3. Select `x86_64` architecture and `Python 3.12` runtime. Click **Create** to save.



1. In your Lambda function, scroll down to the **Layers** section and click the **Add a Layer** button.



1. Select the **Custom layers** option, then choose the layer you created from the **Custom layers dropdown** and pick the **Version**.



1. Your Lambda function should now work, as the `Requests` module is successfully loaded.

Software

The cloud function software for use with Ngenea Hub is available on the Ngenea Hub server at `/opt/arcapix/samples/ngenea-hub/cloud_functions/`, or at the same location in the `ngenea-hub` rpm.

Configuration

The configuration file `config.json` is the key to setting up Cloud Functions and defining how they should behave. This file is written in JSON, and it tells the system what actions to take when certain events occur. You will configure the settings to ensure everything works correctly.

Here are the most important sections of the configuration file:

version

The version field specifies the version of the configuration format. Currently, the only supported version is `1.0`.

hub_access

To integrate and interact with Ngenea Hub, you need to configure a set of settings to establish communication with its `REST API`. These settings ensure that your system or application can properly authenticate, submit files, and trigger workflows for

processing. Below is an explanation of each configuration setting, along with example values.

- **hub_ip** : The IP address of the Ngenea Hub REST API. This identifies the location of the Ngenea Hub server within your network or on the internet. Example: `hub_ip = "192.168.1.100"`
- **hub_port** : The port number used by the Ngenea Hub REST API. By default, the Ngenea Hub API may be set to use port 8000, but this can vary depending on your specific setup. Example: `hub_port = 8000`
- **hub_protocol** : The protocol used for communicating with the Ngenea Hub API. You can choose between http (insecure) and https (secure). Example: `hub_protocol = "https"`
- **api_key**: An API key used for authentication with the Ngenea Hub. This key ensures that your system has the necessary permissions to interact with the Hub. Example: `api_key = "your-api-key-here"`
- **workflow**: The name of the workflow you wish to submit for processing an event file. Workflows define a set of tasks or steps that Ngenea Hub follows to process data.
  - **reverse_stub**: Typically used for new files, reversing or processing them in a specific way. Example: `workflow = "reverse_stub"`
  - **recall**: Another default workflow for recalling or processing files.
- **workflow_flags**: A mapping of additional settings you can pass to customize the behavior of the selected workflow. These flags allow you to fine-tune the process by specifying certain options.Example: `workflow_flags = {"hydrate": true}`
- **Custom Workflows**: By default, Ngenea Hub does not have a delete workflow. If you need to delete files, you must create a custom workflow to handle file deletion. As shown below, you might create a workflow named `delete_file` for this purpose:

```
{
  "name": "delete_file",
  "label": "delete_file",
  "icon_classes": [],
  "discovery": null,
  "enabled": true,
  "visible": true,
  "fields": [],
  "filter_rules": [
    {
      "type": "all",
      "state": "all",
      "action": [
        {
          "name": "dynamo.tasks.delete_paths_from_gpfs",
          "recursive": false
        }
      ],
      "description": "Delete the file at a given path"
    }
  ]
}
```


Sites

In systems like Ngenea Hub, events (such as file updates or deletions) need to be reflected across multiple sites to maintain consistency.

- **Site Name** : Each site in the Ngenea Hub system is identified by a **name**. This name is crucial for tracking and managing events on a particular site.
- **Default Mode for 'Recall' Workflows**: When an event occurs, the system needs to decide how to recall or reflect that event across sites. If the event path does not match a specific action, the **default mode** is used.
- The two available default modes are:
  - **Stub**: A placeholder or temporary record used when the system doesn't have detailed information.
  - **Premigrate**: This mode prepares the system for future migration or change before the event is fully applied.
- **Skip from Ngenea**:
  - If the system is syncing files between multiple sites, there may be situations where certain files should not be reflected or recalled on sites. This is where the **skip_from_ngenea** setting comes into play.
  - When **skip_from_ngenea** is set to **True**, files created or transferred via Ngenea Hub will be skipped for recall. This is useful because files transferred through Ngenea Hub are already in sync between sites, so recalling them again is unnecessary.
  - However, if a file is uploaded directly to the cloud (outside of Ngenea Hub), it still needs to be reflected or recalled across the sites to ensure consistency.

- **Handling Delete Events**:
  - Deleting a file or event can sometimes be tricky, especially when using cloud platforms like GCP (Google Cloud Platform). In these cases, it may be unclear where the file was deleted from, which can complicate the process of reflecting that delete action.
  - As a result, when a **delete event** occurs and the system cannot determine its origin, the system will reflect the delete on **all sites** to ensure that no inconsistent or outdated data remains across the sites.

By configuring these settings carefully, you can ensure that the right events are reflected accurately across all your sites, while avoiding unnecessary updates or recalls for already-synced data.

Actions

Mapping of actions – `stub` or `premigrate` – to path prefixes.

This section explains how actions such as “stub” or “premigrate” are mapped to specific path prefixes. These mappings determine whether a file should be included in a “recall” workflow, where the file is downloaded and prepared for use.

If a path matches multiple actions, the longest matching prefix will take priority. For example, consider the following configuration:

```
{
    "stub": ["data"],
    "premigrate": ["data/cats"]
}
```

In this case:

- The path `data/cats/cat-01.jpg` will undergo **premigrate**, meaning this action prepares a file for migration to a different storage location, typically involving downloading and preparing the file for a transition or backup.
- The path `data/cats-02.jpg` will be **stubbed**, meaning it will remain as a placeholder without being fully downloaded.

If no specific action is defined for a given path, the default action for the site (as mentioned earlier) will be applied.

However, if the default action is not set, or if `hydrate` is explicitly specified in the `workflow_flags`, that setting will take priority and override the default behavior. If no configuration is provided at all, the default action is to stub the file.

This system allows for more flexibility and precise control over how files are handled during the migration and recall processes.

optional

Optional settings are additional settings that can be configured in the `config.json` file to customize the behavior of the system. While their use is not mandatory, they offer increased flexibility for those seeking to refine and optimize the system's functionality. Below is a detailed overview:

- **ngenea_prefix** : Allows you to map a cloud path to a local storage path. For example, if you have a file in the cloud at `data/cats-01.jpg`, but want it to appear in a specific folder on your local system, like `/mmfs1/data/cats-01.jpg`, this setting adds a prefix to the cloud file path. This ensures it matches your local storage path. By **default**, the setting is empty `('')`, meaning no mapping occurs unless you specify one.
- **excludes** : Allows you to exclude specific paths or files from being processed. For example, if you want to ignore files in the `logs/` folder, you can add `logs` to the `excludes` list. By **default**, this setting is an empty list `([])`, meaning no exclusions are applied unless explicitly specified.
- **append_jobs** : The `append_jobs` setting controls whether multiple tasks should be grouped under the same job ID. If set to `true`, it groups tasks that occur within the same hour under one job ID. If set to `false`, each task will be assigned its own unique job ID. By **default**, this setting is `false`, meaning tasks are handled separately unless modified.
- **verbose** : The `verbose` refers to the level of detail included in the output, especially in logs or messages. If set to `true`, it displays general `info level` logs, which provide information on the system's activities. If set to `false`, only `minimal log` information will be shown. The **default** value is `false`, meaning only basic logs are shown unless specified otherwise.
- **debug** : The `debug` setting provides detailed logs for troubleshooting. When set to `true`, it displays 'debug level`logs, which contain in-depth technical information for resolving issues. If set to`false`, it does not show detailed debug information. The **default** value is` false`, meaning fewer details are shown unless enabled.
    - **Note** : If both verbose and debug are enabled, the debug logs will take priority and provide more detailed information.

vendor

This setting tells the system which cloud service is being used. Right now, there are two options:

- **AWS** (Amazon Web Services)
- **GCP** (Google Cloud Platform)

So, if you're using AWS or GCP for your system, you would set this option to either "AWS" or "GCP" to let the system know which cloud platform it should work with.

vendors

This section contains settings that are specific to a particular cloud platform (vendor), such as AWS. **Currently, these settings are only used for AWS.**

- **AWS-specific setting - ngeneabackupuser**:
  - The `ngeneabackupuser` is the name of the user account that Ngenea uses within AWS. It's used to identify whether a file came from the Ngenea system when processing files.
  - This identification is helpful for situations where you might want to skip files that were already uploaded or created by Ngenea, especially when the `skip_from_ngenea` setting is **enabled**.
  - Essentially, this ensures that files associated with Ngenea are handled appropriately, preventing redundant actions or unnecessary processing.

Complete Example

```
{
    "version": 1.0,
    "hub_access": {
        "hub_ip": "192.168.0.1",
        "hub_port": 8000,
        "hub_protocol": "http",
        "api_key": "pixitmedia.123456",
        "workflow": "reverse_stub",
        "workflow_flags": {
            "hydrate": false,
            "overwrite": true
        }
    },
    "sites": [
        {
            "site": "uk",
            "default": "stub"
        }
    ],
    "actions": {
        "stub": [],
        "premigrate": []
    },
    "optional": {
        "ngenea_prefix": "",
        "excludes": [],
        "append_jobs": true,
        "verbose": true,
        "debug": true
    },
    "vendor": "AWS",
    "vendors": {
        "AWS": {
            "ngeneabackupuser": ""
        },
        "GCP": {},
```

```
        "Azure": {}
    }
}
```

Disaster Recovery / Cold Failover

It's possible to configure Ngenea Hub to be able to cold-failover to another node if it's running on a PixStor.

Setup

Configure datastore

Configure Ngenea Hub to store it's persistent data on the GPFS filesystem so it can be read by multiple nodes. This is done by settings the following setting in `/etc/sysconfig/ngeneahub`

```
DATA_DIR=/mmfs1/.arcapix/ngeneahub/data
```

Configure Networking

It's strongly recommended to configure a floating IP that can be used for the Ngenea Worker to connect to. This will allow cold failover without having to reconfigure workers.

This can be done by setting the following settings in `/etc/sysconfig/ngeneahub`:

- `SERVICE_CIDR`. Set this to the IP and netmask of the IP you want to be managed by ngeneahub. e.g. `192.168.2.3/24` for the IP `192.168.2.3` on a network with a netmask of `255.255.255.0`
- `SERVICE_INTERFACE`. Set this to the name of the interface the IP address should be added to. e.g. `man0`

Configure the workers to use this IP by editing `/etc/ngenea/ngenea-worker.conf` on each worker node and modifying `broker_url` and `result_backend`

Install Ngenea Hub

Install Ngenea Hub on multiple nodes as usual. Make sure `/etc/sysconfig/ngeneahub` are in sync across these nodes. Enable and start the service on one node only. Leave the service disabled and stopped on the other nodes.

Performing failover

In the case of a node failure, after confirming the services are no longer running on the other node, the following steps can be performed to bring the service up on another node:

important You must be certain the service is not running anywhere else before continuing, otherwise data loss can occur.

- Remove the lock file from `${DATA_DIR}/.lock`.
- Start the Ngenea Hub service

Migration from local datastore

After setting `DATA_DIR` in `/etc/sysconfig/ngeneahub` and restarting the service, data will automatically be migrated. This is a one-way operation.

Installing SSL Certificates

In a PixStor environment, foundational web server setup tasks—such as installing NGINX, configuring it, and setting up a proxy for Hub access—are already handled as part of the standard deployment. This simplifies the process for administrators, as they don't need to repeat these initial steps.

Generating an SSL certificate falls outside the scope of our guidance (as this typically depends on organizational security policies or certificate authorities).

Let us understand, **How to Install SSL Certificates**

PixStor systems include self-signed SSL certificates. To replace the default certifications with custom certificates, overwrite the certificate files on each of the PixStor nodes:

`/etc/pki/tls/certs/arcapix.crt` - the full server certificate

`/etc/pki/tls/private/arcapix.key` - the private key for the certificate

Ensure that the file permissions are set appropriately on these files:

```
chmod 644 /etc/pki/tls/certs/arcapix.crt
chown root:root /etc/pki/tls/certs/arcapix.crt
chmod 640 /etc/pki/tls/private/arcapix.key
chown root:nginx /etc/pki/tls/private/arcapix.key
```

Once updated these files, restart the `nginx` web server to load the new certificates:

```
systemctl restart nginx
```

Note: To learn about **"How to set up an nginx proxy for hub if you're not using PixStor"**, please contact us.

Queues

When a system needs to handle many tasks (called jobs), it organizes them into different **queues**. A **queue** is like a waiting line where jobs sit until a worker is ready to run them.

By using **custom job queues**, you can control how and when certain jobs run. Each queue can have its own settings, like how many jobs it can run at the same time.

Here are some examples:

- **High Priority Queue**: This queue is for important jobs that need to be finished quickly. It uses more threads (like more workers), so these jobs are handled faster.
- **Low Priority Queue**: This queue is for jobs that are not urgent. It uses fewer threads, which means jobs here will take longer, but that's okay because they're not time-sensitive.
- **Transparent Recall Queue**: This special queue is only for transparent recall jobs. It ensures these jobs aren't delayed by other tasks, so they always get the attention they need.

In addition to these custom queues, there is always a default queue. If you don't choose a specific queue when submitting a job, it will automatically go to the default queue.

Configuration

Queues are set up in a file called the worker configuration file, which is located at: `/etc/ngenea/ngenea-worker.conf`

To create a new queue, you simply add a new section to this file using the format: `[Queue queue_name]`

For example, to create a queue named **highpriority** that can run 20 jobs at the same time, you would add this:

```
[Queue highpriority]
threads = 20
```

Rules for Naming Queues:

- You can use **letters, numbers, underscores (_), and hyphens (-)**.
- **Spaces are not allowed** in queue names.

Each queue can have settings to control how it works. The most important one is:

```
- **threads**: This tells the system how many jobs it can run at
the same time in that queue.
```

Inheriting Settings

If you don't add any settings to a queue, it will **inherit the default (global) settings** from a special section in the same file called `[settings]`.

Here's an example:

```
[settings]
...
```

```
threads = 4

[Queue transparent_recall]

[Queue highpriority]
threads = 10
```

In this case:

- The **highpriority** queue will run up to **10 jobs at once** because it's explicitly set.
- The **transparent_recall** queue doesn't specify a number of threads, so it **uses the global setting**, which is 4.
- If there is no global `threads` setting at all, the system uses **10 threads by default**.

Functions

Job queues can be used for different types of tasks, called functions. Each queue supports a specific set of these functions.

Functions in Custom Queues

Custom queues (the ones you define yourself) can run the following functions:

- `worker` – runs the main workflow tasks
- `discovery` – runs discovery tasks like recursive scans or snapdiff
- `custom` – runs tasks from custom plugins.

Functions in the Default Queue

The **default queue**, which always exists, supports all the functions listed above **plus two more**:

- `interactive` – handles internal tasks like browsing files
- `settings` – handles configuration tasks like creating spaces or setting policies

Note: **Custom queues do not support** `interactive` **or** `settings` **functions**. Only the default queue can run these.

By default, all functions in a queue share the same number of threads. For example, if a queue has 20 threads, each function can use those 20 threads as needed.

You can also **customize how many threads each function gets**.

Here's an example:

```
[Queue highpriority]
threads = 20
custom = {"threads": 10}
```

In this example:

- The **worker** and **discovery** functions will each use up to **20 threads**.
- The **custom** function is limited to **10** threads, even though the total thread count is 20.

Note: If you set per-function settings in the `[settings]` section (the global configuration), they **only apply to the default queue**. These settings are **not inherited** by your custom queues.

Disabling Functions

You can **turn off (disable)** specific functions for a queue if you don't want certain types of tasks to run there.

For example, if you don't want a queue to run custom plugin tasks, you can disable the `custom` function by setting it to `false`.

Example:

```
[Queue no-custom]
custom = false
```

In this example:

- The custom function is disabled.
- That means **custom plugin jobs will not run** in the `highpriority` queue.
- This is useful when you want a queue to handle only specific types of jobs and avoid others.

In the same way, `default` queue functions can be disabled.

For example, in a cluster, you may set the management node to only run settings tasks

```
# management node
[settings]
discovery = false
worker = false
custom = false
```

And, you set ngenea nodes to only run workflow tasks, and not run settings tasks

```
# ngenea node
[settings]
settings = false
```

Warning: Each default queue function must be enabled on at least one node within a cluster, or else Hub will not function correctly.

Queue discovery

When you update the worker config file to **add a new queue**, it will automatically start up. If you **remove a queue** from the config, it will automatically shut down.

If this doesn't happen right away, you can force the worker to reload the config with this command:

```
systemctl reload ngenea-worker
```

This reloads the worker config without affecting any queues that haven't changed.

Queue Visibility in Ngenea Hub

Ngenea Hub automatically picks up new queues shortly after they're started, using worker heartbeats. New queues will show up in the Hub and can be used in workflows almost immediately.

However, **removing a queue** from the config **doesn't automatically remove it from Ngenea Hub**.

Removed queues must be manually de-registered using:

```
ngeneahubctl manage remove_queue <site> <queue>
```

Re-using a Queue Name

If you want to re-use a queue name that was previously removed:

First, restart the worker completely: `systemctl restart ngenea-worker`

Then remove the queue using this command:

```
ngeneahubctl manage remove_queue --offline-only <site> <queue>
```

- `--offline-only` will only remove the queue if it's not running. If the queue is still active, it will be recreated.
- To ensure a queue is completely removed and doesn't come back, shut it down first and then use `--offline-only`.

Note: If you submit a job to a queue which has been shutdown and not de-registered, the job will not be processed unless the queue is configured and started up again.

Queue removal

When a queue is removed from the worker configuration file and the worker is reloaded, the queue will be shutdown.

However, the removed queue will *not* be automatically de-registered from Ngenea Hub.

Warning: If you submit a job to a queue which has been shutdown and not de-registered, the job will not be processed unless the queue is configured and started up again.

Removed queues **must** be manually de-registered using the following command

```
ngeneahubctl manage remove_queue <site> <queue>
```

If the queue is brought back online, or if a new queue is created with the same name, it **will not** be recreated in Ngenea Hub until the queue has expired.

Removed queues expire when no heartbeats have been received for some interval after being removed. The expiry interval is configured using the `REMOVED_QUEUE_CLEANUP_INTERVAL` setting, which defaults to 1 day.

To immediately re-use the same queue name of a previously removed queue, the `--offline-only` flag can be used. This must be preceded by a complete worker restart

```
systemctl restart ngenea-worker
```

- `--offline-only` will remove the queue only if the queue is offline.
- If the queue is still online, or comes back online, it will be automatically recreated.

Note: If you send a job to a queue that has been shut down but not de-registered, the job won't be processed until the queue is set up and started again.

Memory considerations

Queues use about **40MB of RAM per thread** when no tasks are running.

Each queue has three functions, and if all three functions in a queue are set up with the same number of threads, the memory needed will be roughly: **threads * 125MB**.

For example:

- A high-priority queue with 20 threads per function (60 threads in total) will use about 2.5GB of memory when no tasks are running.
- A low-priority queue with 5 threads per function (15 threads in total) will use about 625MB of memory when no tasks are running.

Keep in mind that **task execution** will require additional memory on top of this baseline. The more threads there are, the more tasks can run simultaneously, which means a queue with many threads can use a lot of memory at peak times.

Workflows

When submitting a workflow, you can specify a queue to use. If no queue is provided, the **default queue** will be used.

For workflows that involve tasks running on **multiple sites** (like send or sync workflows), the **same queue** will be used across all sites. If the specified queue doesn't exist on one of the sites, the **default queue** will be used instead.

You can also assign a different queue to each task within a workflow. This is done by specifying the queue in the task parameters, like so:

```
{
    "name": "dynamo.tasks.migrate",
    "queue": "highpriority"
}
```

Selecting Queues at Runtime

When you're setting up a workflow, you can choose which queue to use for each task at the time the workflow runs (this is known as "runtime"). This allows you to dynamically decide which queue a task should use, based on the situation or inputs you provide while the workflow is executing.

You can use workflow **fields** to define which queue to use for tasks within that workflow. These fields allow you to pass values into the workflow at runtime.

For example, in the built-in **send workflow**, there's a field called `destinationqueue`. This field can be used to specify the queue that should be used for the task, and it can be passed when the workflow runs, just like how you pass the `destinationsite` field to specify the site the task should go to.

Here's an example to show how it works:

```
{
    "paths": [
        "/mmfs1/data/project_one",
    ],
    "site": "london",
    "queue": "highpriority",
    "workflow": "send_to_site",
    "fields": {
        "destinationsite": "dublin",
        "destinationqueue": "lowpriority"
    }
}
```

For more information on constructing and running workflows, see Custom Workflows

Usage

Monitoring and Management

Managing Ngenea Hub with `systemd`

`systemd` is the tool that makes sure all the necessary programs and services on a Linux system start automatically, run properly, and restart if they crash.

The `ngeneahub` systemd service ensures that the application starts automatically on boot, stays running in the background, and can be easily managed using standard Linux commands like systemctl.

The service name is: `ngeneahub`

You can use the following commands to manage it:

```
sudo systemctl status ngeneahub     # Check if it's running
sudo systemctl start ngeneahub      # Start the service
sudo systemctl stop ngeneahub       # Stop the service
sudo systemctl restart ngeneahub    # Restart the service
```

Using the Command-Line Tool: ngeneahubctl

`ngeneahubctl` is a command-line tool designed to manually control the individual components of Ngenea Hub, such as starting, stopping, or checking the status of services like the web app or background jobs.

Unlike systemd, which manages the entire Ngenea Hub service as a whole, `ngeneahubctl` offers more fine-grained control and is especially useful for debugging or development purposes.

This tool is particularly helpful when systemd is not functioning correctly or if you want to test specific parts of the application without restarting the entire service.

In short ``ngeneahubctl`` is a component-level tool that complements systemd (but doesn't replace it) by enabling quick debugging, testing, and troubleshooting of individual Ngenea Hub components.

Docker Containers

Docker containers are compact, isolated environments that package an application along with all its libraries and configurations, allowing it to run consistently across different systems without interference.

Ngenea Hub uses Docker containers to run its different components in isolated environments, which improves stability, scalability, and ease of management. Each container serves a specific function within the system:

Core Components

- `ngeneahub_backend`: A middleware service built on Django. It provides the **REST API**, orchestrates jobs and workflows, and acts as the central logic layer connecting the database, UI, and background tasks.
- `ngeneahub_db`: A **PostgreSQL** database container. This third-party service stores all persistent data, including configurations, logs, and metadata for tasks and jobs.
- `ngeneahub_redis`: A **Redis** container acting as a fast in-memory **key-value store**. It serves multiple roles:
    - Acts as a **Celery results backend**, storing temporary task results.
    - In some configurations, serves as the **task queue** (in place of RabbitMQ).
    - Facilitates **direct communication** of task/job states between the Hub and workers.
- `ngeneahub_rabbitmq` (optional): A **RabbitMQ** container used as the **task queue**. Tasks are transmitted to workers via a pub-sub model. In some setups, Redis replaces RabbitMQ for this role.

Task Management & Scheduling

- `celery (internal worker)`: Despite the name, this container is an **internal Celery worker** responsible for asynchronous internal tasks like job refreshes and housekeeping. It does not represent the full Celery engine.
- `celery-dags`: Handles **DAG-based workflow** orchestration. This worker processes callbacks, submits child tasks, and handles the state and results of each step in a DAG.
- `celery-events`: Specialized for receiving streamed events from **discovery services** (e.g., SnapDiff). This replaces large event payloads that might exceed Celery's size limits.
- `celery-results`: Receives streamed task results for tasks that produce large outputs, avoiding Celery payload size issues.
- `celery-beats`: A **Celery scheduler**. Triggers both internal and user-defined periodic tasks (e.g., data syncs, housekeeping routines).
- `celery-monitor`: Monitors the health and availability of Celery workers by listening for **heartbeat** signals and updating the central database accordingly.
- `celery-exporter`: Collects Celery **performance metrics**, such as queue lengths and worker status, and exposes them to Prometheus.
- `task-daemon`: A lightweight internal service responsible for updating task state at the moment a task is picked from the queue. Communicates via **Redis**.

Monitoring & Metrics

- `prometheus`: A metrics storage and query engine, It gathers system and **Celery metrics** (via celery-exporter) and provides an API for data access (not publicly exposed).
- `grafana`: A visualization dashboard that displays system metrics collected by Prometheus. It is accessible via the `/hubmetrics` endpoint.

Hub Task Metrics

The Hub Task Metrics is a monitoring feature provided by Hub. This feature helps you monitor the status of tasks that are currently queued or being processed in the Hub, as well as the status of the workers that execute these tasks.

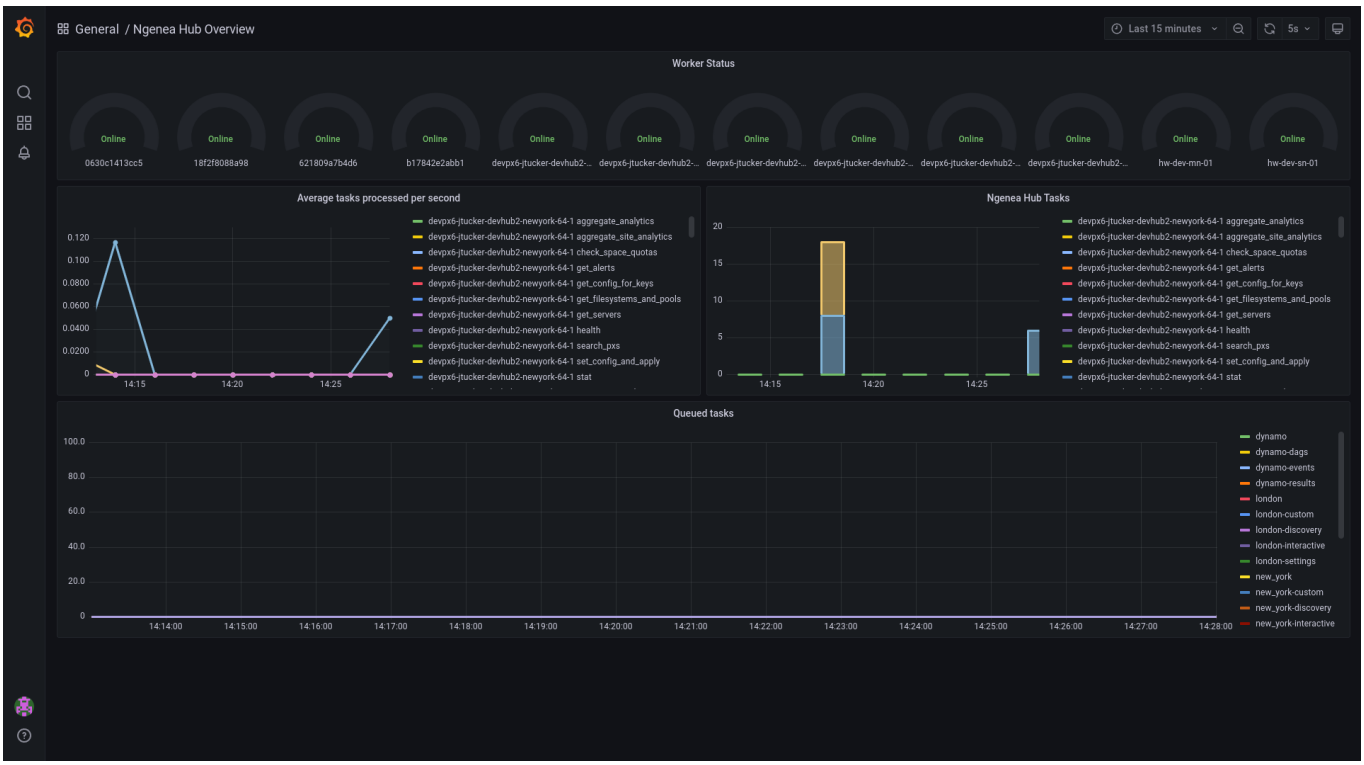
The Grafana Panel

- Grafana is a popular open-source platform for visualizing time-series data like metrics and logs.
- The Hub Task Metrics use Grafana panels to display data visually.
- The panel shows the current queue of tasks waiting to be processed and the status (like idle, busy, or down) of each worker.

You can access the Hub Metrics Grafana Panel by clicking the Hub Metrics button in the main left menu bar. refer to the User Guide.

Tip: The Hub Metrics are only visible to Hub Administrators when Hub is operating in `BROKER_PROVIDER=redis` mode.

Authentication is required to access the metrics page to protect sensitive Hub data, so if you're not logged in, you'll be redirected to the login page or see an error.



Health Endpoints

Health endpoints are special URLs provided by a system to report the current status or health of various components it manages. They are used to monitor and check if everything is working properly.

Component Health

To check the status of all the Ngenea Hub infrastructure components, you can send a `GET request` to the special URL endpoint `/api/health`.

This endpoint asks the Hub to provide information about:

- The health of the **Hub database** (PostgreSQL)
- The health of the **Hub task queue**
- The health of **Redis**

The Hub then tells you the health status of components using the following states:

- *OK*: The service is online and Ngenea Hub can make use of it
- *Unavailable*: The service could not be contacted or reported an error. The response will detail the specific error.

So, by making this `GET` request, you get a quick overview of which components are healthy and which might have issues.

To make a `GET` request, use the following command:

```
curl -X 'GET' \
  'http://myhub/api/health/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <api_key>'
```

Note: You can run the `curl` commands in a **bash environment**. If you're using **Windows**, it's recommended to use a tool like **Git Bash**.

Make sure to replace `<api_key>` with your actual API key.

For more details on generating API keys, refer to the Programming Guide.

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

```
{
    "CeleryPingHealthCheck": "OK",
    "DatabaseBackend": "OK",
    "RabbitMQHealthCheck": "OK",
    "RedisHealthCheck": "Unavailable: Unable to connect to Redis:
Connection Error"
}
```

Site Health

To check the status of all the sites and the nodes (servers or machines) managed by Ngenea Hub, you can send a `GET request` to the special URL endpoint `/api/health/sites`.

This request asks the Hub to provide information about:

- All the sites it knows about.
- The nodes within each site.
- The status of the Hub service itself.

The Hub then tells you the health status based on how many nodes are currently online and working at each site using the following states:

- `ok`: All nodes in the site are working fine.
- `warning`: Some nodes in the site are offline or not responding.
- `critical`: One or more entire sites are completely offline (no nodes are working).

So, by making this `GET` request, you get a quick overview of which sites are healthy and which might have issues.

To make a `GET` request, use the following command:

```
curl -X 'GET' \
  'http://myhub/api/health/sites/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <api_key>'
```

Note: You can run the `curl` commands in a **bash environment**. If you're using **Windows**, it's recommended to use a tool like **Git Bash**.

Make sure to replace `<api_key>` with your actual API key.

For more details on generating API keys, refer to the Programming Guide.

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


```
{
    "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:

```
{
    "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.

Name	Description
`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.
`discovery_options`	A json containing any additional options to pass to the workflow default discovery. Described in more detail below.

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 be “processed” or “modified” more details about this are in the discovery section.	Yes
`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

Name	Description	Required
`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

Name	Description	Required
`queue`	The name of the queue the task should run on. The queue must exist on the site the task will run against. If not provided it will use the queue provided in the workflow call, or the `default` queue if one is not provided	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",
            "queue": "highpriority"
        },
        {
            "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

Name	Description	Required
`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 ◦ `queue` - A list of all queues available on the selected site • `list` - a list of values of any scalar type	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]"
    }
]
```

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": [],
    "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_classes": [],
    "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
`queue`	The queue to run workflow tasks on. The `default` queue will be used if not provided.	String	None
`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",
    "queue": "highpriority",
    "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"
        }
    ]
}
```

Discovery Options

Additional options can be passed to the discovery task by setting `discovery_options` on the workflow. The supported options are those described in Discovery Steps.

Example 12 - Passing options to the discovery

```
{
    "discovery": "recursive",
    "discovery_options": {
        "skip_missing": false
    }
}
```

Discovery options will be used with the workflow default discovery only. If a different discovery is set at runtime, the discovery options will be ignored. Discovery options cannot currently be overwritten at runtime.

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.

Pre and Post Actions

Custom workflows support optional `pre_action` and `post_action` configuration to execute tasks before the discovery action or after the workflow completion.

These can be configured:

- directly on the workflow definition
- at runtime when calling `/api/file/workflow`
- on schedules through `SubscribedWorkflow`

For detailed configuration examples and schedule-specific behaviour, see Pre and Post Actions.

When `pre_action` or `post_action` are provided at runtime, they override any actions already defined on the workflow.

Pre and Post Actions

`pre_action` and `post_action` define additional tasks that run before or after the main workflow execution.

- **pre_action** → Tasks executed *before* the discovery action. These are typically used for preparation steps such as setup, or pre-processing.
- **post_action** → Tasks executed *after* the main workflow completes. These are typically used for cleanup, notifications, or post-processing.

Configure Pre and Post Actions on a Workflow

`pre_action` and `post_action` can be specified during workflow creation or patched onto an existing workflow with `filter_rules` being the workflows primary actions.

Example

```
{
    "pre_action": [
        {
            "name": "dynamo.custom.slack_notify"
        }
    ],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.tasks.migrate",
                    "lock_level": "*lock_level"
                }
            ],
            "description": "Migrates a file off from a given path"
        }
    ],
    "post_action": [
        {
            "name": "dynamo.tasks.templated_email",
            "email_from": "ngeneahub-noreply@localhost",
            "email_to": ["root@localhost"],
        }
    ]
}
```

Stream Input Behaviour

For DAG Workflows, The `stream_input` flag controls how input paths are passed to actions:

- `true` - The task receives streamed path URL references and auth_token instead of raw paths
- `false` - The task receives the original input paths directly

When `stream_input=True`:

- The target task **must support** `expand_paths` to correctly resolve streamed inputs

```
    from arcapix.dynamo.server.utils.inputs import expand_paths

    @app.task(bind=True, base=ChainedTask, name="dynamo.custom.sl
ack_notify")
    def slack_notify(self, jobid, paths, *args, **kwargs):
        input_paths = expand_paths(paths)
```

If the task does not support `expand_paths`, it may fail or not resolve input paths correctly.

Note: The `post_action` supports streamed path URL references by default for SnapDiff workflows.

Example

```
{
    "name": "test_wf_pre_post",
    "label": "test wf",
    "icon_classes": ["hurricane-01"],
    "discovery": "recursive",
    "enabled": True,
    "visible": True,
    "schedule_only": False,
    "external_only": False,
    "fields": [
        {
            "name": "lock_level",
            "type": "choices",
            "label": "Locking mode",
            "choices": [{"label": "Per block (Implicit)", "value":
"implicit"}],
            "default": "implicit",
        }
    ],
    "pre_action": [
        {
            "name": "dynamo.custom.slack_notify",
        }
    ],
    "post_action": [
        {
            "name": "dynamo.tasks.templated_email",
            "email_from": "ngeneahub-noreply@localhost",
            "email_to": ["root@localhost"],
        }
    ],
    "filter_rules": [
```

```
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.tasks.migrate",
                    "lock_level": "*lock_level",
                    "stream_input": True,
                }
            ],
        }
    ],
}
```

Update or Remove Pre/Post Actions

To update `pre_action` and `post_action`, send a PATCH request with the updated payload.

To remove them entirely, set the fields to `null`.

Example

```
{
    "pre_action": null,
    "post_action": null
}
```

Configure Pre and Post Actions at Runtime

To configure `pre_action` and `post_action` at runtime, include them in the payload when calling the API `/api/file/workflow`. These will override any pre/post actions already defined on the workflow.

Example

```
{
    "discovery": "recursive",
    "paths": ["/mmfs1/data/aws/"],
    "site": "siteA",
    "workflow": "test_wf_pre_post",
    "pre_action": [
        {
            "name": "dynamo.custom.slack_notify",
            "hook": "https://hooks.slack.com/services/",
            "message": "Job {jobid} completed on site {site} with
paths {paths}"
        }
    ],
    "post_action": [
        {
            "name": "dynamo.tasks.templated_email",
            "email_from": "ngeneahub-noreply@localhost",
            "email_to": ["root@localhost"]
```

```
        }
    ]
}
```

Disable Runtime Pre/Post Actions

To disable runtime `pre_action` and `post_action`, pass `null` for those fields.

Example

```
{
    "discovery": "recursive",
    "paths": ["/mmfs1/data/aws/"],
    "site": "site1",
    "workflow": "test_wf_pre_post",
    "pre_action": null,
    "post_action": null
}
```

Runtime Fields

`pre_action` and `post_action` support runtime field references.

For example, if:

```
"site": "*destinationsite"
```

is specified, the action will run on the `destinationsite` value provided at runtime.

Example

```
{
    "fields": [
        {
            "name": "destinationsite",
            "type": "enum[site]",
            "label": "Destination Site"
        }
    ],
    "pre_action": [
        {
            "name": "dynamo.tasks.migrate",
            "site": "*destinationsite"
        }
    ]
}
```

Configure Pre and Post Actions on a Schedule

To configure `pre_action` and `post_action` for a schedule, include them in the `SubscribedWorkflow` api `/api/schedules/{id}/workflows/`

Note: Any `pre_action` or `post_action` already defined on the workflow itself will be ignored for schedules. The actions configured on the `subscribed_workflow` take precedence.

Example

```
{
    "site": "siteA",
    "workflow": "site_sync",
    "fields": {
        "destinationsite": "siteB",
        "destinationqueue": "default",
        "sync_preference": "newest"
    },
    "queue": "default",
    "pre_action": [
        {
            "name": "dynamo.custom.slack_notify",
            "hook": "https://hooks.slack.com/services/",
            "message": "Job {jobid} completed on site {site} with
paths {paths}"
        }
    ],
    "post_action": [
        {
            "name": "dynamo.tasks.templated_email",
            "email_from": "ngeneahub-noreply@localhost",
            "email_to": ["root@localhost"]
        }
    ]
}
```

Schedule Behaviour

SnapDiff Workflows

For SnapDiff workflows, the `pre_action` and `post_action` of the first `SubscribedWorkflow` are respected when the schedule is triggered.

DAG Workflows

For DAG workflows, the `pre_action` and `post_action` are executed for each workflow individually.

Bidirectional Sync Schedules

For bidirectional sync schedules, the `post_action` is triggered only after the second rotate task completes.

Jobs

Housekeeping

Job details are kept for a time after job completion. During this time, they can be viewed in the UI.

But older jobs are periodically culled from the database, to keep it to a manageable size. The housekeeping process runs once a day, and removes results for any job that completed more than 90 days ago (by default). A different 'time-to-live' (TTL) can be set using the `jobs_ttl` configuration - see Configuration for more information.

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

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

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.

Refer to the Global Settings to configure search backend.

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}`
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",
      },
      "proxies": {
          "_thumbnail": "/media/123/456/789/123456789012345.png"
      }
    }
  ],
  "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.

Proxies

The `proxies` field on each item contains links to proxies of the file, such as thumbnails and previews.

These will only be available if the backend is `pixstor_search`, and PixStor Search itself has been configured and ingested in 'search plus' mode.

The links are relative urls, which must be combined with the `public_url` from the corresponding `/sites` endpoint to fetch the actual proxy files, e.g. `https://pixstor-site1/media/...`.

Typical proxies may include:

- `_thumbnail` - a 150x150 px thumbnail image
- `image.preview` - a 400x300 px preview image
- `video.preview` - a 400x300 px, lower resolution version of a video
- `audio.preview` - a downscaled version of an audio file

Proxies may not be available for all file types, depending on which search plugins have been enabled and ingested.

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`	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.

name	description
`group_by_name`	When this url parameter is 'true' results are merged based on matching name, e.g. `?group_by_name=true`

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.

Results grouped by name

To group search results by name, add the url parameter `group_by_name=true`.

This will return results from any of the requested sites. Results within a group may belong to different directories.

```
$ curl 'http://example.com/api/search/1/?group_by_name=true' -H "
Authorization: Api-Key $TOKEN"
{
  "count": 60,
  "next": "http://testserver/api/search/1/?
group_by_name=true&page=2",
  "previous": "http://testserver/api/search/1/?
group_by_name=true",
  "more_results": false,
  "items": [
    {"test-10": [
      {
        "site": "testsite",
        "path": "/mmfs1/data",
        "metadata": {...}
      },
      {
        "site": "anothersite",
        "metadata": {...}
      }
    ]
    },
    {"test-11": [
      {
        "site": "testsite",
        "path": "/mmfs1/data",
        "metadata": {...}
      },
      {
        "site": "anothersite",
        "path": "/mmfs1/archive",
        "metadata": {...}
      }
    ]
    },
    {"test-19": [
      {
```

```
        "site": "testsite",
        "path": "/mmfs1/data",
        "metadata": {...}
      },
      {
        "site": "anothersite",
        "path": "/mmfs1/data",
        "metadata": {...}
      },
      ...
    ]
    }
  ],
  "complete": true,
  "metadata_fields": null,
  "errors": {}
}
```

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 "Authorizati
on: 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.

Bandwidth configuration in the UI

The bandwidth setting can be updated both via the REST API and directly through the UI. Any changes to the `bandwidth` attribute on a `Site` instance will take effect immediately.

The UI on the Site details page now includes a field for setting the bandwidth limit, displayed 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
jobs_ttl	How long job details should be stored after job completion, in days.	90

Name	Description	Default
search_backend	Backend to use when performing searches. Currently supported backends: `analytics`, `pixstor_search`.	`pixstor_sea`
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 more storage space. Fetching fewer results may lead to some files being missing. Note, some backends have a hard limit of 10,000 results.	200
analytics_timeout	How long to wait for results from search results when configured to use analytics, in seconds.	10
snapshot_create_delete_retry_timeout	Time in seconds after which workers will give up retrying snapshot create and delete operations. These occur in the `dynamo.tasks.snapdiff` and `dynamo.tasks.rotate_snapshot` tasks. The default 1000s will give 4 or more retries. Set to 0 to disable retries.	1000
stat_timeout	How long to wait for results from the `/api/file/` endpoint, in seconds.	10
stat_refresh_period	How long a files details will be retained as a cache within the file browser, in seconds.	10
task_invalidation_timeout	How long in minutes for a task in the STARTED state will wait before being invalidated.	360
snapdiff_stream_timeout	Idle timeout when waiting for delete and move tasks to complete during snapdiff workflows, in minutes.	10
force_local_managed_users	Allow local NAS users to be managed on AD joined sites.	False
custom_statistics_tasks	A list containing the names of custom tasks that add their number of reported files to the job total, it will do so regardless of when these tasks are executed within a workflow.	[“dynamo.cu “dynamo.cu
maximum_external_results	Maximum number of items to retrieve per page from an external target scan.	100

Name	Description	Default
node_health_last_heartbeat_interval	The maximum allowed time (in seconds) between the last recorded heartbeat and the current time for a node to be considered online. If the node's last heartbeat occurred more than this interval ago, the node will be marked as offline.	120
initial_uid	Minimum uid value for the nas user that can be configured	100000
initial_gid	Minimum gid value for the nas group that can be configured	100000
schedules_enabled	Allow the processing of automated tasks (Policies and Scheduled Workflows).	True
health_timeout	Maximum wait time in seconds for a health check to determine available sites.	1
salt_task_timeout	How long to wait for settings tasks to return their results, in seconds.	60
default_space_location	Path of a directory, relative to the file system root, to use as the default location for space creation	data
keycloak_jwks_timeout	Number of seconds to cache SSO tokens between Hub's SSO services	3600

Note: If the file details are not returned for large fileset, increase the default value of `stat_refresh_period` in the configuration tab of hub admin page http:///:/admin or do a patch on configuration API.

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 'A
ccept: application/json' -H "Authorization: Bearer $JWT_ACCESS_TO
KEN" -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 evaluated on the node(s) on which the site worker is running.	`localhost:9200`

Name	Description	Default
pixstor_search_url	URL used to interact with PixStor Search when `search_backend` is set to `pixstor_search` (see **Global Configurations** above). The URL is evaluated on the node(s) on which the site worker is running.	`https://localhost/`
public_url	Public URL that can be used to reach this site. Typically this will be the hostname or external IP address of the site management node.	*not set*
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
enable_auto_file_batch_sizing	If Dynamic file batching should be enabled for this site. See **Dynamic File Batching** below	`True`
soft_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*
iscanThreads	Defines the number of threads that can be used during snapdiff operations.	*not set*
iscanBuckets	Defines the number of buckets that can be used during snapdiff operations.	*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).

Dynamic File Batching

The Dynamic File Batching feature is designed to improve the processing of tasks with large sets of files by adjusting batch size based on the total number of files being processed from using a predefined set of ranges.

By enabling the auto file batch sizing configuration option, the system automatically adjusts file batch sizes to optimize resource utilization and enhance performance.

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,
```

```
    ...
}
```

Certificate Lifetime

The certificate lifetime refers to the duration for which a certificate is valid. You can view and modify this duration through the `/api/pki/lifetime/1/` endpoint.

Viewing the Certificate Lifetime

To view the current certificate lifetime, you can send a `GET` request to the `/api/pki/lifetime/1/` endpoint. This will return a JSON response with the current lifetime in days.

```
{
  "days": 2147483647,
  "updated_at": "2024-04-12T09:19:43.732Z",
  "updated_by": 0
}
```

Modifying the Certificate Lifetime

To modify the certificate lifetime, you can send a `PATCH` request to the `/api/pki/lifetime/1/` endpoint. The request body should be a JSON object with a `days` field specifying the new lifetime in days.

Please note that only administrators and users with the `filebrowser.change_certificatelifetime` permission can modify the certificate lifetime.

Example:

```
curl -X PATCH -H "Content-Type: application/json" -d '{"days":
365}' https://your-domain.com/api/pki/lifetime/1/
```

In this example, we are setting the certificate lifetime to 365 days.

Remember to replace the `days` value with the desired lifetime in days.

Summary

The `/api/pki/lifetime/1/` endpoint allows you to view and modify the certificate lifetime. Remember that to use the changed lifetime in the certificates, then Ngenea Hub needs to be restarted.

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

There are currently no feature flags.

Note: The feature flags in the examples below are for demonstration purposes only. They may not exist or may have been removed.

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 an 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.

Iris

Spaces API: Iris Integration and User Filtering

The `/api/spaces/` endpoint now supports filtering spaces by username to determine what spaces a user can access. This feature allows Hub Admins to query a user's space access based on their group memberships and Iris permissions.

Additionally, spaces and their associated sites include `iris` information to indicate whether they are Versity S3 presented, and what level of access the user has on each site.

Iris state

- **enabled: true** - The space (and its associated sites) is Versity S3 presented.
- **enabled: false** - The space is not Versity S3 presented.

Permissions and Group-Based Access

Iris permissions are defined `per space per site`, and are granted to `groups`. User Permissions are derived from the groups the user belongs to.

The possible permissions are

1. `readonly (default)` - A user is granted readonly access if they belong only to groups with readonly permission (or no groups at all). Readonly users can list spaces but cannot upload/download to buckets.
2. `readwrite` - A user is granted readwrite access if any of their groups have readwrite permission for a space-site. Readwrite users can fully upload/ download to buckets.

Note: If a user belongs to multiple groups with different permissions on the same space-site, the highest permission (readwrite) is applied.

Permissions are advisory — they do not enforce access within Hub, but provided information to Iris.

REST API

```
GET /api/spaces/?username=<target_username>
```

- The authenticated user must have admin privileges.
- If `username` is not provided, the API returns spaces for the authenticated user by default and does not include Iris permissions.

Example for Iris Response Payload Structure

When Iris is enabled:

```
"iris": {
    "enabled": true,
    "user_permission": "readwrite"
}
```

When Iris is not enabled, `user_permission` is omitted from the payload.

```
"iris": {
    "enabled": false
}
```

Example Response with Iris User Permissions.

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "url": "http://10.201.2.195:8000/api/spaces/2/",
            "id": 2,
            "uuid": "67d887c4-50c6-40f1-95f5-f1c346b12494",
            "name": "space-67",
            "mountpoint": "/mmfs1/data/space-67",
            "sites": [
                {
                    "id": 2,
                    "name": "siteA",
                    "label": "siteA",
                    "pool": "sas1",
                    "usage": null,
                    "iris": {
```

```
                        "enabled": true,
                        "user_permission": "readwrite"
                    }
                },
            ],
            "created": "2025-06-17T11:05:41.815258Z",
            "size": 10737418240,
            "snapshot_schedule": {
                "id": 1,
                "frequency": "1 00:00:00",
                "time": "00:05:00",
                "duration": "7 00:00:00",
                "space": 2
            },
            "on_sites": {
                "siteA": {
                    "name": "space-67",
                    "size": 10737418240,
                    "present": true
                }
            },
            "immutable": false,
            "editable": true,
            "color": [191, 99, 64],
            "permission_mode": null,
            "relationships": {
                "shares": [],
                "policies": [],
                "schedules": []
            },
            "nested_shares": {
                "samba": [],
                "nfs": []
            },
            "is_ready": true
        }
    ]
}
```


Example Response When User Has No Accessible Spaces


```
{
    "count": 0,
    "next": null,
    "previous": null,
    "results": []
}
```


Summary

1. Returns spaces for the authenticated user if username is not provided.
2. Returns 200 OK with no spaces if the target user does not exist.

3. Returns 200 OK with no spaces if the user exists but has no group memberships.
4. Returns 200 OK with no spaces if user's groups do not have view_space permission.
5. Returns all spaces where user's groups have view_space permission.
6. Supports additional filtering by site_id when both username and site_id are provided.
7. Returns all spaces if the username is omitted or username=.

Groups API: Iris Permissions

The `/api/groups/` endpoint supports assigning and retrieving Iris permissions for spaces.

Permissions are defined `per space per site`, and groups cannot have duplicate permissions on the same space-site combination.

Permissions currently support two levels:

- readonly (default)
- readwrite

If a user is a member of multiple groups with different permissions on the same space-site, the highest level (readwrite) is applied.

Example: View Group with Iris Permissions

```
GET /api/groups/1/
{
  "id": 1,
  "name": "iris-admin",
  "users": [
    {
      "username": "bob"
    }
  ],
  "iris_permissions": [
    {
      "space": "Project1",
      "site": "london",
      "permission": "readwrite"
    },
    {
      "space": "Project1",
      "site": "paris",
      "permission": "readonly"
    }
  ]
}
```

Example: Create Group with Iris Permissions

```
POST /api/groups/
{
  "name": "london-admin",
  "iris_permissions": [
    {
      "space": "Project1",
      "site": "london",
      "permission": "readwrite"
    }
  ]
}
```

Example: Update Group Permissions

```
PATCH /api/groups/1/
{
  "iris_permissions": [
    {
      "space": "Project1",
      "site": "london",
      "permission": "readwrite"
    },
    {
      "space": "Project1",
      "site": "paris",
      "permission": "readonly"
    }
  ]
}
```

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.

Snapshot Rotation

By default, snapdiff snapshots will always rotate, even if an error occurs during sync.

Traditionally, snapshots are rolled-back on error. However, this can lead to changes being replayed inappropriately, leading to further errors and conflicts. Because of this, the default behaviour was changed to always rotate.

The downside to this approach is that some changes may be missed by sync. For example, if an network issues prevent a file from being synced, that file will not be re-synced unless or until it changes again. Note, however, there are retries within a sync run to mitigate such temporary issues.

If a sync job does fail, it will be necessary to determine the source of the error and manually resolve any issues. The job details will report on which paths failed overall. Looking at the individual task details will give more information on where and why a specific path failed.

To disable this behaviour, so that snapshots will rollback on error, set the runtime field `snapdiff_rotate_on_error` to False

> **Note:** If the sync is run within a schedule, it will use the first subscribed workflow's snapdiff_rotate_on_error value. This does not apply to bidirectional_sync as it can only have one subscribed workflow.

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.name.<fileset_name>.id.<fileset_id>`

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.

As discussed above, this is the default behaviour. Note that this may result in some file changes not being synced. For a safer option, see **Re-sync all** above.

The following steps can be used as a one-off when the default rotate-on-error function has been disabled.

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.name.<fileset_name>.id.<fileset_id>`

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.name.<fileset_name>.id.<fileset_id>.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`, `none`. 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`, `none`. Default=`implicit`

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`, `none`. Default=`implicit`

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
- `destinationqueue` (string): queue on the destination site to run the task on
- `hydrate` (bool): hydrate files on the destination site

unmanage

Removes a correspondence between files and storage endpoints.

discovery: `recursive`

steps:

- dynamo.tasks.unmanage

fields:

- `no_flock` (optional): Disable using lock files. Sets lock level to `implicit` if it is not set. Conflicts with `partial` lock level.

- `lock_level` (choice): Filesystem locking level for ngenea operations. One of: `partial`, `implicit`, `none`. Default=`implicit`
- `storage_key` (pattern): Removes the link between files and storage endpoints, optionally filtering by keys that match a specified extended glob pattern (multiple patterns can be separated by `|`). By default, it removes all keys.

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
- `destinationqueue` (string): queue on the destination site to run the task on
- `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
  - `abort_missing`: 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
  - `abort_missing`: 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

import_to_site_hydrated

This workflow is designed to import fully hydrated remote objects from external ngenea targets to the local filesystem.

discovery: `null`

steps:

- dynamo.tasks.import_files_from_external_target
    - `hydrate`: True

fields:

- `endpoint` (string): Name of the ngenea target configuration
- `location` (string): An absolute path for where to place the file/folder in the local filesystem, this can be a different path than the remote location. Defaults to `None`
- `lock_level` (string): Filesystem locking level for ngenea operations. One of: `partial`, `implicit`, `none`. Default=`implicit`

import_to_site_dehydrated

This workflow is designed to import stubbed remote objects from external ngenea targets to the local filesystem.

discovery: `null`

steps:

- dynamo.tasks.import_files_from_external_target
    - `hydrate`: False

fields:

- `endpoint` (string): Name of the ngenea target configuration
- `location` (string): An absolute path for where to place the file/folder in the local filesystem, this can be a different path than the remote location. Defaults to `None`
- `lock_level` (string): Filesystem locking level for ngenea operations. One of: `partial`, `implicit`, `none`. Default=`implicit`

Hidden Workflows

This section documents all the workflows which come installed in Ngenea Hub by default that are not provided in the UI. Some of these are intended for advanced users making use of the API, and some are used for GPFS policy automation.

transparent_recall

This is the workflow that will be performed on dehydrated files on the system when accessed, if the policy is configured to make use of Ngenea Hub within its Transparent Recall policy.

Typically, all of these Transparent Recalls are placed into a single job within Ngenea Hub.

steps:

- dynamo.tasks.recall
  - `lock_level`: “partial”

reverse_stub

This workflow is designed to create stubs files or fully hydrated files using remote content on a given site without the need for an existing file on the site for a given path. Due to this requiring the data to be within the cloud and not present on the filesystem, it requires the paths to target cloud targets through the API.

steps:

- dynamo.tasks.reverse_stub

fields:

- `hydrate` (bool): If the file should be fully hydrated during the recall operation
- `overwrite` (bool): If the operation should overwrite a local file that already exists

delete_file

This workflow deletes one or more file/folder paths from the filesystem.

steps:

- dynamo.tasks.delete_paths_from_gpfs

fields:

- `recursive` (bool): If non-empty directories should be deleted. if recursive is `false` additional jobs will be needed to delete empty directories. Defaults to `false`

iris_ingest

This workflow generates a file created notification for Iris, and extracts some file metadata.

steps:

- `dynamo.tasks.iris.filenotify`
- `dynamo.tasks.iris.mediainfo`
    - `extra_flags`: `["--File_TestContinuousFileNames=0"]`
- `dynamo.tasks.iris.exiftool`

The `extra_flags` parameter allows for passing `mediainfo` command line flags and should not be changed unless advised by support.

Discovery Steps

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

`dynamo.tasks.recursive_navigate` - `recursive` - Function: discovery

Navigates down any folder tree provided using the initial paths as the base of this navigation. It will process every discovered item and can have its behaviour defined using type rules using `file|link|folder` or action on all items with `all` as its type and state.

Upon discovering non-empty directories, it will create another instance of this task to navigate down that directory while still processing the current directory. Once a batch of items has been found it will action the appropriate rule on them.

Specifically within a recursive action, to process empty directories there must exist a rule within the workflow that explicitly handles the `empty_directory` type to correctly process them. Without this action empty directories will not be processed even with an existing `directory` rule.

Argument	Type	Default	Description
`skip_missing`	`bool`	`False`	Allows the processing of any files directly provided to the discovery step regardless of it being on the filesystem.

dynamo.tasks.recursive_action - recursive(deprecated) - Function: discovery

Navigates down any folder tree provided to it and actions any rule defined with `all` as its type and state on all found files.

Argument	Type	Default	Description
`skip_missing`	`bool`	`False`	Allows the processing of any files directly provided to the discovery step regardless of it being on the filesystem.

dynamo.tasks.snapdiff - snapdiff - Function: discovery

For use only with a GPFS Independent Fileset, it will create a GPFS snapshot and it will then process the list of differences between the time of the initial run and subsequent runs. On the first run of this, it will ingest all the files within that Fileset.

Argument	Type	Default	Description
`skip_old_ctimes`	`bool`	`False`	Skips any files within the snapdiff difference list if the ctime of the file is older than the oldest snapshot.
`condense_moves`	`bool`	`True`	If a directory is moved, this will condense all the move operations into a single operation for move based tasks, otherwise it will action against every effected file.
`extra_fields`	`list`	none	Additional metadata to fetch for discovered files. When available, this will be included in the path objects passed to tasks, under the `additional_metadata` field

Available `extra_fields`:

- `inode`: file inode number
- `generation`: file inode generation number
- `is_metadata_only_update`: for `updated` files, this field will be `True` if only the file's metadata was changed and `False` if the file contents may have been changed (based on file `mtime`).

Workflow Steps

The following are all the currently supported steps in Ngenea Hub. All of these steps are ran on the Ngenea Worker provided site on its respective function queue when running a workflow. See Custom Workflows for guidance on how to use these steps in your own workflows.

Each task alongside its name has its function queue detailed, these functions can be controlled through the Ngenea Worker configuration

dynamo.tasks.migrate - Function: Worker

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
fail_on_mismatch	bool	False	fail a file migration if a remote object exists but has different hash or metadata. In that case, the task errors
lock_level	string	implicit	Defined the locking mode that ngenea will use when performing the migrate
endpoint	string		specify the endpoint to migrate
abort_missing	bool	False	allow the migrate task to make any missing files end up in the “aborted” state instead of “failed”
migrate_offline_files	bool	False	If set to True, the migrate task will process offline files in the same manner as regular online files. If set to False, the migrate task will process the offline files using --sync-metadata
sub_batch	int	optional	The number of paths to process in each sequential chunk. When set, the task will be split and executed in smaller batches of this size. By default, sub_batch=None processes all paths in a single batch

dynamo.tasks.recall - Function: Worker

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

Argument	Type	Default	Description
`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`		When a file is recalled, it uses this GID if one is not set on the remote object
`update_atime`	`bool`	`false`	When a files is recalled, update its access time (atime) to 'now'
`update_mtime`	`bool`	`false`	When a files is recalled, update its modification time (mtime) to 'now'
`delete_remote`	`bool`	`false`	If set, when a file is recalled, it deletes the file in the remote location
`sub_batch`	`int`	`optional`	The number of paths to process in each sequential chunk. When set, the task will be split and executed in smaller batches of this size. By default, `sub_batch=None` processes all paths in a single batch

`dynamo.tasks.reverse_stub` - Function: Worker

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, except files with only metadata changes.
`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

Argument	Type	Default	Description
`update_atime`	`bool`	`false`	When a files is recalled, update its access time (atime) to ‘now’
`update_mtime`	`bool`	`false`	When a files is recalled, update its modification time (mtime) to ‘now’
`conflict_preference`	`string`	`None`	Dictates what state the local file should be to pass the check. Options are “newest” 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.
`sub_batch`	`int`	`optional`	The number of paths to process in each sequential chunk. When set, the task will be split and executed in smaller batches of this size. By default, `sub_batch=None` processes all paths in a single batch

`dynamo.tasks.unmanage` - Function: Worker

Removes a correspondence between files and storage endpoints.

Argument	Type	Default	Description
`lock_level`	`string`	`implicit`	Defines the locking level ngenea will use during the unmanage
`no_flock`	`bool`	`False`	Disable using lock files during the unmanage. Conflicts with `partial` lock level
`storage_key`	`string`	`optional`	Only remove keys that match an extended glob pattern. Multiple patterns can be separated by `|`.
`sub_batch`	`int`	`optional`	The number of paths to process in each sequential chunk. When set, the task will be split and executed in smaller batches of this size. By default, `sub_batch=None` processes all paths in a single batch

`dynamo.tasks.ngenea_sync_metadata` - Function: Worker

Sync the local ngenea metadata on a file with the remote target using Ngenea.

Argument	Type	Default	Description
`skip_hash`	`bool`	`False`	If the sync should skip checking the hash of the file
`endpoint`	`string`		Specify which endpoint(site) to recall from
`default_uid`	`string`		When a file is synced, it uses this UID if one is not set on the remote object

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

`dynamo.tasks.delete_paths_from_gpfs` - Function: Worker

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. It is important to note that the recursive behaviour of removing the entire directory tree will not apply to filesets or if the target directory contains files or directories that are restricted from being deleted such as `snapshots`, in such cases the task will silently ignore those files or directories and report the task as successful.

`dynamo.tasks.check_sync_state` - Function: Worker

Uses a previously executed `stat call<stat_paths>` to checks a provided site against the calling sites to ensure that the local file is in a specified state compared to the provided stat call.

If used within a non `snapdiff discovery<snapdiff>` workflow, it will *not* prepend a matching `stat call<stat_paths>` task and must be explicitly defined in the workflow itself, otherwise this task will not execute correctly.

When used within a `snapdiff discovery<snapdiff>` using this task will also perform `dynamo.tasks.stat_paths` on the provided site before execution without the inclusion of the task within the workflow.

Argument	Type	Default	Description
`sync_preference`	`string`	`ignore`	Dictates what state the local file should be to pass the check. Options are “newest” 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.
`site`	`string`		The target site to compare
`abort_outdated`	`bool`	`False`	If this bool is set files that do not need to be executed will be marked as aborted as opposed to skipped
`hash_includes_acl`	`bool`	`False`	If this bool is set the metadata comparison for directories will also compare Access Control Lists

dynamo.tasks.stat_paths - Function: Worker

Performs a GPFS `stat` on all provided paths and returns a key value map of all paths and their respective stat results.

This task does *not* follow the standard chain formatting of Ngenea Hub workflows, it can only be chained directly onto `check_sync_state<check_sync_task>` tasks or called without proceeding tasks. If chained to other tasks, the following tasks will not execute as expected.

This task is automatically pre-run within `snapdiff discovery<snapdiff>` and `bidirectional_sync<bidirectionalsync_schedule>` based workflows when the `check_sync_state<check_sync_task>` task is used.

Argument	Type	Default	Description
`show_details`	`string`		Display child information for a given path

dynamo.tasks.move_paths_on_gpfs - Function: Worker

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

This task moves files in two steps (via an intermediate temporary location), to avoid move conflicts (for example, this task correctly handles cases where files are 'swapped': `fileA` moved to `fileB`, and `fileB` moved to `fileA`).

Argument	Type	Default	Description
`delete_remote_xattrs`	`bool`	`False`	If set, after a file has been moved all remote location xattrs will be removed
`source_missing_signature`	`json`	`null`	A signature to send any paths to where the source is missing. If this isn't set, a missing source is treated as a failure.
`target_max_age`	`int`	`null`	If the target file exists, it is overwritten by default. If this optional parameter is set, the target has to have a ctime <= this timestamp for the move to proceed. Otherwise, the source file is removed. Given in seconds since epoch. Primarily used for sync workflows.

dynamo.tasks.one_step_move_paths_on_gpfs - Function: Worker

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

This task is not used in the default Ngenea Hub workflows. It is less robust than `dynamo.tasks.move_paths_on_gpfs`, because it moves files directly from source to their new location (in one step). So for example, trying to 'swap' files (`fileA` moved to `fileB`, and `fileB` moved to `fileA`) with this task will effectively result in one of these files being deleted on the target site.

However, this task is for cases where doing moves in two steps is not an acceptable option.

Argument	Type	Default	Description
`delete_remote_xattrs`	`bool`	`False`	If set, after a file has been moved all remote location xattrs will be removed
`source_missing_signature`	`json`	`null`	A signature to send any paths to where the source is missing. If this isn't set, a missing source is treated as a failure.
`target_max_age`	`int`	`null`	If the target file exists, it is overwritten by default. If this optional parameter is set, the target has to have a ctime <= this timestamp for the move to proceed. Otherwise, the source file is removed. Given in seconds since epoch. Primarily used for sync workflows.

`dynamo.tasks.remove_location_xattrs_for_moved` - Function: Worker

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

This step takes no additional arguments.

`dynamo.tasks.delete_from_target` - Function: Worker

Deletes file system objects from their related remote storage target using provided paths.

Argument	Type	Default	Description
`endpoint`	`string`		specify which endpoint(site) to recall from

Note: when chained, this task will handle both 'processed' and 'skipped' paths from the parent task.

`dynamo.tasks.move_in_cloud` - Function: Worker

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` - Function: Worker

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` - Function: Worker

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.

`dynamo.tasks.import_files_from_external_target` - Function: Worker

Recalls a list of files from an external target to a predefined local object using Ngenea. The input files must be a list of remote object paths not local file paths. (eg, `projects/`)

Argument	Type	Default	Description
`endpoint`	`string`		Specify which endpoint(ngenea target name) to recall from
`lock_level`	`string`	`implicit`	Defines the locking level ngenea will use during the recall
`location`	`string`	`null`	Specify an absolute path for where to place the file/folder in the local filesystem
`hydrate`	`bool`	`false`	If the remote object should be premigrated instead of a regular stub
`extra_flags`	`list`		List of extra arguments that can be passed to recall command
`skip_hash`	`bool`	`false`	If the recall should skip checking the hash of the file
`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
`update_atime`	`bool`	`false`	When a files is recalled, update its access time (atime) to 'now'
`update_mtime`	`bool`	`false`	When a files is recalled, update its modification time (mtime) to 'now'

dynamo.tasks.copy_files_from_external_target

Copies a list of files from an external target to a predefined local location using Ngenea. The input files should be provided as a list of remote object paths, not local file paths (e.g., projects/). Once copied, these files are no longer associated with the target.

Argument	Type	Default	Description
`endpoint`	`string`		Specify which endpoint(ngenea target name) to recall from
`lock_level`	`string`	`implicit`	Defines the locking level ngenea will use during the recall
`location`	`string`	`null`	Specify an absolute path for where to place the file/folder in the local filesystem
`default_uid`	`string`	`requires ignore_metadata to be: True, else ignored`	When a file is recalled, it uses this UID if one is not set on the remote object
`default_gid`	`string`	`requires ignore_metadata to be: True, else ignored`	When a file is recalled, it uses this GID if one is not set on the remote object
`extra_flags`	`list`		List of extra arguments that can be passed to recall command
`ignore_metadata`	`bool`	`false`	Do not read user, group and timestamps etc, from data to be copied

dynamo.tasks.import_bytes - Function: Worker

Recalls specific byte ranges of files from an external target to a predefined local location using Ngenea. The input files should be provided as a list of remote object paths (e.g., projects/), and the recall operation will be based on the byte-level data of these files, rather than entire files.

Argument	Type	Default	Description
`endpoint`	`string`		Specify which endpoint(ngenea target name) to recall from
`lock_level`	`string`	`implicit`	Defines the locking level ngenea will use during the recall
`location`	`string`	`null`	Specify an absolute path for where to place the file/folder in the local filesystem

Argument	Type	Default	Description
`remote_offset_start`	`int`	`0`	Beginning offset of a segment of a remote object to download
`remote_offset_end`	`int`	`remote object size`	Ending offset of a segment of a remote object to download
`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
`default_mode_file`	`int`		Specify mode bits (in octal format) to set for stubbed/premigrated files if there are no mode bits associated with remote objects
`default_mode_dir`	`int`		Specify mode bits (in octal format) to set for directories created locally if there are no mode bits associated with remote objects

Warning: This task is incompatible with recursive discovery.

`dynamo.tasks.recall_bytes` - Function: Worker

Recalls specific byte ranges of files from local stubbed paths to a predefined local location using Ngenea. The input should be a list of local stubbed file paths, and the recall will only retrieve the byte-level data of those files, rather than entire files.

Argument	Type	Default	Description
`lock_level`	`string`	`implicit`	Defines the locking level ngenea will use during the recall
`location`	`string`	`null`	Specify an absolute path for where to place the file/folder in the local filesystem
`remote_offset_start`	`int`	`0`	Beginning offset of a segment of a remote object to download
`remote_offset_end`	`int`	`remote object size`	Ending offset of a segment of a remote object to download
`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
`default_mode_file`	`int`		Specify mode bits (in octal format) to set for stubbed/premigrated files if there are no mode bits associated with remote objects

Argument	Type	Default	Description
`default_mode_dir`	`int`		Specify mode bits (in octal format) to set for directories created locally if there are no mode bits associated with remote objects

Warning: This task is incompatible with recursive discovery.

Automated Tasks

All detailed tasks below are run on automated schedules defined by the hub itself, some of these can have their schedules adjusted and thresholds adjusted.

Every minute

`alerts.tasks.fetch_alerts`

Submits the task `dynamo.tasks.get_alerts` to retrieve and sync the alerts on all sites with the Ngenea Hub's internal database. This is run on the `discovery` function of the Ngenea Worker. The returning task that stores the alerts in the database runs within the internal `celery` queue within Ngenea Hub.

The schedule of this task can be controlled by using the SYNC_ALERTS_INTERVAL config setting, any provided value must comply with cron syntax.

Every 30 minutes

`dynamocore.tasks.admin.load_storagepools`

Fetches the GPFS storage pools from all active sites registered within Ngenea Hub and stores them within the database. Additionally, it will also create the default filesystem root spaces if they don't exist for a site.

The schedule of this task can be controlled by using the REFRESH_SITE_ANALYTICS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.space_site.send_check_space_site_quota`

Sends a task to the Ngenea Hub's internal `celery` queue to collect all spaces within every site named `dynamocore.tasks.space_site.send_check_space_site_quota` which will then queue the worker task `dynamo.tasks.check_space_quotas` on every site's `worker` function queue. It will then send a task back to the hub `celery` queue to store the quotas in the database.

The schedule of this task can be controlled by using the SYNC_SPACES_QUOTA_INTERVAL config setting, any provided value must comply with cron syntax.

Hourly

`dynamocore.tasks.jobs.update_state_as_failed_for_inactive_tasks`

This task will assess all tasks currently that have been started in Ngenea Hub and will ensure that they are not lost due to a communication error.

Any tasks found to not be actively running within any of the defined queues will be marked as invalid after the configurable time threshold `task_invalidation_timeout` in the configuration API.

The schedule of this task can be controlled by using the INACTIVE_TASKS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.jobs.invalidate_cancelled_job_tasks`

After jobs are cancelled within the Ngenea Hub some tasks can be left in an un-cancelled state if the service un-expectedly exits leaving tasks in a non-cancelled state indefinitely. This task will automatically clean up these tasks that are in incorrect states.

The schedule of this task can be controlled by using the INVALIDATE_CANCELLED_JOB_TASKS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.network.ensure_consistent_tc_rules`

With bandwidth controls enabled for any sites this task ensures that these rules are enacted on each site. Sites that are already compliant with their networking rules will not see any effect.

`dynamocore.tasks.events.cleanup_old_events`

This task automatically removes bidirectional sync events within the database. It will clear all objects for all but the latest 2 jobs along with any jobs that have been cancelled and failed

The schedule of this task can be controlled by using the CLEANUP_OLD_EVENTS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.browsing.browsing_task_clear`

Removes all cached task results for API requests for file details for the file browser in the UI. It will remove all entries that are older than an hour along with any that received errors when retrieving results.

`dynamocore.tasks.system_settings.create_or_update_site_settings_auto`

This task starts up a task in the main `celery` queue within Ngenea Hub for updating or creating outstanding settings across all active sites within the management of Ngenea Hub. This starts by executing `dynamocore.tasks.site.active_sites` followed immediately by

`dynamocore.tasks.system_settings.create_or_update_site_settings_for_sites`. This will cause Ngenea Hub to retrieve the settings for every site using `dynamo.tasks.get_config_for_keys` and update the hub database accordingly using an additional task on the main `celery` queue within Ngenea Hub.

This call uses the main `worker` function queue with the Ngenea Worker, more details on queue functions are available in the function configuration.

The schedule of this task can be controlled by using the SYNC_SITE_SETTINGS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.system_settings.sync_global_settings_auto`

Starts up a task in the main `celery` within Ngenea Hub that creates a collection of tasks to retrieve the global settings and external Ngenea targets within the hub database and apply them to every Ngenea Worker. It achieves this with the Ngenea Worker task `dynamo.tasks.set_config_and_apply` this is run on the main `worker` function.

The schedule of this task can be controlled by using the SYNC_GLOBAL_SETTINGS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.spaces.sync_spaces`

When a space is created or updated and the job fails, this automated task schedules a retry attempt on making the space. If this is already being processed, it will not update the space within this interval.

The schedule of this task can be controlled by using the SYNC_SPACES_INTERVAL config setting, any provided value must comply with cron syntax.

Every 12 Hours

`dynamocore.tasks.site.fetch_site_analytics`

.. note:: This has an offset of 37 minutes to prevent overlapping system tasks.

Retrieves the site-wide analytics from all sites and then stores them within the Ngenea Hub database. The timeout for this operation is controlled via the `analytics_timeout` in the configuration API. This analytics task is run on the `discovery` function within the Ngenea Hub.

The schedule of this task can be controlled by using the REFRESH_SITE_ANALYTICS_INTERVAL config setting, any provided value must comply with cron syntax.

Daily

`dynamocore.tasks.search.remove_old_search_results`

This automated task removes all the old search result caches for searches submitted by the UI. The time period for this lifetime is controlled by the `search_result_ttl` in the configuration API, the default for this threshold is 7 days.

The schedule of this task can be controlled by using the REMOVE_OLD_SEARCH_RESULTS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.jobs.expire_old_jobs`

This automated task removes all the old jobs (settings, workflows, scheduled) that have been submitted and processed within Ngenea Hub.

The threshold for this can be controlled by the `jobs_ttl` in the configuration API, the default for this is 30 days.

The schedule of this task can be controlled by using the EXPIRE_OLD_JOBS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.network.update_cloud_ips_for_s3`

Once per day, Ngenea Hub collects the IP addresses published by Amazon for their S3 service for uses in controlling the bandwidth of any S3 related operations going through Ngenea Worker.

If there are any changes in the address listings, this will be reported to the Ngenea Worker which will adjust any network interface it may be managing.

If bandwidth controls are not enabled, the changes will have no effect on any existing Ngenea Worker instances.

`dynamocore.tasks.network.update_cloud_ips_for_azure`

Once per day, Ngenea Hub collects the IP addresses published by Microsoft for their Azure Blob service for uses in controlling the bandwidth of any Azure related operations going through Ngenea Worker.

If there are any changes in the address listings, this will be reported to the Ngenea Worker which will adjust any network interface it may be managing.

If bandwidth controls are not enabled, the changes will have no effect on any existing Ngenea Worker instances.

`dynamocore.tasks.network.update_cloud_ips_for_google`

Once per day, Ngenea Hub collects the IP addresses published by Google for their GCS service for uses in controlling the bandwidth of any GCS related operations going through Ngenea Worker.

If there are any changes in the address listings, this will be reported to the Ngenea Worker which will adjust any network interface it may be managing.

If bandwidth controls are not enabled, the changes will have no effect on any existing Ngenea Worker instances.

`dynamocore.tasks.server.get_remote_servers`

Once per day Ngenea Hub has to retrieve the currently configured CTDB servers on each Ngenea Worker. It achieves this through the Ngenea Worker task `dynamo.tasks.get_servers` on the `worker` function of each site. This is then stored through the main Ngenea Hub `celery` queue through the `dynamocore.tasks.server.store_server_callback` task.

The schedule of this task can be controlled by using the SYNC_REMOTE_SERVERS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.jobs.expire_old_fsobjects`

Removes the stored long term file browser caching entries in the database and will expire any entries that have not been accessed within 7 days.

The schedule of this task can be controlled by using the EXPIRE_OLD_FSOBJECTS_INTERVAL config setting, any provided value must comply with cron syntax.

`dynamocore.tasks.refresh.delete_removed_queues`

Once queues are marked for deletion after being removed from a sites config file and the queue removal tool has been run, it will collect these queues and remove them along with and alerts linked to them.

The schedule of this task can be controlled by using the REMOVED_QUEUE_CLEANUP_INTERVAL config setting, any provided value must comply with cron syntax.

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
`Pausing`	The job has been manually paused via request and is waiting for on-going tasks to complete

State	Description
`Paused`	The job has been manually paused via request and can be resumed or cancelled manually
`Cancelled`	The job has been manually closed via request and all remaining task have been cancelled
`Dequeuing`	The job has been manually cancelled via request and is in the process of removing tasks from the queue

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.

File States

As a file is processed by a workflow, each task reports whether the state of the file following the task. The containing job reports the state of the file after all workflow steps have completed.

A file can have any of the following states

Processed

The file was successfully processed by the workflow step without issues

Skipped

The files was not processed because it was already in the expected state, e.g. a delete was skipped because the file wasn't found, a migration was skipped because the file was already offline.

Because it is in the expected state, skipped files can be processed by subsequent tasks in the workflow.

Failed

An error occurred meaning the file couldn't be processed. Because of this error, the file will not be processed by later tasks in the workflow.

The overall job will be marked as failed if it contains any failed files. Workflows which use the `snapdiff` discovery will be 'rolled back' in this case.

Failed files typically indicate an issue that needs to be manually resolved, such as network issues.

Aborted

An error occurred meaning the file couldn't be processed. Because the file was aborted, it will not be processed by later tasks in the workflow.

The overall job is **not** marked as failed if it contains aborted files. Workflows which use the `snapdiff` discovery will 'rotate' in this case.

The intended use case for this state is in scheduled (recurring) jobs, where the any aborted files are expected to be successfully processed in a later run.

For example, if a file changes while it is being sent, the send will be aborted to prevent data corruption. But because the file was modified, it will be identified as modified in the next snapdiff scan and processed again; effectively retried.

Templated email

Overview

The templated email task supports custom templates using **Jinja2**. Templates allow you to control the structure and content of notification emails generated by the `templated_email` task.

If no template is provided, a default template will be used.

Default Template

If no template is provided, the system uses a default template:

```
Job {{ jobid }} completed

Total paths: {{ path_count }}

Files:
- <each file path>
```

Supported Placeholders

The following placeholders are available and will be replaced at runtime:

Placeholder	Type	Description
`{{ jobid }}`	string	Job ID
`{{ paths }}`	list	List of processed and skipped file paths
`{{ path_count }}`	integer	Total number of processed/skipped files

Note: Currently, only the placeholders listed above are supported.

Example Template

```
Job {{ jobid }} completed successfully.

Total files processed: {{ path_count }}

Files:
{% for p in paths %}
- {{ p }}
{% endfor %}
```

Usage

Provide the template file path when triggering the task:

```
from arcapix.dynamo.server.tasks.mailtemplate import templated_em
ail

templated_email(
    jobid="12345",
    email_from="noreply@example.com",
    email_to=["user@example.com"],
    paths=[{"path": "/mmfs1/data/file1"}, {"path": "/mmfs1/data/
file2"}],
    template="/path/to/template.txt"
)
```

Template Requirements

- Templates must follow **Jinja2 syntax**
- Only the documented placeholders are supported

- The template file must exist on the worker node where the task will run. If the worker runs on multiple nodes in a site, place the template in a location all nodes can access, such as under `/mmfs1`

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 /alerts/

Query Parameters:

- **name** (*string*) – name
- **node** (*string*) – node

- **site** (*string*) – site
- **severity** (*string*) – severity
- **summary** (*string*) – summary
- **timestamp** (*string*) – timestamp
- **suppressed** (*string*) – suppressed

Status Codes:

- 200 OK –

Response JSON Object:

- **[].created** (*string*) – Time that the alert was recorded in Hub
- **[].id** (*integer*) – (read only)
- **[].name** (*string*) – Name of of the alert (required)
- **[].node** (*string*) – Hostname of the node the alert originated from
- **[].queue** (*string*) – Name of the queue that the alert relates to
- **[].severity** (*string*) – (required)
- **[].site** (*string*) – (read only)
- **[].summary** (*string*) – Summary description of the alert
- **[].suppressed** (*boolean*) – Whether the alert is suppressed
- **[].timestamp** (*string*) – Time when the alert was triggered (required)

`GET /alerts/{id}/`

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **created** (*string*) – Time that the alert was recorded in Hub
- **id** (*integer*) – (read only)
- **name** (*string*) – Name of of the alert (required)
- **node** (*string*) – Hostname of the node the alert originated from
- **queue** (*string*) – Name of the queue that the alert relates to
- **severity** (*string*) – (required)
- **site** (*string*) – (read only)
- **summary** (*string*) – Summary description of the alert
- **suppressed** (*boolean*) – Whether the alert is suppressed
- **timestamp** (*string*) – Time when the alert was triggered (required)

`GET /analytics/`

Retrieves list of files under given path for given site.

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.
- **path** (*string*) – Target directory path
- **site** (*string*) – Site name
- **children** (*boolean*) – Include children directories

- **cache_ttl** (*integer*) – How long the cache will last for the target path
- **custom_timeout** (*integer*) – Timeout for fetching analytics
- **raise_exception** (*boolean*) – Raise exception if timeout
- **download** (*string*) – Download details as file e.g. CSV `?download=csv`.

Status Codes:

- 200 OK –

`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)
- **results[].user** (*string*) – (required)

`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)
- **user** (*string*) – (required)

`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)
- **user** (*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)
- **user** (*string*) – (read only)

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

API endpoint for managing client keys

Parameters:

- **id** (*string*) –

Status Codes:

- 204 No Content –

`GET /auth/iris/token/`

Exchange a Hub token for an IRIS/Keycloak JWT token.

Requires that the authenticated user has previously logged in using IRIS SSO and Hub has captured the offline refresh token for that user.

Status Codes:

- 200 OK –

`POST /auth/keycloak/token/`

Exchange a Keycloak token for Django JWT tokens.

Validates the Keycloak token and returns Django JWT access/refresh tokens along with user information.

Status Codes:

- 201 Created –

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 /automatedschedules/

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*) – (required)
- **results[].day_of_week** (*string*) – (required)
- **results[].discovery_id** (*integer*) – (read only)
- **results[].discovery_managed_paths** (*string*) – (read only)
- **results[].discovery_name** (*string*) – (read only)
- **results[].discovery_workflow** (*string*) – (read only)
- **results[].enabled** (*boolean*) – (required)
- **results[].hour** (*string*) – (required)
- **results[].minute** (*string*) – (required)
- **results[].month_of_year** (*string*) – (required)
- **results[].name** (*string*) – (required)
- **results[].policy_id** (*integer*) – (read only)
- **results[].policy_metadata_query** (*string*) – (read only)
- **results[].policy_name** (*string*) – (read only)
- **results[].policy_type** (*string*) – (read only)
- **results[].schedule_type** (*string*) – (required)
- **results[].site_name** (*string*) – (read only)
- **results[].spaces** (*string*) – (read only)

`GET /buckets/`

Model View Set for the Bucket model.

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[].access_key** (*string*) – (read only)
- **results[].access_key_id** (*string*) – (read only)
- **results[].advanced_settings** (*object*) – Advanced settings for the target
- **results[].container** (*string*) – Azure storage container
- **results[].credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **results[].credentials_json** (*string*) – (read only)
- **results[].endpoint** (*string*) – Hostname of the target storage service

- **results[].external_targets[]** (*integer*) –
- **results[].id** (*integer*) – (read only)
- **results[].label** (*string*) – Bucket label
- **results[].migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **results[].name** (*string*) – Bucket name (required)
- **results[].port** (*integer*) – Port of the target storage service
- **results[].region** (*string*) – Region where the bucket is located
- **results[].scheme** (*string*) – Connection protocol
- **results[].secret_access_key** (*string*) – (read only)
- **results[].soft_deleting** (*boolean*) – The bucket is soft-deleting. Remote tasks are still running.
- **results[].ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **results[].storage_account** (*string*) – Azure storage account name
- **results[].storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)

`POST /buckets/`

Model View Set for the Bucket model.

Request JSON Object:

- **access_key** (*string*) – Azure-specific access key
- **access_key_id** (*string*) – Access key ID for the storage service
- **advanced_settings** (*object*) – Advanced settings for the target
- **container** (*string*) – Azure storage container
- **credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **credentials_json** (*object*) – JSON blob containing Google Cloud credentials
- **endpoint** (*string*) – Hostname of the target storage service
- **id** (*integer*) – (read only)
- **label** (*string*) – Bucket label
- **migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **name** (*string*) – Bucket name (required)
- **port** (*integer*) – Port of the target storage service
- **region** (*string*) – Region where the bucket is located
- **scheme** (*string*) – Connection protocol
- **secret_access_key** (*string*) – Secret access key for the storage service
- **ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **storage_account** (*string*) – Azure storage account name
- **storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)

Status Codes:

- 201 Created –

Response JSON Object:

- **access_key** (*string*) – Azure-specific access key
- **access_key_id** (*string*) – Access key ID for the storage service
- **advanced_settings** (*object*) – Advanced settings for the target
- **container** (*string*) – Azure storage container
- **credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **credentials_json** (*object*) – JSON blob containing Google Cloud credentials

- **endpoint** (*string*) – Hostname of the target storage service
- **id** (*integer*) – (read only)
- **label** (*string*) – Bucket label
- **migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **name** (*string*) – Bucket name (required)
- **port** (*integer*) – Port of the target storage service
- **region** (*string*) – Region where the bucket is located
- **scheme** (*string*) – Connection protocol
- **secret_access_key** (*string*) – Secret access key for the storage service
- **ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **storage_account** (*string*) – Azure storage account name
- **storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)

`GET /buckets/{id}/`

Model View Set for the Bucket model.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **access_key** (*string*) – (read only)
- **access_key_id** (*string*) – (read only)
- **advanced_settings** (*object*) – Advanced settings for the target
- **container** (*string*) – Azure storage container
- **credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **credentials_json** (*string*) – (read only)
- **endpoint** (*string*) – Hostname of the target storage service
- **external_targets[]** (*integer*) –
- **id** (*integer*) – (read only)
- **label** (*string*) – Bucket label
- **migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **name** (*string*) – Bucket name (required)
- **port** (*integer*) – Port of the target storage service
- **region** (*string*) – Region where the bucket is located
- **scheme** (*string*) – Connection protocol
- **secret_access_key** (*string*) – (read only)
- **soft_deleting** (*boolean*) – The bucket is soft-deleting. Remote tasks are still running.
- **ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **storage_account** (*string*) – Azure storage account name
- **storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)

`PATCH /buckets/{id}/`

Model View Set for the Bucket model.

Parameters:

- **id** (*string*) –

Request JSON Object:

- **access_key** (*string*) – Azure-specific access key
- **access_key_id** (*string*) – Access key ID for the storage service
- **advanced_settings** (*object*) – Advanced settings for the target
- **container** (*string*) – Azure storage container
- **credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **credentials_json** (*object*) – JSON blob containing Google Cloud credentials
- **endpoint** (*string*) – Hostname of the target storage service
- **id** (*integer*) – (read only)
- **label** (*string*) – Bucket label
- **migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **name** (*string*) – Bucket name (required)
- **port** (*integer*) – Port of the target storage service
- **region** (*string*) – Region where the bucket is located
- **scheme** (*string*) – Connection protocol
- **secret_access_key** (*string*) – Secret access key for the storage service
- **ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **storage_account** (*string*) – Azure storage account name
- **storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)

Status Codes:

- 200 OK –

Response JSON Object:

- **access_key** (*string*) – Azure-specific access key
- **access_key_id** (*string*) – Access key ID for the storage service
- **advanced_settings** (*object*) – Advanced settings for the target
- **container** (*string*) – Azure storage container
- **credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **credentials_json** (*object*) – JSON blob containing Google Cloud credentials
- **endpoint** (*string*) – Hostname of the target storage service
- **id** (*integer*) – (read only)
- **label** (*string*) – Bucket label
- **migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **name** (*string*) – Bucket name (required)
- **port** (*integer*) – Port of the target storage service
- **region** (*string*) – Region where the bucket is located
- **scheme** (*string*) – Connection protocol
- **secret_access_key** (*string*) – Secret access key for the storage service
- **ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **storage_account** (*string*) – Azure storage account name
- **storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)

`DELETE /buckets/{id}/`

Delete the bucket and apply the settings to delete them on external targets on all sites that they are available.

Parameters:

- **request** –
- **id** (*string*) –

Status Codes:

- 204 No Content –

`GET /configurations/`

API endpoint for viewing and setting configurations.

Status Codes:

- 200 OK –

Response JSON Object:

- **alert_completed_failed_jobs** (*boolean*) – Alert only failure jobs
- **alert_completed_has_failures_jobs** (*boolean*) – Alert only has_failure jobs
- **alert_completed_success_jobs** (*boolean*) – Alert only success jobs
- **alert_email_address[]** (*string*) –
- **alert_inprogress_failing_jobs** (*boolean*) – Alert only failing jobs
- **alert_notprogressed_jobs** (*boolean*) – Alert only not progressed jobs
- **alert_notstarted_jobs** (*boolean*) – Alert only not started jobs
- **alert_scheduled** (*boolean*) – Alert only scheduled jobs
- **alert_unscheduled** (*boolean*) – Alert only unscheduled jobs
- **analytics_timeout** (*integer*) – Maximum time to wait for results from analytics in seconds
- **custom_statistics_tasks** (*object*) – A list of custom tasks to be considered discovery tasks
- **db_backup_dir** (*string*) – Database backup spool dir
- **db_backup_enabled** (*boolean*) – Database backup enabled
- **db_backup_report** (*boolean*) – Database backup report
- **db_backup_report_to** (*string*) – Database backup report recipient (e.g. “hub” or empty)
- **db_backup_retain_versions** (*integer*) – Database backup retain versions
- **db_backup_schedule** (*string*) –
- **db_backup_type** (*string*) – Database backup type
- **default_space_location** (*string*) – Default location for space
- **filebrowser_proxies_backend** (*string*) – Filebrowser proxies backend
- **force_local_managed_users** (*boolean*) – Allow NAS users to be managed on AD joined sites
- **health_timeout** (*integer*) – Maximum time to wait health check to see what sites are available in seconds
- **initial_gid** (*integer*) – Minimum GID to be configured
- **initial_uid** (*integer*) – Minimum UID to be configured
- **jobs_ttl** (*integer*) – Time to store job details after the job completes, in days
- **keycloak_jwks_timeout** (*integer*) – Configurable timeout to avoid repeated HTTP requests to JWKS endpoint
- **maximum_external_results** (*integer*) – Maximum number of items to retrieve per page from an external target scan
- **node_health_last_heartbeat_interval** (*integer*) – The time since the last health heartbeat was recorded in seconds
- **notprogressed_job_threshold** (*integer*) – Time in seconds before a job is considered not progressed.

- **notstarted_job_threshold** (*integer*) – Time in seconds before a job is considered not started.
- **salt_task_timeout** (*number*) – Salt timeout in seconds
- **schedules_enabled** (*boolean*) – Allow processing of automated schedules
- **search_backend** (*string*) – Search backend
- **search_max_results** (*integer*) – Maximum search results
- **search_result_ttl** (*integer*) – Maximum time to store search results in days
- **snapdiff_stream_timeout** (*integer*) – Maximum amount of minutes to wait for task results
- **snapshot_create_delete_retry_timeout** (*integer*) – Maximum time for workers to retry snapshot create and delete operations, in seconds
- **stat_refresh_period** (*integer*) – Maximum time to wait for results from stat in seconds
- **stat_timeout** (*integer*) – Maximum time to wait for results from stat in seconds
- **task_invalidation_timeout** (*integer*) – Maximum amount of minutes before a task in the STARTED state is considered invalid

`PATCH /configurations/`

API endpoint for viewing and setting configurations.

Request JSON Object:

- **alert_completed_failed_jobs** (*boolean*) – Alert only failure jobs
- **alert_completed_has_failures_jobs** (*boolean*) – Alert only has_failure jobs
- **alert_completed_success_jobs** (*boolean*) – Alert only success jobs
- **alert_email_address[]** (*string*) –
- **alert_inprogress_failing_jobs** (*boolean*) – Alert only failing jobs
- **alert_notprogressed_jobs** (*boolean*) – Alert only not progressed jobs
- **alert_notstarted_jobs** (*boolean*) – Alert only not started jobs
- **alert_scheduled** (*boolean*) – Alert only scheduled jobs
- **alert_unscheduled** (*boolean*) – Alert only unscheduled jobs
- **analytics_timeout** (*integer*) – Maximum time to wait for results from analytics in seconds
- **custom_statistics_tasks** (*object*) – A list of custom tasks to be considered discovery tasks
- **db_backup_dir** (*string*) – Database backup spool dir
- **db_backup_enabled** (*boolean*) – Database backup enabled
- **db_backup_report** (*boolean*) – Database backup report
- **db_backup_report_to** (*string*) – Database backup report recipient (e.g. “hub” or empty)
- **db_backup_retain_versions** (*integer*) – Database backup retain versions
- **db_backup_schedule** (*string*) –
- **db_backup_type** (*string*) – Database backup type
- **default_space_location** (*string*) – Default location for space
- **filebrowser_proxies_backend** (*string*) – Filebrowser proxies backend
- **force_local_managed_users** (*boolean*) – Allow NAS users to be managed on AD joined sites
- **health_timeout** (*integer*) – Maximum time to wait health check to see what sites are available in seconds
- **initial_gid** (*integer*) – Minimum GID to be configured

- **initial_uid** (*integer*) – Minimum UID to be configured
- **jobs_ttl** (*integer*) – Time to store job details after the job completes, in days
- **keycloak_jwks_timeout** (*integer*) – Configurable timeout to avoid repeated HTTP requests to JWKS endpoint
- **maximum_external_results** (*integer*) – Maximum number of items to retrieve per page from an external target scan
- **node_health_last_heartbeat_interval** (*integer*) – The time since the last health heartbeat was recorded in seconds
- **notprogressed_job_threshold** (*integer*) – Time in seconds before a job is considered not progressed.
- **notstarted_job_threshold** (*integer*) – Time in seconds before a job is considered not started.
- **salt_task_timeout** (*number*) – Salt timeout in seconds
- **schedules_enabled** (*boolean*) – Allow processing of automated schedules
- **search_backend** (*string*) – Search backend
- **search_max_results** (*integer*) – Maximum search results
- **search_result_ttl** (*integer*) – Maximum time to store search results in days
- **snapdiff_stream_timeout** (*integer*) – Maximum amount of minutes to wait for task results
- **snapshot_create_delete_retry_timeout** (*integer*) – Maximum time for workers to retry snapshot create and delete operations, in seconds
- **stat_refresh_period** (*integer*) – Maximum time to wait for results from stat in seconds
- **stat_timeout** (*integer*) – Maximum time to wait for results from stat in seconds
- **task_invalidation_timeout** (*integer*) – Maximum amount of minutes before a task in the STARTED state is considered invalid

Status Codes:

- 200 OK –

Response JSON Object:

- **alert_completed_failed_jobs** (*boolean*) – Alert only failure jobs
- **alert_completed_has_failures_jobs** (*boolean*) – Alert only has_failure jobs
- **alert_completed_success_jobs** (*boolean*) – Alert only success jobs
- **alert_email_address[]** (*string*) –
- **alert_inprogress_failing_jobs** (*boolean*) – Alert only failing jobs
- **alert_notprogressed_jobs** (*boolean*) – Alert only not progressed jobs
- **alert_notstarted_jobs** (*boolean*) – Alert only not started jobs
- **alert_scheduled** (*boolean*) – Alert only scheduled jobs
- **alert_unscheduled** (*boolean*) – Alert only unscheduled jobs
- **analytics_timeout** (*integer*) – Maximum time to wait for results from analytics in seconds
- **custom_statistics_tasks** (*object*) – A list of custom tasks to be considered discovery tasks
- **db_backup_dir** (*string*) – Database backup spool dir
- **db_backup_enabled** (*boolean*) – Database backup enabled
- **db_backup_report** (*boolean*) – Database backup report
- **db_backup_report_to** (*string*) – Database backup report recipient (e.g. “hub” or empty)
- **db_backup_retain_versions** (*integer*) – Database backup retain versions
- **db_backup_schedule** (*string*) –

- **db_backup_type** (*string*) – Database backup type
- **default_space_location** (*string*) – Default location for space
- **filebrowser_proxies_backend** (*string*) – Filebrowser proxies backend
- **force_local_managed_users** (*boolean*) – Allow NAS users to be managed on AD joined sites
- **health_timeout** (*integer*) – Maximum time to wait health check to see what sites are available in seconds
- **initial_gid** (*integer*) – Minimum GID to be configured
- **initial_uid** (*integer*) – Minimum UID to be configured
- **jobs_ttl** (*integer*) – Time to store job details after the job completes, in days
- **keycloak_jwks_timeout** (*integer*) – Configurable timeout to avoid repeated HTTP requests to JWKS endpoint
- **maximum_external_results** (*integer*) – Maximum number of items to retrieve per page from an external target scan
- **node_health_last_heartbeat_interval** (*integer*) – The time since the last health heartbeat was recorded in seconds
- **notprogressed_job_threshold** (*integer*) – Time in seconds before a job is considered not progressed.
- **notstarted_job_threshold** (*integer*) – Time in seconds before a job is considered not started.
- **salt_task_timeout** (*number*) – Salt timeout in seconds
- **schedules_enabled** (*boolean*) – Allow processing of automated schedules
- **search_backend** (*string*) – Search backend
- **search_max_results** (*integer*) – Maximum search results
- **search_result_ttl** (*integer*) – Maximum time to store search results in days
- **snapdiff_stream_timeout** (*integer*) – Maximum amount of minutes to wait for task results
- **snapshot_create_delete_retry_timeout** (*integer*) – Maximum time for workers to retry snapshot create and delete operations, in seconds
- **stat_refresh_period** (*integer*) – Maximum time to wait for results from stat in seconds
- **stat_timeout** (*integer*) – Maximum time to wait for results from stat in seconds
- **task_invalidation_timeout** (*integer*) – Maximum amount of minutes before a task in the STARTED state is considered invalid

`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 /external_targets/`

Model View Set for filebrowser's ExternalTarget model.

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.
- **bucket** (*string*) – External target bucket name

Status Codes:

- 200 OK –

Response JSON Object:

- **count** (*integer*) – (required)
- **next** (*string*) –
- **previous** (*string*) –
- **results[].advanced_settings** (*object*) – Advanced settings for the target
- **results[].available_on_sites[]** (*integer*) – Sites where this target is available
- **results[].backup_only** (*boolean*) – Mark this target as backup only
- **results[].backup_site** (*integer*) – Site which claimed this target for backup
- **results[].bucket.access_key** (*string*) – (read only)
- **results[].bucket.access_key_id** (*string*) – (read only)
- **results[].bucket.advanced_settings** (*object*) – Advanced settings for the target
- **results[].bucket.container** (*string*) – Azure storage container
- **results[].bucket.credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **results[].bucket.credentials_json** (*string*) – (read only)
- **results[].bucket.endpoint** (*string*) – Hostname of the target storage service
- **results[].bucket.external_targets[]** (*integer*) –

- **results[].bucket.id** (*integer*) – (read only)
- **results[].bucket.label** (*string*) – Bucket label
- **results[].bucket.migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **results[].bucket.name** (*string*) – Bucket name (required)
- **results[].bucket.port** (*integer*) – Port of the target storage service
- **results[].bucket.region** (*string*) – Region where the bucket is located
- **results[].bucket.scheme** (*string*) – Connection protocol
- **results[].bucket.secret_access_key** (*string*) – (read only)
- **results[].bucket.soft_deleting** (*boolean*) – The bucket is soft-deleting. Remote tasks are still running.
- **results[].bucket.ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **results[].bucket.storage_account** (*string*) – Azure storage account name
- **results[].bucket.storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)
- **results[].bucket_instance** (*integer*) – Associated bucket
- **results[].delete_on_recall** (*boolean*) – Delete files from the external storage on recall
- **results[].enabled** (*boolean*) – Whether this ngenea target is enabled. Default is true
- **results[].endpoint_name** (*string*) – (read only)
- **results[].id** (*integer*) – (read only)
- **results[].include_bucket_in_endpoint_name** (*boolean*) – Set the endpoint name to <target>-<bucket>
- **results[].is_ready** (*string*) – (read only)
- **results[].local_file_regex** (*string*) – Regex filter to match files for migration to this target (required)
- **results[].name** (*string*) – Ngenea target name (required)
- **results[].space.mountpoint** (*string*) – Mount point (required)
- **results[].space.name** (*string*) – Space Name (required)
- **results[].space.sites[]** (*integer*) –

`POST /external_targets/`

Model View Set for filebrowser's ExternalTarget model.

Request JSON Object:

- **advanced_settings** (*object*) – Advanced settings for the target
- **available_on_sites[]** (*integer*) – Sites where this target is available
- **backup_only** (*boolean*) – Mark this target as backup only
- **backup_site** (*integer*) – Site which claimed this target for backup
- **bucket.access_key** (*string*) – (read only)
- **bucket.access_key_id** (*string*) – (read only)
- **bucket.advanced_settings** (*object*) – Advanced settings for the target
- **bucket.container** (*string*) – Azure storage container
- **bucket.credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **bucket.credentials_json** (*string*) – (read only)
- **bucket.endpoint** (*string*) – Hostname of the target storage service
- **bucket.external_targets[]** (*integer*) –

- **bucket.id** (*integer*) – (read only)
- **bucket.label** (*string*) – Bucket label
- **bucket.migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **bucket.name** (*string*) – Bucket name (required)
- **bucket.port** (*integer*) – Port of the target storage service
- **bucket.region** (*string*) – Region where the bucket is located
- **bucket.scheme** (*string*) – Connection protocol
- **bucket.secret_access_key** (*string*) – (read only)
- **bucket.soft_deleting** (*boolean*) – The bucket is soft-deleting. Remote tasks are still running.
- **bucket.ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **bucket.storage_account** (*string*) – Azure storage account name
- **bucket.storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)
- **bucket_instance** (*integer*) – Associated bucket
- **delete_on_recall** (*boolean*) – Delete files from the external storage on recall
- **enabled** (*boolean*) – Whether this ngenea target is enabled. Default is true
- **endpoint_name** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **include_bucket_in_endpoint_name** (*boolean*) – Set the endpoint name to <target>-<bucket>
- **is_ready** (*string*) – (read only)
- **local_file_regex** (*string*) – Regex filter to match files for migration to this target (required)
- **name** (*string*) – Ngenea target name (required)
- **space.mountpoint** (*string*) – Mount point (required)
- **space.name** (*string*) – Space Name (required)
- **space.sites[]** (*integer*) –

Status Codes:

- 201 Created –

Response JSON Object:

- **advanced_settings** (*object*) – Advanced settings for the target
- **available_on_sites[]** (*integer*) – Sites where this target is available
- **backup_only** (*boolean*) – Mark this target as backup only
- **backup_site** (*integer*) – Site which claimed this target for backup
- **bucket.access_key** (*string*) – (read only)
- **bucket.access_key_id** (*string*) – (read only)
- **bucket.advanced_settings** (*object*) – Advanced settings for the target
- **bucket.container** (*string*) – Azure storage container
- **bucket.credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **bucket.credentials_json** (*string*) – (read only)
- **bucket.endpoint** (*string*) – Hostname of the target storage service
- **bucket.external_targets[]** (*integer*) –
- **bucket.id** (*integer*) – (read only)
- **bucket.label** (*string*) – Bucket label
- **bucket.migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **bucket.name** (*string*) – Bucket name (required)

- **bucket.port** (*integer*) – Port of the target storage service
- **bucket.region** (*string*) – Region where the bucket is located
- **bucket.scheme** (*string*) – Connection protocol
- **bucket.secret_access_key** (*string*) – (read only)
- **bucket.soft_deleting** (*boolean*) – The bucket is soft-deleting. Remote tasks are still running.
- **bucket.ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **bucket.storage_account** (*string*) – Azure storage account name
- **bucket.storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)
- **bucket_instance** (*integer*) – Associated bucket
- **delete_on_recall** (*boolean*) – Delete files from the external storage on recall
- **enabled** (*boolean*) – Whether this ngenea target is enabled. Default is true
- **endpoint_name** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **include_bucket_in_endpoint_name** (*boolean*) – Set the endpoint name to <target>-<bucket>
- **is_ready** (*string*) – (read only)
- **local_file_regex** (*string*) – Regex filter to match files for migration to this target (required)
- **name** (*string*) – Ngenea target name (required)
- **space.mountpoint** (*string*) – Mount point (required)
- **space.name** (*string*) – Space Name (required)
- **space.sites[]** (*integer*) –

`GET /external_targets/{id}/`

Model View Set for filebrowser's ExternalTarget model.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **advanced_settings** (*object*) – Advanced settings for the target
- **available_on_sites[]** (*integer*) – Sites where this target is available
- **backup_only** (*boolean*) – Mark this target as backup only
- **backup_site** (*integer*) – Site which claimed this target for backup
- **bucket.access_key** (*string*) – (read only)
- **bucket.access_key_id** (*string*) – (read only)
- **bucket.advanced_settings** (*object*) – Advanced settings for the target
- **bucket.container** (*string*) – Azure storage container
- **bucket.credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **bucket.credentials_json** (*string*) – (read only)
- **bucket.endpoint** (*string*) – Hostname of the target storage service
- **bucket.external_targets[]** (*integer*) –
- **bucket.id** (*integer*) – (read only)
- **bucket.label** (*string*) – Bucket label
- **bucket.migration_target_folder** (*string*) – Filepath of migration target for FS storage type

- **bucket.name** (*string*) – Bucket name (required)
- **bucket.port** (*integer*) – Port of the target storage service
- **bucket.region** (*string*) – Region where the bucket is located
- **bucket.scheme** (*string*) – Connection protocol
- **bucket.secret_access_key** (*string*) – (read only)
- **bucket.soft_deleting** (*boolean*) – The bucket is soft-deleting. Remote tasks are still running.
- **bucket.ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **bucket.storage_account** (*string*) – Azure storage account name
- **bucket.storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)
- **bucket_instance** (*integer*) – Associated bucket
- **delete_on_recall** (*boolean*) – Delete files from the external storage on recall
- **enabled** (*boolean*) – Whether this ngenea target is enabled. Default is true
- **endpoint_name** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **include_bucket_in_endpoint_name** (*boolean*) – Set the endpoint name to <target>-<bucket>
- **is_ready** (*string*) – (read only)
- **local_file_regex** (*string*) – Regex filter to match files for migration to this target (required)
- **name** (*string*) – Ngenea target name (required)
- **space.mountpoint** (*string*) – Mount point (required)
- **space.name** (*string*) – Space Name (required)
- **space.sites[]** (*integer*) –

`PATCH /external_targets/{id}/`

Model View Set for filebrowser's ExternalTarget model.

Parameters:

- **id** (*string*) –

Request JSON Object:

- **advanced_settings** (*object*) – Advanced settings for the target
- **available_on_sites[]** (*integer*) – Sites where this target is available
- **backup_only** (*boolean*) – Mark this target as backup only
- **backup_site** (*integer*) – Site which claimed this target for backup
- **bucket.access_key** (*string*) – (read only)
- **bucket.access_key_id** (*string*) – (read only)
- **bucket.advanced_settings** (*object*) – Advanced settings for the target
- **bucket.container** (*string*) – Azure storage container
- **bucket.credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **bucket.credentials_json** (*string*) – (read only)
- **bucket.endpoint** (*string*) – Hostname of the target storage service
- **bucket.external_targets[]** (*integer*) –
- **bucket.id** (*integer*) – (read only)
- **bucket.label** (*string*) – Bucket label
- **bucket.migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **bucket.name** (*string*) – Bucket name (required)

- **bucket.port** (*integer*) – Port of the target storage service
- **bucket.region** (*string*) – Region where the bucket is located
- **bucket.scheme** (*string*) – Connection protocol
- **bucket.secret_access_key** (*string*) – (read only)
- **bucket.soft_deleting** (*boolean*) – The bucket is soft-deleting. Remote tasks are still running.
- **bucket.ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **bucket.storage_account** (*string*) – Azure storage account name
- **bucket.storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)
- **bucket_instance** (*integer*) – Associated bucket
- **delete_on_recall** (*boolean*) – Delete files from the external storage on recall
- **enabled** (*boolean*) – Whether this ngenea target is enabled. Default is true
- **endpoint_name** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **include_bucket_in_endpoint_name** (*boolean*) – Set the endpoint name to <target>-<bucket>
- **is_ready** (*string*) – (read only)
- **local_file_regex** (*string*) – Regex filter to match files for migration to this target (required)
- **name** (*string*) – Ngenea target name (required)
- **space.mountpoint** (*string*) – Mount point (required)
- **space.name** (*string*) – Space Name (required)
- **space.sites[]** (*integer*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **advanced_settings** (*object*) – Advanced settings for the target
- **available_on_sites[]** (*integer*) – Sites where this target is available
- **backup_only** (*boolean*) – Mark this target as backup only
- **backup_site** (*integer*) – Site which claimed this target for backup
- **bucket.access_key** (*string*) – (read only)
- **bucket.access_key_id** (*string*) – (read only)
- **bucket.advanced_settings** (*object*) – Advanced settings for the target
- **bucket.container** (*string*) – Azure storage container
- **bucket.credentials_file** (*string*) – JSON file containing Google Cloud credentials
- **bucket.credentials_json** (*string*) – (read only)
- **bucket.endpoint** (*string*) – Hostname of the target storage service
- **bucket.external_targets[]** (*integer*) –
- **bucket.id** (*integer*) – (read only)
- **bucket.label** (*string*) – Bucket label
- **bucket.migration_target_folder** (*string*) – Filepath of migration target for FS storage type
- **bucket.name** (*string*) – Bucket name (required)
- **bucket.port** (*integer*) – Port of the target storage service
- **bucket.region** (*string*) – Region where the bucket is located
- **bucket.scheme** (*string*) – Connection protocol
- **bucket.secret_access_key** (*string*) – (read only)

- **bucket.soft_deleting** (*boolean*) – The bucket is soft-deleting. Remote tasks are still running.
- **bucket.ssl_verify** (*boolean*) – Verify SSL certificate when connecting
- **bucket.storage_account** (*string*) – Azure storage account name
- **bucket.storage_type** (*string*) – Type of external storage (e.g., Amazon S3, Azure, etc.) (required)
- **bucket_instance** (*integer*) – Associated bucket
- **delete_on_recall** (*boolean*) – Delete files from the external storage on recall
- **enabled** (*boolean*) – Whether this ngenea target is enabled. Default is true
- **endpoint_name** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **include_bucket_in_endpoint_name** (*boolean*) – Set the endpoint name to <target>-<bucket>
- **is_ready** (*string*) – (read only)
- **local_file_regex** (*string*) – Regex filter to match files for migration to this target (required)
- **name** (*string*) – Ngenea target name (required)
- **space.mountpoint** (*string*) – Mount point (required)
- **space.name** (*string*) – Space Name (required)
- **space.sites[]** (*integer*) –

`DELETE /external_targets/{id}/`

Model View Set for filebrowser's ExternalTarget model.

Parameters:

- **id** (*string*) –

Status Codes:

- 204 No Content –

`GET /external_targets/{id}/files/`

Model View Set for filebrowser's ExternalTarget model.

Parameters:

- **id** (*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.
- **path** (*string*) – Target remote path, value can be a glob eg 'aws/cats*.jpg'. NOTE: For globs the scan is'nt recursive
- **max_items** (*integer*) – Maximum number of items to retrieve per page
- **marker** (*string*) – Last pagination response marker

Status Codes:

- 200 OK –

Response JSON Object:

- **directories[].name** (*string*) – (read only)

- **directories[].path** (*string*) – (required)
- **directories[].size** (*string*) – (read only)
- **directories[].type** (*string*) – (required)
- **files[].name** (*string*) – (read only)
- **files[].path** (*string*) – (required)
- **files[].size** (*string*) – (read only)
- **files[].type** (*string*) – (required)
- **marker** (*string*) – (required)
- **next_marker** (*string*) – (required)
- **path** (*string*) – (required)
- **summary** (*object*) – (required)

`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:

- **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.
- **path** (*string*) – Target directory path
- **site** (*string*) – Site name
- **details** (*boolean*) – Show details of children objects
- **restricted** (*boolean*) – Include restricted objects
- **cache_ttl** (*integer*) – How long the cache will last for the target path
- **type** (*string*) – item type

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*) –
- **post_action[]** (*object*) –
- **pre_action[]** (*object*) –
- **queue** (*string*) – Queue name
- **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*) –
- **post_action[]** (*object*) –
- **pre_action[]** (*object*) –
- **queue** (*string*) – Queue name
- **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 /filesystems/`

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[].mountpoint** (*string*) – Filesystem mount point (required)
- **results[].name** (*string*) – Name of the filesystem (required)
- **results[].site** (*string*) – Site the filesystem belongs to (required)

`GET /filesystems/{id}/`

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **mountpoint** (*string*) – Filesystem mount point (required)
- **name** (*string*) – Name of the filesystem (required)
- **site** (*string*) – Site the filesystem belongs to (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[].description** (*string*) – (read only)
- **results[].id** (*integer*) – (read only)
- **results[].iris_permissions[].permission** (*string*) –
- **results[].iris_permissions[].site** (*string*) – (required)
- **results[].iris_permissions[].space** (*string*) – (required)
- **results[].name** (*string*) – (required)
- **results[].nas_group.gid** (*integer*) – GID number for the user. This is automatically generated and cannot be changed. (read only)
- **results[].object_permissions[].model** (*string*) – (required)
- **results[].object_permissions[].object_pk** (*string*) – (required)
- **results[].object_permissions[].permission** (*integer*) – (required)
- **results[].permissions[]** (*integer*) –
- **results[].users[].date_joined** (*string*) –
- **results[].users[].email** (*string*) –
- **results[].users[].first_name** (*string*) –
- **results[].users[].last_login** (*string*) –
- **results[].users[].last_name** (*string*) –
- **results[].users[].username** (*string*) – Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. (required)

`POST /groups/`

API endpoint for managing groups.

Request JSON Object:

- **description** (*string*) –
- **id** (*integer*) – (read only)
- **iris_permissions[].permission** (*string*) – (required)
- **iris_permissions[].site** (*string*) – (required)

- **iris_permissions[].space** (*string*) – (required)
- **name** (*string*) – (required)
- **nas_group** (*boolean*) –
- **object_permissions[].model** (*string*) – (required)
- **object_permissions[].object_pk** (*string*) – (required)
- **object_permissions[].permission** (*integer*) – (required)
- **permissions[]** (*integer*) –
- **users[]** (*string*) –

Status Codes:

- 201 Created –

Response JSON Object:

- **description** (*string*) –
- **id** (*integer*) – (read only)
- **iris_permissions[].permission** (*string*) – (required)
- **iris_permissions[].site** (*string*) – (required)
- **iris_permissions[].space** (*string*) – (required)
- **name** (*string*) – (required)
- **nas_group** (*boolean*) –
- **object_permissions[].model** (*string*) – (required)
- **object_permissions[].object_pk** (*string*) – (required)
- **object_permissions[].permission** (*integer*) – (required)
- **permissions[]** (*integer*) –
- **users[]** (*string*) –

`POST /groups/sync/`

API to sync nas groups on each site

Status Codes:

- 201 Created –

Response JSON Object:

- **description** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **iris_permissions[].permission** (*string*) –
- **iris_permissions[].site** (*string*) – (required)
- **iris_permissions[].space** (*string*) – (required)
- **name** (*string*) – (required)
- **nas_group.gid** (*integer*) – GID number for the user. This is automatically generated and cannot be changed. (read only)
- **object_permissions[].model** (*string*) – (required)
- **object_permissions[].object_pk** (*string*) – (required)
- **object_permissions[].permission** (*integer*) – (required)
- **permissions[]** (*integer*) –
- **users[].date_joined** (*string*) –
- **users[].email** (*string*) –
- **users[].first_name** (*string*) –
- **users[].last_login** (*string*) –
- **users[].last_name** (*string*) –
- **users[].username** (*string*) – Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. (required)

GET /groups/{id}/

API endpoint for managing groups.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **description** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **iris_permissions[].permission** (*string*) –
- **iris_permissions[].site** (*string*) – (required)
- **iris_permissions[].space** (*string*) – (required)
- **name** (*string*) – (required)
- **nas_group.gid** (*integer*) – GID number for the user. This is automatically generated and cannot be changed. (read only)
- **object_permissions[].model** (*string*) – (required)
- **object_permissions[].object_pk** (*string*) – (required)
- **object_permissions[].permission** (*integer*) – (required)
- **permissions[]** (*integer*) –
- **users[].date_joined** (*string*) –
- **users[].email** (*string*) –
- **users[].first_name** (*string*) –
- **users[].last_login** (*string*) –
- **users[].last_name** (*string*) –
- **users[].username** (*string*) – Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. (required)

PATCH /groups/{id}/

API endpoint for managing groups.

Parameters:

- **id** (*string*) –

Request JSON Object:

- **description** (*string*) –
- **id** (*integer*) – (read only)
- **iris_permissions[].permission** (*string*) – (required)
- **iris_permissions[].site** (*string*) – (required)
- **iris_permissions[].space** (*string*) – (required)
- **name** (*string*) – (required)
- **nas_group** (*boolean*) –
- **object_permissions[].model** (*string*) – (required)
- **object_permissions[].object_pk** (*string*) – (required)
- **object_permissions[].permission** (*integer*) – (required)
- **permissions[]** (*integer*) –
- **users[]** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **description** (*string*) –

- **id** (*integer*) – (read only)
- **iris_permissions[].permission** (*string*) – (required)
- **iris_permissions[].site** (*string*) – (required)
- **iris_permissions[].space** (*string*) – (required)
- **name** (*string*) – (required)
- **nas_group** (*boolean*) –
- **object_permissions[].model** (*string*) – (required)
- **object_permissions[].object_pk** (*string*) – (required)
- **object_permissions[].permission** (*integer*) – (required)
- **permissions[]** (*integer*) –
- **users[]** (*string*) –

`DELETE /groups/{id}/`

API endpoint for managing groups.

Parameters:

- **id** (*string*) –

Status Codes:

- 204 No Content –

`GET /health/sites/`

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.
- **datastore_id** (*integer*) – Data store 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.
- **workflow** (*string*) – Job workflow type
- **state** (*array*) – Job states
- **owner_ids** (*array*) – Job owner user IDs. Send `-1` for the `Unknown` owner. Send `-2` for the `System` jobs.
- **clientkey_ids** (*array*) – Job clientkey IDs
- **schedule_ids** (*array*) – Job schedule IDs
- **site** (*string*) – Job site name
- **site_id** (*integer*) – Job site ID
- **input_paths_prefix** (*string*) – Path prefix for the job paths
- **search_keyword** (*string*) – Keyword to filter by
- **hide_noop** (*boolean*) – Hide successful jobs with no processed files
- **queue** (*string*) – Job queue name
- **queue_id** (*integer*) – Job queue ID

Status Codes:

- 200 OK –

`POST /jobs/`

API endpoint for managing jobs.

Status Codes:

- 201 Created –

`GET /jobs/jobtype_choices/`

Get all custom jobtypes.

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 –

`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 –

`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.
- **workflow** (*string*) – Job workflow type
- **state** (*array*) – Job states

- **owner_ids** (*array*) – Job owner user IDs. Send `-1` for the `Unknown` owner. Send `-2` for the `System` jobs.
- **clientkey_ids** (*array*) – Job clientkey IDs
- **schedule_ids** (*array*) – Job schedule IDs
- **site** (*string*) – Job site name
- **site_id** (*integer*) – Job site ID
- **input_paths_prefix** (*string*) – Path prefix for the job paths
- **search_keyword** (*string*) – Keyword to filter by
- **hide_noop** (*boolean*) – Hide successful jobs with no processed files
- **queue** (*string*) – Job queue name
- **queue_id** (*integer*) – Job queue ID

Status Codes:

- 200 OK –

`GET /jobs/{id}/`

API endpoint for managing jobs.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

`PATCH /jobs/{id}/`

API endpoint for managing jobs.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

`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 –

`GET /jobs/{id}/files/`

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

Parameters:

- **id** (*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.
- **category** (*string*) –

Status Codes:

- 200 OK –

`POST /jobs/{id}/pause/`

API endpoint to pause the active dag job

Parameters:

- **id** (*string*) –

Status Codes:

- 201 Created –

`GET /jobs/{id}/progress/`

Returns the job completion percentage for the given job id

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

`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 –

`POST /jobs/{id}/resume/`

API endpoint to resume the paused dag job

Parameters:

- **id** (*string*) –

Status Codes:

- 201 Created –

`GET /jobs/{id}/statistics/`

Returns the statistics of all files for the given job id

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

GET /jobstats/

API endpoint for managing jobstat.

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.
- **state** (*array*) – Job states
- **job** (*integer*) – Job ID
- **path** (*string*) – Target path
- **history** (*boolean*) – Returns the task history along with job stat

Status Codes:

- 200 OK –

Response JSON Object:

- **count** (*integer*) – (required)
- **next** (*string*) –
- **previous** (*string*) –
- **results[].code** (*string*) –
- **results[].id** (*integer*) – (read only)
- **results[].job** (*integer*) – Related job id (required)
- **results[].message** (*string*) –
- **results[].path** (*string*) – Path to object on filesystem
- **results[].state** (*string*) –
- **results[].task** (*integer*) – Latest related dag task (required)
- **results[].url** (*string*) – (read only)

GET /jobstats/{id}/

API endpoint for managing jobstat.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **code** (*string*) –
- **id** (*integer*) – (read only)
- **job** (*integer*) – Related job id (required)
- **message** (*string*) –
- **path** (*string*) – Path to object on filesystem
- **state** (*string*) –
- **task** (*integer*) – Latest related dag task (required)

GET /jobstats/{id}/history/

Returns the task history for the job stat ID

Parameters:

- **id** (*string*) –

Query Parameters:

- **state** (*array*) – Job states
- **job** (*integer*) – Job ID
- **path** (*string*) – Target path
- **history** (*boolean*) – Returns the task history along with job stat

Status Codes:

- 200 OK –

Response JSON Object:

- **code** (*string*) –
- **id** (*integer*) – (read only)
- **job** (*integer*) – Related job id (required)
- **message** (*string*) –
- **path** (*string*) – Path to object on filesystem
- **state** (*string*) –
- **task** (*integer*) – Latest related dag task (required)

`GET /my-permissions/`

API endpoint for viewing the requesting user permissions.

Status Codes:

- 200 OK –

Response JSON Object:

- **[].group_object_permissions.app_label** (*string*) – (required)
- **[].group_object_permissions.codename** (*string*) – (required)
- **[].group_object_permissions.id** (*integer*) – (required)
- **[].group_object_permissions.model** (*string*) – (required)
- **[].group_object_permissions.name** (*string*) – (required)
- **[].group_object_permissions.object_pk** (*string*) – (required)
- **[].group_permissions.app_label** (*string*) – (required)
- **[].group_permissions.codename** (*string*) – (required)
- **[].group_permissions.id** (*integer*) – (read only)
- **[].group_permissions.model** (*string*) – (required)
- **[].group_permissions.name** (*string*) – (required)
- **[].user_object_permissions.app_label** (*string*) – (required)
- **[].user_object_permissions.codename** (*string*) – (required)
- **[].user_object_permissions.id** (*integer*) – (required)
- **[].user_object_permissions.model** (*string*) – (required)
- **[].user_object_permissions.name** (*string*) – (required)
- **[].user_object_permissions.object_pk** (*string*) – (required)
- **[].user_permissions.app_label** (*string*) – (required)
- **[].user_permissions.codename** (*string*) – (required)
- **[].user_permissions.id** (*integer*) – (read only)
- **[].user_permissions.model** (*string*) – (required)
- **[].user_permissions.name** (*string*) – (required)

`GET /nodeclasses/`

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*) – (required)
- **results[].nodes[]** (*string*) –
- **results[].site** (*string*) – (required)

`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[].has_default_queue** (*string*) – (read only)
- **results[].id** (*integer*) – (read only)
- **results[].last_heartbeat** (*string*) – Time the node sent its last heartbeat event
- **results[].name** (*string*) – Hostname for a given worker node (required)
- **results[].online** (*string*) – (read only)
- **results[].site** (*string*) – (required)
- **results[].url** (*string*) – (read only)

`GET /nodes/{id}/`

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **has_default_queue** (*string*) – (read only)

- **id** (*integer*) – (read only)
- **last_heartbeat** (*string*) – Time the node sent its last heartbeat event
- **name** (*string*) – Hostname for a given worker node (required)
- **online** (*string*) – (read only)
- **site** (*string*) – (required)
- **url** (*string*) – (read only)

`PATCH /nodes/{id}/`

Parameters:

- **id** (*string*) –

Request JSON Object:

- **has_default_queue** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **last_heartbeat** (*string*) – Time the node sent its last heartbeat event
- **name** (*string*) – Hostname for a given worker node (required)
- **online** (*string*) – (read only)
- **site** (*string*) – (required)
- **url** (*string*) – (read only)

Status Codes:

- 200 OK –

Response JSON Object:

- **has_default_queue** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **last_heartbeat** (*string*) – Time the node sent its last heartbeat event
- **name** (*string*) – Hostname for a given worker node (required)
- **online** (*string*) – (read only)
- **site** (*string*) – (required)
- **url** (*string*) – (read only)

`DELETE /nodes/{id}/`

Parameters:

- **id** (*string*) –

Status Codes:

- 204 No Content –

`GET /permissions/`

API endpoint for viewing permissions.

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[].app_label** (*string*) – (required)
- **results[].codename** (*string*) – (required)
- **results[].id** (*integer*) – (read only)
- **results[].model** (*string*) – (required)
- **results[].name** (*string*) – (required)

`GET /permissions/{id}/`

API endpoint for viewing permissions.

Parameters:
- **id** (*string*) –

Status Codes:
- 200 OK –

Response JSON Object:
- **app_label** (*string*) – (required)
- **codename** (*string*) – (required)
- **id** (*integer*) – (read only)
- **model** (*string*) – (required)
- **name** (*string*) – (required)

`GET /pki/ca/pub/`

Status Codes:
- 200 OK –

`GET /pki/lifetime/`

Status Codes:
- 200 OK –

Response JSON Object:
- **[].days** (*integer*) – The number of days the certificate is valid for.
- **[].updated_at** (*string*) – (read only)
- **[].updated_by** (*integer*) – (read only)

`GET /pki/lifetime/{id}/`

Parameters:
- **id** (*integer*) – A unique integer value identifying this certificate lifetime.

Status Codes:
- 200 OK –

Response JSON Object:
- **days** (*integer*) – The number of days the certificate is valid for.
- **updated_at** (*string*) – (read only)
- **updated_by** (*integer*) – (read only)

`PATCH /pki/lifetime/{id}/`

Parameters:
- **id** (*integer*) – A unique integer value identifying this certificate lifetime.

Request JSON Object:
- **days** (*integer*) – The number of days the certificate is valid for.
- **updated_at** (*string*) – (read only)
- **updated_by** (*integer*) – (read only)

Status Codes:

- 200 OK –

Response JSON Object:

- **days** (*integer*) – The number of days the certificate is valid for.
- **updated_at** (*string*) – (read only)
- **updated_by** (*integer*) – (read only)

GET /pki/service/private/

Status Codes:

- 200 OK –

GET /pki/service/pub/

Status Codes:

- 200 OK –

POST /pki/site/

Status Codes:

- 201 Created –

GET /policies/

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[].condition_groups[].date_conditions[].age** (*string*) –
- **results[].condition_groups[].date_conditions[].condition_group** (*integer*) – (required)
- **results[].condition_groups[].date_conditions[].date_field** (*string*) –
- **results[].condition_groups[].date_conditions[].id** (*integer*) – (read only)
- **results[].condition_groups[].date_conditions[].include** (*boolean*) –
- **results[].condition_groups[].date_conditions[].older_or_newer** (*string*) –
- **results[].condition_groups[].filesize_conditions[].condition_group** (*integer*) – (required)
- **results[].condition_groups[].filesize_conditions[].greater_or_less** (*string*) – (required)
- **results[].condition_groups[].filesize_conditions[].id** (*integer*) – (read only)
- **results[].condition_groups[].filesize_conditions[].include** (*boolean*) –

- **results[].condition_groups[].filesize_conditions[].size** (*integer*) – (required)
- **results[].condition_groups[].filetype_conditions[].condition_group** (*integer*) – (required)
- **results[].condition_groups[].filetype_conditions[].filetypes[]** (*string*) –
- **results[].condition_groups[].filetype_conditions[].id** (*integer*) – (read only)
- **results[].condition_groups[].filetype_conditions[].include** (*boolean*) –
- **results[].condition_groups[].id** (*integer*) – (read only)
- **results[].condition_groups[].name** (*string*) – (required)
- **results[].condition_groups[].policy** (*integer*) – (required)
- **results[].created** (*string*) – (read only)
- **results[].enabled** (*boolean*) –
- **results[].filesystem** (*string*) –
- **results[].id** (*integer*) – (read only)
- **results[].iscan_buckets** (*integer*) –
- **results[].iscan_threads** (*integer*) –
- **results[].max_files** (*integer*) – Bucket Size
- **results[].name** (*string*) – (required)
- **results[].node_class[]** (*string*) –
- **results[].order.by** (*string*) –
- **results[].order.id** (*integer*) – (read only)
- **results[].order.reverse** (*boolean*) –
- **results[].owner** (*integer*) –
- **results[].policy_type** (*string*) –
- **results[].schedule.day_of_month** (*string*) – Cron Days Of The Month to Run. Use “*” for “all”. (Example: “1,15”)
- **results[].schedule.day_of_week** (*string*) – Cron Days Of The Week to Run. Use “*” for “all”. (Example: “0,5”)
- **results[].schedule.first_occur** (*boolean*) – Whether the first occurrence will be at the stipulated time.
- **results[].schedule.hour** (*string*) – Cron Hours to Run. Use “*” for “all”. (Example: “8,20”)
- **results[].schedule.id** (*integer*) – (read only)
- **results[].schedule.minute** (*string*) – Cron Minutes to Run. Use “*” for “all”. (Example: “0,30”)
- **results[].schedule.month_of_year** (*string*) – Cron Months Of The Year to Run. Use “*” for “all”. (Example: “0,6”)
- **results[].schedule.start_date_time** (*string*) – Start date time based on the time provided, timezone and if self.first_occur is true.
- **results[].schedule.time** (*string*) – Time to start running on the policy’s SITE. REMEMBER: The Ngenea hub and the site may be running in different timezones
- **results[].schedule.timezone** (*string*) – (required)
- **results[].site** (*integer*) –
- **results[].spaces[]** (*integer*) –
- **results[].threads** (*integer*) – Number of threads to run policy on
- **results[].triggers** (*string*) – (read only)
- **results[].vision_metadata_query** (*object*) – Vision Metadata Query

POST /policies/

Request JSON Object:

- **created** (*string*) – (read only)
- **enabled** (*boolean*) –
- **filesystem** (*string*) –
- **id** (*integer*) – (read only)
- **iscan_buckets** (*integer*) –
- **iscan_threads** (*integer*) –
- **max_files** (*integer*) – Bucket Size
- **name** (*string*) – (required)
- **node_class[]** (*string*) –
- **order.by** (*string*) –
- **order.id** (*integer*) – (read only)
- **order.reverse** (*boolean*) –
- **owner** (*integer*) –
- **policy_type** (*string*) –
- **run_immediately** (*boolean*) –
- **schedule.day_of_month** (*string*) – Cron Days Of The Month to Run. Use “*” for “all”. (Example: “1,15”)
- **schedule.day_of_week** (*string*) – Cron Days Of The Week to Run. Use “*” for “all”. (Example: “0,5”)
- **schedule.first_occur** (*boolean*) – Whether the first occurrence will be at the stipulated time.
- **schedule.hour** (*string*) – Cron Hours to Run. Use “*” for “all”. (Example: “8,20”)
- **schedule.id** (*integer*) – (read only)
- **schedule.minute** (*string*) – Cron Minutes to Run. Use “*” for “all”. (Example: “0,30”)
- **schedule.month_of_year** (*string*) – Cron Months Of The Year to Run. Use “*” for “all”. (Example: “0,6”)
- **schedule.time** (*string*) – Time to start running on the policy’s SITE. REMEMBER: The Ngenea hub and the site may be running in different timezones
- **site** (*integer*) –
- **spaces[]** (*integer*) –
- **threads** (*integer*) – Number of threads to run policy on
- **triggers[].is_cloud** (*boolean*) –
- **triggers[].lower_threshold** (*number*) –
- **triggers[].max_utilisation** (*number*) –
- **triggers[].pool_name** (*string*) – (required)
- **triggers[].premigrate_threshold** (*number*) –
- **triggers[].upper_threshold** (*number*) –
- **vision_metadata_query** (*object*) – Vision Metadata Query

Status Codes:

- 201 Created –

Response JSON Object:

- **created** (*string*) – (read only)
- **enabled** (*boolean*) –
- **filesystem** (*string*) –
- **id** (*integer*) – (read only)

- **iscan_buckets** (*integer*) –
- **iscan_threads** (*integer*) –
- **max_files** (*integer*) – Bucket Size
- **name** (*string*) – (required)
- **node_class[]** (*string*) –
- **order** (*integer*) –
- **owner** (*integer*) –
- **policy_type** (*string*) –
- **run_immediately** (*boolean*) –
- **schedule.day_of_month** (*string*) – Cron Days Of The Month to Run. Use “*” for “all”. (Example: “1,15”)
- **schedule.day_of_week** (*string*) – Cron Days Of The Week to Run. Use “*” for “all”. (Example: “0,5”)
- **schedule.first_occur** (*boolean*) – Whether the first occurrence will be at the stipulated time.
- **schedule.hour** (*string*) – Cron Hours to Run. Use “*” for “all”. (Example: “8,20”)
- **schedule.id** (*integer*) – (read only)
- **schedule.minute** (*string*) – Cron Minutes to Run. Use “*” for “all”. (Example: “0,30”)
- **schedule.month_of_year** (*string*) – Cron Months Of The Year to Run. Use “*” for “all”. (Example: “0,6”)
- **schedule.time** (*string*) – Time to start running on the policy’s SITE. REMEMBER: The Ngenea hub and the site may be running in different timezones
- **site** (*integer*) –
- **spaces[]** (*integer*) –
- **threads** (*integer*) – Number of threads to run policy on
- **triggers[].id** (*integer*) – (read only)
- **triggers[].is_cloud** (*boolean*) –
- **triggers[].lower_threshold** (*number*) –
- **triggers[].max_utilisation** (*number*) –
- **triggers[].pool_name** (*string*) – (required)
- **triggers[].premigrate_threshold** (*number*) –
- **triggers[].upper_threshold** (*number*) –
- **vision_metadata_query** (*object*) – Vision Metadata Query

GET /policies/{id}/

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **condition_groups[].date_conditions[].age** (*string*) –
- **condition_groups[].date_conditions[].condition_group** (*integer*) – (required)
- **condition_groups[].date_conditions[].date_field** (*string*) –
- **condition_groups[].date_conditions[].id** (*integer*) – (read only)
- **condition_groups[].date_conditions[].include** (*boolean*) –
- **condition_groups[].date_conditions[].older_or_newer** (*string*) –

- **condition_groups[].filesize_conditions[].condition_group** (*integer*) – (required)
- **condition_groups[].filesize_conditions[].greater_or_less** (*string*) – (required)
- **condition_groups[].filesize_conditions[].id** (*integer*) – (read only)
- **condition_groups[].filesize_conditions[].include** (*boolean*) –
- **condition_groups[].filesize_conditions[].size** (*integer*) – (required)
- **condition_groups[].filetype_conditions[].condition_group** (*integer*) – (required)
- **condition_groups[].filetype_conditions[].filetypes[]** (*string*) –
- **condition_groups[].filetype_conditions[].id** (*integer*) – (read only)
- **condition_groups[].filetype_conditions[].include** (*boolean*) –
- **condition_groups[].id** (*integer*) – (read only)
- **condition_groups[].name** (*string*) – (required)
- **condition_groups[].policy** (*integer*) – (required)
- **created** (*string*) – (read only)
- **enabled** (*boolean*) –
- **filesystem** (*string*) –
- **id** (*integer*) – (read only)
- **iscan_buckets** (*integer*) –
- **iscan_threads** (*integer*) –
- **max_files** (*integer*) – Bucket Size
- **name** (*string*) – (required)
- **node_class[]** (*string*) –
- **order.by** (*string*) –
- **order.id** (*integer*) – (read only)
- **order.reverse** (*boolean*) –
- **owner** (*integer*) –
- **policy_type** (*string*) –
- **schedule.day_of_month** (*string*) – Cron Days Of The Month to Run. Use “*” for “all”. (Example: “1,15”)
- **schedule.day_of_week** (*string*) – Cron Days Of The Week to Run. Use “*” for “all”. (Example: “0,5”)
- **schedule.first_occur** (*boolean*) – Whether the first occurrence will be at the stipulated time.
- **schedule.hour** (*string*) – Cron Hours to Run. Use “*” for “all”. (Example: “8,20”)
- **schedule.id** (*integer*) – (read only)
- **schedule.minute** (*string*) – Cron Minutes to Run. Use “*” for “all”. (Example: “0,30”)
- **schedule.month_of_year** (*string*) – Cron Months Of The Year to Run. Use “*” for “all”. (Example: “0,6”)
- **schedule.start_date_time** (*string*) – Start date time based on the time provided, timezone and if self.first_occur is true.
- **schedule.time** (*string*) – Time to start running on the policy’s SITE. REMEMBER: The Ngenea hub and the site may be running in different timezones
- **schedule.timezone** (*string*) – (required)
- **site** (*integer*) –
- **spaces[]** (*integer*) –

- **threads** (*integer*) – Number of threads to run policy on
- **triggers** (*string*) – (read only)
- **vision_metadata_query** (*object*) – Vision Metadata Query

PATCH /policies/{id}/

Parameters:

- **id** (*string*) –

Request JSON Object:

- **created** (*string*) – (read only)
- **enabled** (*boolean*) –
- **filesystem** (*string*) –
- **id** (*integer*) – (read only)
- **iscan_buckets** (*integer*) –
- **iscan_threads** (*integer*) –
- **max_files** (*integer*) – Bucket Size
- **name** (*string*) – (required)
- **node_class[]** (*string*) –
- **order.by** (*string*) –
- **order.id** (*integer*) – (read only)
- **order.reverse** (*boolean*) –
- **owner** (*integer*) –
- **policy_type** (*string*) –
- **schedule.day_of_month** (*string*) – Cron Days Of The Month to Run. Use “*” for “all”. (Example: “1,15”)
- **schedule.day_of_week** (*string*) – Cron Days Of The Week to Run. Use “*” for “all”. (Example: “0,5”)
- **schedule.first_occur** (*boolean*) – Whether the first occurrence will be at the stipulated time.
- **schedule.hour** (*string*) – Cron Hours to Run. Use “*” for “all”. (Example: “8,20”)
- **schedule.id** (*integer*) – (read only)
- **schedule.minute** (*string*) – Cron Minutes to Run. Use “*” for “all”. (Example: “0,30”)
- **schedule.month_of_year** (*string*) – Cron Months Of The Year to Run. Use “*” for “all”. (Example: “0,6”)
- **schedule.time** (*string*) – Time to start running on the policy’s SITE. REMEMBER: The Ngenea hub and the site may be running in different timezones
- **site** (*integer*) –
- **spaces[]** (*integer*) –
- **threads** (*integer*) – Number of threads to run policy on
- **triggers[].is_cloud** (*boolean*) –
- **triggers[].lower_threshold** (*number*) –
- **triggers[].max_utilisation** (*number*) –
- **triggers[].pool_name** (*string*) – (required)
- **triggers[].premigrate_threshold** (*number*) –
- **triggers[].upper_threshold** (*number*) –
- **vision_metadata_query** (*object*) – Vision Metadata Query

Status Codes:

- 201 Created –

Response JSON Object:

- **created** (*string*) – (read only)
- **enabled** (*boolean*) –
- **filesystem** (*string*) –
- **id** (*integer*) – (read only)
- **iscan_buckets** (*integer*) –
- **iscan_threads** (*integer*) –
- **max_files** (*integer*) – Bucket Size
- **name** (*string*) – (required)
- **node_class[]** (*string*) –
- **order** (*integer*) –
- **owner** (*integer*) –
- **policy_type** (*string*) –
- **run_immediately** (*boolean*) –
- **schedule.day_of_month** (*string*) – Cron Days Of The Month to Run. Use “*” for “all”. (Example: “1,15”)
- **schedule.day_of_week** (*string*) – Cron Days Of The Week to Run. Use “*” for “all”. (Example: “0,5”)
- **schedule.first_occur** (*boolean*) – Whether the first occurrence will be at the stipulated time.
- **schedule.hour** (*string*) – Cron Hours to Run. Use “*” for “all”. (Example: “8,20”)
- **schedule.id** (*integer*) – (read only)
- **schedule.minute** (*string*) – Cron Minutes to Run. Use “*” for “all”. (Example: “0,30”)
- **schedule.month_of_year** (*string*) – Cron Months Of The Year to Run. Use “*” for “all”. (Example: “0,6”)
- **schedule.time** (*string*) – Time to start running on the policy’s SITE. REMEMBER: The Ngenea hub and the site may be running in different timezones
- **site** (*integer*) –
- **spaces[]** (*integer*) –
- **threads** (*integer*) – Number of threads to run policy on
- **triggers[].id** (*integer*) – (read only)
- **triggers[].is_cloud** (*boolean*) –
- **triggers[].lower_threshold** (*number*) –
- **triggers[].max_utilisation** (*number*) –
- **triggers[].pool_name** (*string*) – (required)
- **triggers[].premigrate_threshold** (*number*) –
- **triggers[].upper_threshold** (*number*) –
- **vision_metadata_query** (*object*) – Vision Metadata Query

DELETE /policies/{id}/

Parameters:

- **id** (*string*) –

Status Codes:

- 204 No Content –

GET /policies/{id}/files/

Parameters:

- **id** (*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.
- **timeout** (*integer*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **created** (*string*) – (read only)
- **enabled** (*boolean*) –
- **filesystem** (*string*) –
- **id** (*integer*) – (read only)
- **iscan_buckets** (*integer*) –
- **iscan_threads** (*integer*) –
- **max_files** (*integer*) – Bucket Size
- **name** (*string*) – (required)
- **node_class[]** (*string*) –
- **order.by** (*string*) –
- **order.id** (*integer*) – (read only)
- **order.reverse** (*boolean*) –
- **owner** (*integer*) –
- **policy_type** (*string*) –
- **schedule.day_of_month** (*string*) – Cron Days Of The Month to Run. Use “*” for “all”. (Example: “1,15”)
- **schedule.day_of_week** (*string*) – Cron Days Of The Week to Run. Use “*” for “all”. (Example: “0,5”)
- **schedule.first_occur** (*boolean*) – Whether the first occurrence will be at the stipulated time.
- **schedule.hour** (*string*) – Cron Hours to Run. Use “*” for “all”. (Example: “8,20”)
- **schedule.id** (*integer*) – (read only)
- **schedule.minute** (*string*) – Cron Minutes to Run. Use “*” for “all”. (Example: “0,30”)
- **schedule.month_of_year** (*string*) – Cron Months Of The Year to Run. Use “*” for “all”. (Example: “0,6”)
- **schedule.time** (*string*) – Time to start running on the policy’s SITE. REMEMBER: The Ngenea hub and the site may be running in different timezones
- **site** (*integer*) –
- **spaces[]** (*integer*) –
- **threads** (*integer*) – Number of threads to run policy on
- **triggers[].is_cloud** (*boolean*) –
- **triggers[].lower_threshold** (*number*) –
- **triggers[].max_utilisation** (*number*) –

- **triggers[].pool_name** (*string*) – (required)
- **triggers[].premigrate_threshold** (*number*) –
- **triggers[].upper_threshold** (*number*) –
- **vision_metadata_query** (*object*) – Vision Metadata Query

POST /policies/{id}/run/

Parameters:

- **id** (*string*) –

Status Codes:

- 202 Accepted – Task id of the task fired for the associated request.

GET /queues/

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[].label** (*string*) – UI facing label for the queue
- **results[].last_heartbeat** (*string*) –
- **results[].name** (*string*) – Queue name. Must match [a-zA-Z0-9-_]+ (required)
- **results[].nodes** (*string*) – (read only)
- **results[].site** (*string*) – (read only)

GET /queues/{id}/

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **id** (*integer*) – (read only)
- **label** (*string*) – UI facing label for the queue
- **last_heartbeat** (*string*) –
- **name** (*string*) – Queue name. Must match [a-zA-Z0-9-_]+ (required)
- **nodes** (*string*) – (read only)
- **site** (*string*) – (read only)

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.
- **site_id** (*integer*) – Site ID
- **space_id** (*integer*) – Space ID

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[].discovery_options** (*object*) –
- **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*) – (required)
- **results[].space** (*string*) – (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*) –
- **discovery_options** (*object*) –
- **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*) – (required)
- **space** (*string*) –
- **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*) –
- **discovery_options** (*object*) –
- **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*) – (required)
- **space** (*string*) –
- **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*) –
- **discovery_options** (*object*) –
- **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*) – (required)
- **space** (*string*) – (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*) –
- **discovery_options** (*object*) –
- **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*) – (required)
- **space** (*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*) –
- **discovery_options** (*object*) –
- **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*) – (required)
- **space** (*string*) –

DELETE /schedules/{id}/

Parameters:

- **id** (*string*) –

Status Codes:

- 204 No Content –

POST /schedules/{id}/run/

Parameters:

- **id** (*string*) –

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*) –
- **discovery_options** (*object*) –
- **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*) – (required)
- **space** (*string*) – (required)
- **url** (*string*) – (read only)

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[].post_action** (*object*) –
- **results[].pre_action** (*object*) –
- **results[].queue** (*string*) – (required)
- **results[].site** (*string*) – (required)
- **results[].url** (*string*) – (read only)
- **results[].workflow** (*string*) – (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)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **queue** (*string*) – Queue name
- **site** (*string*) – (required)
- **workflow** (*string*) – (required)

Status Codes:

- 201 Created –

Response JSON Object:

- **fields** (*object*) – Mapping of path to operation for task usage
- **id** (*integer*) – (read only)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **queue** (*string*) – Queue name
- **site** (*string*) – (required)
- **workflow** (*string*) – (required)

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

Parameters:

- **parent_lookup_schedule** (*string*) –

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **fields** (*object*) – Mapping of path to operation for task usage
- **id** (*integer*) – (read only)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **queue** (*string*) – (required)
- **site** (*string*) – (required)
- **url** (*string*) – (read only)
- **workflow** (*string*) – (required)

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

Parameters:

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

Request JSON Object:

- **fields** (*object*) – Mapping of path to operation for task usage
- **id** (*integer*) – (read only)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **queue** (*string*) – (required)
- **site** (*string*) – (required)
- **url** (*string*) – (read only)
- **workflow** (*string*) – (required)

Status Codes:

- 200 OK –

Response JSON Object:

- **fields** (*object*) – Mapping of path to operation for task usage
- **id** (*integer*) – (read only)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **queue** (*string*) – (required)
- **site** (*string*) – (required)
- **url** (*string*) – (read only)
- **workflow** (*string*) – (required)

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

Parameters:

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

Status Codes:

- 204 No Content –

`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*) –

`PATCH /search/metadata/`

API endpoint for file search

Request JSON Object:

- **id** (*integer*) – (read only)
- **url** (*string*) – (read only)

Status Codes:

- 200 OK –

Response JSON Object:

- **id** (*integer*) – (read only)
- **url** (*string*) – (read only)

`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*) – 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.
- **sort** (*string*) – One or more fields to sort results by
- **group_by_name** (*boolean*) –
  When this url parameter is 'True' results are merged based on matching file name. e.g.

```
[
  {
    data: [
      {site: site1, path: /mmfs1, ...},
      {site: site2, path: /mmfs1, ...}
    ]
  },
  {
    foo: [
      {site: site1, path: /mmfs1/data, ...}
      # not present on site2
    ]
  }
}
```

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)
- **proxies** (*object*) – File proxies (thumbnails, etc.)
- **site** (*string*) – Site Name

DELETE /search/{id}/

API endpoint for file search

Parameters:

- **id** (*string*) –

Status Codes:

- 204 No Content –

GET /servers/

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[].list_of_interfaces** (*object*) – List of interfaces of this server
- **results[].name** (*string*) – Server name (required)
- **results[].site** (*integer*) – Site correlated with this server (required)

`GET /servers/{id}/`

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **list_of_interfaces** (*object*) – List of interfaces of this server
- **name** (*string*) – Server name (required)
- **site** (*integer*) – Site correlated with this server (required)

`GET /settings/`

API to get a list of global settings.

Status Codes:

- 200 OK –

Response JSON Object:

- **[].desc** (*string*) – (read only)
- **[].id** (*integer*) – (read only)
- **[].key** (*string*) – (required)
- **[].value** (*object*) –

`PATCH /settings/`

API to set a setting or a group of global settings. A group of settings should be sent in key-value schema. Requested data will be stored in the database, and then sent to each site. Terminology Setting - An individual setting e.g. ngenea:realm

Notes on use:

- Non-global settings cannot be set on this endpoint either alone or with global settings
- Setting a key to None will delete it from the database and set null on all active sites

e.g. {“values”: [{“key”: “ngeneahsm_targets:devaws:Storage”, “value”: None}]}

Request JSON Object:

- **values[].desc** (*string*) – (read only)
- **values[].id** (*integer*) – (read only)
- **values[].key** (*string*) – (required)
- **values[].value** (*object*) –

Status Codes:

- 202 Accepted – Task id of the task fired for the associated request per site.

`POST /settings/sync/`

API to sync global settings on each site Stored data will be sent to the given site or to each site if not given.

Query Parameters:

- **site** (*string*) – Site to run the sync operation on if provided, otherwise all sites

Status Codes:

- 202 Accepted – Task id of the task fired for the associated request per site.

`GET /settingstasks/`

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** (*array*) – Task states
- **job_id** (*integer*) – Job ID

Status Codes:

- 200 OK –

Response JSON Object:

- **count** (*integer*) – (required)
- **next** (*string*) –
- **previous** (*string*) –
- **results[].id** (*integer*) – (read only)
- **results[].job** (*integer*) –
- **results[].site** (*string*) – (required)
- **results[].started** (*string*) – Time that the task started running
- **results[].state** (*string*) –
- **results[].task_id** (*string*) – (read only)
- **results[].tasktype** (*string*) – (read only)
- **results[].url** (*string*) – (read only)

`GET /settingstasks/{id}/`

API endpoint for viewing tasks.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **completed** (*string*) – Time of the task completion
- **friendly_name** (*string*) – Text to be used to describe what this task is doing.
- **id** (*integer*) – (read only)
- **job** (*integer*) –
- **request** (*string*) – (read only)
- **results** (*string*) – (read only)
- **runtime** (*string*) – (read only)
- **site** (*string*) – (required)
- **started** (*string*) – Time that the task started running
- **state** (*string*) –
- **task_id** (*string*) – (read only)
- **tasktype** (*string*) – (read only)
- **url** (*string*) – (read only)

`GET /shares/nfs/`

Status Codes:

- 200 OK –

Response JSON Object:

- **[].fsid** (*integer*) –
- **[].id** (*integer*) – (read only)
- **[].name** (*string*) – Note: This name is for identification only. NFSClients don't have names in remote representations
- **[].path** (*string*) – (required)
- **[].sites[].clients[].advanced_settings** (*object*) –
- **[].sites[].clients[].anonymous_group_id** (*integer*) –
- **[].sites[].clients[].anonymous_user_id** (*integer*) –
- **[].sites[].clients[].asynchronous** (*boolean*) –
- **[].sites[].clients[].force_all_user_id** (*boolean*) –
- **[].sites[].clients[].force_root_user_id** (*boolean*) –
- **[].sites[].clients[].hosts** (*string*) – (required)
- **[].sites[].clients[].id** (*integer*) – (read only)
- **[].sites[].clients[].insecure_ports** (*boolean*) –
- **[].sites[].clients[].nfs_share_site** (*integer*) –
- **[].sites[].clients[].read_only** (*boolean*) –
- **[].sites[].clients[].subtree_check** (*boolean*) –
- **[].sites[].clients[].uuid** (*string*) – (read only)
- **[].sites[].clients[].write_delay** (*boolean*) –
- **[].sites[].name** (*string*) – (required)
- **[].space** (*integer*) – (required)
- **[].uuid** (*string*) – (read only)

`POST /shares/nfs/`

Request JSON Object:

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

- **name** (*string*) – Note: This name is for identification only. NFSClients don't have names in remote representations
- **path** (*string*) – (required)
- **sites[].clients[].advanced_settings** (*object*) –
- **sites[].clients[].anonymous_group_id** (*integer*) –
- **sites[].clients[].anonymous_user_id** (*integer*) –
- **sites[].clients[].asynchronous** (*boolean*) –
- **sites[].clients[].force_all_user_id** (*boolean*) –
- **sites[].clients[].force_root_user_id** (*boolean*) –
- **sites[].clients[].hosts** (*string*) – (required)
- **sites[].clients[].id** (*integer*) – (read only)
- **sites[].clients[].insecure_ports** (*boolean*) –
- **sites[].clients[].nfs_share_site** (*integer*) –
- **sites[].clients[].read_only** (*boolean*) –
- **sites[].clients[].subtree_check** (*boolean*) –
- **sites[].clients[].uuid** (*string*) – (read only)
- **sites[].clients[].write_delay** (*boolean*) –
- **sites[].name** (*string*) – (required)
- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

Status Codes:

- 201 Created –

Response JSON Object:

- **fsid** (*integer*) –
- **id** (*integer*) – (read only)
- **name** (*string*) – Note: This name is for identification only. NFSClients don't have names in remote representations
- **path** (*string*) – (required)
- **sites[].clients[].advanced_settings** (*object*) –
- **sites[].clients[].anonymous_group_id** (*integer*) –
- **sites[].clients[].anonymous_user_id** (*integer*) –
- **sites[].clients[].asynchronous** (*boolean*) –
- **sites[].clients[].force_all_user_id** (*boolean*) –
- **sites[].clients[].force_root_user_id** (*boolean*) –
- **sites[].clients[].hosts** (*string*) – (required)
- **sites[].clients[].id** (*integer*) – (read only)
- **sites[].clients[].insecure_ports** (*boolean*) –
- **sites[].clients[].nfs_share_site** (*integer*) –
- **sites[].clients[].read_only** (*boolean*) –
- **sites[].clients[].subtree_check** (*boolean*) –
- **sites[].clients[].uuid** (*string*) – (read only)
- **sites[].clients[].write_delay** (*boolean*) –
- **sites[].name** (*string*) – (required)
- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

`GET /shares/nfs/{id}/`

Parameters:

- **id** (*integer*) – A unique integer value identifying this nfs share.

Status Codes:

- 200 OK –

Response JSON Object:

- **fsid** (*integer*) –
- **id** (*integer*) – (read only)
- **name** (*string*) – Note: This name is for identification only. NFSClients don't have names in remote representations
- **path** (*string*) – (required)
- **sites[].clients[].advanced_settings** (*object*) –
- **sites[].clients[].anonymous_group_id** (*integer*) –
- **sites[].clients[].anonymous_user_id** (*integer*) –
- **sites[].clients[].asynchronous** (*boolean*) –
- **sites[].clients[].force_all_user_id** (*boolean*) –
- **sites[].clients[].force_root_user_id** (*boolean*) –
- **sites[].clients[].hosts** (*string*) – (required)
- **sites[].clients[].id** (*integer*) – (read only)
- **sites[].clients[].insecure_ports** (*boolean*) –
- **sites[].clients[].nfs_share_site** (*integer*) –
- **sites[].clients[].read_only** (*boolean*) –
- **sites[].clients[].subtree_check** (*boolean*) –
- **sites[].clients[].uuid** (*string*) – (read only)
- **sites[].clients[].write_delay** (*boolean*) –
- **sites[].name** (*string*) – (required)
- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

`PATCH /shares/nfs/{id}/`

Parameters:

- **id** (*integer*) – A unique integer value identifying this nfs share.

Request JSON Object:

- **fsid** (*integer*) –
- **id** (*integer*) – (read only)
- **name** (*string*) – Note: This name is for identification only. NFSClients don't have names in remote representations
- **path** (*string*) – (required)
- **sites[].clients[].advanced_settings** (*object*) –
- **sites[].clients[].anonymous_group_id** (*integer*) –
- **sites[].clients[].anonymous_user_id** (*integer*) –
- **sites[].clients[].asynchronous** (*boolean*) –
- **sites[].clients[].force_all_user_id** (*boolean*) –
- **sites[].clients[].force_root_user_id** (*boolean*) –
- **sites[].clients[].hosts** (*string*) – (required)
- **sites[].clients[].id** (*integer*) – (read only)
- **sites[].clients[].insecure_ports** (*boolean*) –
- **sites[].clients[].nfs_share_site** (*integer*) –
- **sites[].clients[].read_only** (*boolean*) –
- **sites[].clients[].subtree_check** (*boolean*) –
- **sites[].clients[].uuid** (*string*) – (read only)
- **sites[].clients[].write_delay** (*boolean*) –
- **sites[].name** (*string*) – (required)

- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

Status Codes:

- 200 OK –

Response JSON Object:

- **fsid** (*integer*) –
- **id** (*integer*) – (read only)
- **name** (*string*) – Note: This name is for identification only. NFSClients don't have names in remote representations
- **path** (*string*) – (required)
- **sites[].clients[].advanced_settings** (*object*) –
- **sites[].clients[].anonymous_group_id** (*integer*) –
- **sites[].clients[].anonymous_user_id** (*integer*) –
- **sites[].clients[].asynchronous** (*boolean*) –
- **sites[].clients[].force_all_user_id** (*boolean*) –
- **sites[].clients[].force_root_user_id** (*boolean*) –
- **sites[].clients[].hosts** (*string*) – (required)
- **sites[].clients[].id** (*integer*) – (read only)
- **sites[].clients[].insecure_ports** (*boolean*) –
- **sites[].clients[].nfs_share_site** (*integer*) –
- **sites[].clients[].read_only** (*boolean*) –
- **sites[].clients[].subtree_check** (*boolean*) –
- **sites[].clients[].uuid** (*string*) – (read only)
- **sites[].clients[].write_delay** (*boolean*) –
- **sites[].name** (*string*) – (required)
- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

`DELETE /shares/nfs/{id}/`

Parameters:

- **id** (*integer*) – A unique integer value identifying this nfs share.

Status Codes:

- 204 No Content –

`GET /shares/samba/`

Status Codes:

- 200 OK –

Response JSON Object:

- **[].id** (*integer*) – (read only)
- **[].name** (*string*) – (required)
- **[].on_sites** (*object*) –
- **[].path** (*string*) –
- **[].sites** (*string*) – (read only)
- **[].space** (*integer*) – (required)
- **[].uuid** (*string*) – (read only)

`POST /shares/samba/`

Request JSON Object:

- **id** (*integer*) – (read only)
- **name** (*string*) – (required)

- **on_sites** (*object*) –
- **path** (*string*) –
- **sites** (*string*) – (read only)
- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

Status Codes:

- 201 Created –

Response JSON Object:

- **id** (*integer*) – (read only)
- **name** (*string*) – (required)
- **on_sites** (*object*) –
- **path** (*string*) –
- **sites** (*string*) – (read only)
- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

`GET /shares/samba/{id}/`

Parameters:

- **id** (*integer*) – A unique integer value identifying this samba share.

Status Codes:

- 200 OK –

Response JSON Object:

- **id** (*integer*) – (read only)
- **name** (*string*) – (required)
- **on_sites** (*object*) –
- **path** (*string*) –
- **sites** (*string*) – (read only)
- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

`PATCH /shares/samba/{id}/`

Parameters:

- **id** (*integer*) – A unique integer value identifying this samba share.

Request JSON Object:

- **id** (*integer*) – (read only)
- **name** (*string*) – (required)
- **on_sites** (*object*) –
- **path** (*string*) –
- **sites** (*string*) – (read only)
- **space** (*integer*) – (required)
- **uuid** (*string*) – (read only)

Status Codes:

- 200 OK –

Response JSON Object:

- **id** (*integer*) – (read only)
- **name** (*string*) – (required)
- **on_sites** (*object*) –
- **path** (*string*) –
- **sites** (*string*) – (read only)
- **space** (*integer*) – (required)

- **uuid** (*string*) – (read only)

DELETE /shares/samba/{id}/

Parameters:

- **id** (*integer*) – A unique integer value identifying this samba share.

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.
- **site_id** (*integer*) – Site ID
- **datastore_id** (*integer*) – Data store ID

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[].am_i_configured** (*boolean*) – Boolean field determining if UI has been configured for the site
- **results[].bandwidth** (*integer*) – speed for site (in Mb/s)
- **results[].color[]** (*integer*) –
- **results[].directories_count** (*integer*) – How many directories this site has
- **results[].editable** (*string*) – (read only)
- **results[].files_count** (*integer*) – How many files this site has
- **results[].files_hydrated_count** (*integer*) – How many files hydrated this site has
- **results[].files_stubbed_count** (*integer*) – How many files stubbed this site has
- **results[].id** (*integer*) – (read only)
- **results[].iris_enabled** (*boolean*) – Is Iris enabled on this site
- **results[].label** (*string*) – Human readable name for this site
- **results[].maintenance** (*string*) – (read only)
- **results[].maintenance_mode_active** (*string*) – (read only)
- **results[].name** (*string*) – Site Name. Must match [a-zA-Z0-9-_]+ (required)
- **results[].public_url** (*string*) – The base URL by which this site can be reached
- **results[].ready_to_configure** (*boolean*) – Boolean to check if site is ready to configure
- **results[].shortcode** (*string*) – Shortcode
- **results[].spaces[]** (*string*) –
- **results[].timezone** (*string*) –
- **results[].type** (*string*) – Type indicating where the site (pixstor) is deployed
- **results[].url** (*string*) – (read only)

`POST /sites/`

API endpoint for managing sites.

Request JSON Object:

- **am_i_configured** (*boolean*) – Boolean field determining if UI has been configured for the site
- **bandwidth** (*integer*) – speed for site (in Mb/s)
- **color[]** (*integer*) –
- **directories_count** (*integer*) – How many directories this site has

- **elasticsearch_url** (*string*) – URL of the Elasticsearch server to use for the Analytics search backend on this site
- **enable_auto_file_batch_sizing** (*boolean*) – If optimal batch size should be determined at runtime.
- **exclude** (*object*) – Global workflow excludes for this site
- **file_batch_gb** (*integer*) – File batch GB
- **file_batch_size** (*integer*) – File batch size
- **files_count** (*integer*) – How many files this site has
- **files_hydrated_count** (*integer*) – How many files hydrated this site has
- **files_stubbed_count** (*integer*) – How many files stubbed this site has
- **gpfs_iscan_buckets** (*integer*) – Number of buckets to use for the gpfs snapdiff policy
- **gpfs_iscan_threads** (*integer*) – Number of threads to use for the gpfs snapdiff policy
- **id** (*integer*) – (read only)
- **include** (*object*) – Global workflow includes for this site
- **iris_enabled** (*boolean*) – Is Iris enabled on this site
- **label** (*string*) – Human readable name for this site
- **lock_threshold** (*integer*) – Threshold for soft locking snapshot rotations
- **name** (*string*) – Site Name. Must match [a-zA-Z0-9-_]+ (required)
- **pixstor_search_url** (*string*) – The base URL for querying the PixStor API
- **public_url** (*string*) – The base URL by which this site can be reached
- **ready_to_configure** (*boolean*) – Boolean to check if site is ready to configure
- **shortcode** (*string*) – Shortcode (required)
- **spaces[]** (*string*) –
- **timezone** (*string*) –
- **type** (*string*) – Type indicating where the site (pixstor) is deployed

Status Codes:

- 201 Created –

Response JSON Object:

- **am_i_configured** (*boolean*) – Boolean field determining if UI has been configured for the site
- **bandwidth** (*integer*) – speed for site (in Mb/s)
- **color[]** (*integer*) –
- **directories_count** (*integer*) – How many directories this site has
- **editable** (*string*) – (read only)
- **elasticsearch_url** (*string*) – URL of the Elasticsearch server to use for the Analytics search backend on this site
- **enable_auto_file_batch_sizing** (*boolean*) – If optimal batch size should be determined at runtime.
- **exclude** (*object*) – Global workflow excludes for this site
- **file_batch_gb** (*integer*) – File batch GB
- **file_batch_size** (*integer*) – File batch size
- **files_count** (*integer*) – How many files this site has
- **files_hydrated_count** (*integer*) – How many files hydrated this site has
- **files_stubbed_count** (*integer*) – How many files stubbed this site has
- **gpfs_iscan_buckets** (*integer*) – Number of buckets to use for the gpfs snapdiff policy
- **gpfs_iscan_threads** (*integer*) – Number of threads to use for the gpfs snapdiff policy

- **id** (*integer*) – (read only)
- **include** (*object*) – Global workflow includes for this site
- **iris_enabled** (*boolean*) – Is Iris enabled on this site
- **label** (*string*) – Human readable name for this site
- **lock_threshold** (*integer*) – Threshold for soft locking snapshot rotations
- **maintenance** (*string*) – (read only)
- **maintenance_mode_active** (*string*) – (read only)
- **name** (*string*) – Site Name. Must match [a-zA-Z0-9-_]+ (required)
- **pixstor_search_url** (*string*) – The base URL for querying the PixStor API
- **public_url** (*string*) – The base URL by which this site can be reached
- **ready_to_configure** (*boolean*) – Boolean to check if site is ready to configure
- **shortcode** (*string*) – Shortcode
- **spaces[].id** (*integer*) – (read only)
- **spaces[].name** (*string*) – Space Name (required)
- **spaces[].url** (*string*) – (read only)
- **timezone** (*string*) –
- **type** (*string*) – Type indicating where the site (pixstor) is deployed
- **url** (*string*) – (read only)

`GET /sites/{id}/`

API endpoint for managing sites.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **am_i_configured** (*boolean*) – Boolean field determining if UI has been configured for the site
- **bandwidth** (*integer*) – speed for site (in Mb/s)
- **color[]** (*integer*) –
- **directories_count** (*integer*) – How many directories this site has
- **editable** (*string*) – (read only)
- **elasticsearch_url** (*string*) – URL of the Elasticsearch server to use for the Analytics search backend on this site
- **enable_auto_file_batch_sizing** (*boolean*) – If optimal batch size should be determined at runtime.
- **exclude** (*object*) – Global workflow excludes for this site
- **file_batch_gb** (*integer*) – File batch GB
- **file_batch_size** (*integer*) – File batch size
- **files_count** (*integer*) – How many files this site has
- **files_hydrated_count** (*integer*) – How many files hydrated this site has
- **files_stubbed_count** (*integer*) – How many files stubbed this site has
- **gpfs_iscan_buckets** (*integer*) – Number of buckets to use for the gpfs snapdiff policy
- **gpfs_iscan_threads** (*integer*) – Number of threads to use for the gpfs snapdiff policy
- **id** (*integer*) – (read only)
- **include** (*object*) – Global workflow includes for this site
- **iris_enabled** (*boolean*) – Is Iris enabled on this site

- **label** (*string*) – Human readable name for this site
- **lock_threshold** (*integer*) – Threshold for soft locking snapshot rotations
- **maintenance** (*string*) – (read only)
- **maintenance_mode_active** (*string*) – (read only)
- **name** (*string*) – Site Name. Must match [a-zA-Z0-9-_]+ (required)
- **pixstor_search_url** (*string*) – The base URL for querying the PixStor API
- **public_url** (*string*) – The base URL by which this site can be reached
- **ready_to_configure** (*boolean*) – Boolean to check if site is ready to configure
- **shortcode** (*string*) – Shortcode
- **spaces[].id** (*integer*) – (read only)
- **spaces[].name** (*string*) – Space Name (required)
- **spaces[].url** (*string*) – (read only)
- **timezone** (*string*) –
- **type** (*string*) – Type indicating where the site (pixstor) is deployed
- **url** (*string*) – (read only)

`PATCH /sites/{id}/`

API endpoint for managing sites.

Parameters:

- **id** (*string*) –

Request JSON Object:

- **am_i_configured** (*boolean*) – Boolean field determining if UI has been configured for the site
- **bandwidth** (*integer*) – speed for site (in Mb/s)
- **color[]** (*integer*) –
- **directories_count** (*integer*) – How many directories this site has
- **elasticsearch_url** (*string*) – URL of the Elasticsearch server to use for the Analytics search backend on this site
- **enable_auto_file_batch_sizing** (*boolean*) – If optimal batch size should be determined at runtime.
- **exclude** (*object*) – Global workflow excludes for this site
- **file_batch_gb** (*integer*) – File batch GB
- **file_batch_size** (*integer*) – File batch size
- **files_count** (*integer*) – How many files this site has
- **files_hydrated_count** (*integer*) – How many files hydrated this site has
- **files_stubbed_count** (*integer*) – How many files stubbed this site has
- **gpfs_iscan_buckets** (*integer*) – Number of buckets to use for the gpfs snapdiff policy
- **gpfs_iscan_threads** (*integer*) – Number of threads to use for the gpfs snapdiff policy
- **id** (*integer*) – (read only)
- **include** (*object*) – Global workflow includes for this site
- **iris_enabled** (*boolean*) – Is Iris enabled on this site
- **label** (*string*) – Human readable name for this site
- **lock_threshold** (*integer*) – Threshold for soft locking snapshot rotations
- **maintenance** (*object*) –
- **name** (*string*) – Site Name. Must match [a-zA-Z0-9-_]+
- **pixstor_search_url** (*string*) – The base URL for querying the PixStor API
- **public_url** (*string*) – The base URL by which this site can be reached

- **ready_to_configure** (*boolean*) – Boolean to check if site is ready to configure
- **shortcode** (*string*) – Shortcode
- **spaces[]** (*string*) –
- **timezone** (*string*) –
- **type** (*string*) – Type indicating where the site (pixstor) is deployed

Status Codes:

- 200 OK –

Response JSON Object:

- **am_i_configured** (*boolean*) – Boolean field determining if UI has been configured for the site
- **bandwidth** (*integer*) – speed for site (in Mb/s)
- **color[]** (*integer*) –
- **directories_count** (*integer*) – How many directories this site has
- **editable** (*string*) – (read only)
- **elasticsearch_url** (*string*) – URL of the Elasticsearch server to use for the Analytics search backend on this site
- **enable_auto_file_batch_sizing** (*boolean*) – If optimal batch size should be determined at runtime.
- **exclude** (*object*) – Global workflow excludes for this site
- **file_batch_gb** (*integer*) – File batch GB
- **file_batch_size** (*integer*) – File batch size
- **files_count** (*integer*) – How many files this site has
- **files_hydrated_count** (*integer*) – How many files hydrated this site has
- **files_stubbed_count** (*integer*) – How many files stubbed this site has
- **gpfs_iscan_buckets** (*integer*) – Number of buckets to use for the gpfs snapdiff policy
- **gpfs_iscan_threads** (*integer*) – Number of threads to use for the gpfs snapdiff policy
- **id** (*integer*) – (read only)
- **include** (*object*) – Global workflow includes for this site
- **iris_enabled** (*boolean*) – Is Iris enabled on this site
- **label** (*string*) – Human readable name for this site
- **lock_threshold** (*integer*) – Threshold for soft locking snapshot rotations
- **maintenance** (*string*) – (read only)
- **maintenance_mode_active** (*string*) – (read only)
- **name** (*string*) – Site Name. Must match [a-zA-Z0-9-_]+ (required)
- **pixstor_search_url** (*string*) – The base URL for querying the PixStor API
- **public_url** (*string*) – The base URL by which this site can be reached
- **ready_to_configure** (*boolean*) – Boolean to check if site is ready to configure
- **shortcode** (*string*) – Shortcode
- **spaces[].id** (*integer*) – (read only)
- **spaces[].name** (*string*) – Space Name (required)
- **spaces[].url** (*string*) – (read only)
- **timezone** (*string*) –
- **type** (*string*) – Type indicating where the site (pixstor) is deployed
- **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:

- **am_i_configured** (*boolean*) – Boolean field determining if UI has been configured for the site
- **bandwidth** (*integer*) – speed for site (in Mb/s)
- **color[]** (*integer*) –
- **directories_count** (*integer*) – How many directories this site has
- **editable** (*string*) – (read only)
- **files_count** (*integer*) – How many files this site has
- **files_hydrated_count** (*integer*) – How many files hydrated this site has
- **files_stubbed_count** (*integer*) – How many files stubbed this site has
- **id** (*integer*) – (read only)
- **iris_enabled** (*boolean*) – Is Iris enabled on this site
- **label** (*string*) – Human readable name for this site
- **maintenance** (*string*) – (read only)
- **maintenance_mode_active** (*string*) – (read only)
- **name** (*string*) – Site Name. Must match [a-zA-Z0-9-_]+ (required)
- **public_url** (*string*) – The base URL by which this site can be reached
- **ready_to_configure** (*boolean*) – Boolean to check if site is ready to configure
- **shortcode** (*string*) – Shortcode
- **spaces[]** (*string*) –
- **timezone** (*string*) –
- **type** (*string*) – Type indicating where the site (pixstor) is deployed
- **url** (*string*) – (read only)

`POST /sites/{id}/refresh/`

API endpoint for managing sites.

Parameters:

- **id** (*string*) –

Status Codes:

- 201 Created –

Response JSON Object:

- **am_i_configured** (*boolean*) – Boolean field determining if UI has been configured for the site
- **bandwidth** (*integer*) – speed for site (in Mb/s)
- **color[]** (*integer*) –
- **directories_count** (*integer*) – How many directories this site has
- **editable** (*string*) – (read only)

- **files_count** (*integer*) – How many files this site has
- **files_hydrated_count** (*integer*) – How many files hydrated this site has
- **files_stubbed_count** (*integer*) – How many files stubbed this site has
- **id** (*integer*) – (read only)
- **iris_enabled** (*boolean*) – Is Iris enabled on this site
- **label** (*string*) – Human readable name for this site
- **maintenance** (*string*) – (read only)
- **maintenance_mode_active** (*string*) – (read only)
- **name** (*string*) – Site Name. Must match [a-zA-Z0-9-_]+ (required)
- **public_url** (*string*) – The base URL by which this site can be reached
- **ready_to_configure** (*boolean*) – Boolean to check if site is ready to configure
- **shortcode** (*string*) – Shortcode
- **spaces[]** (*string*) –
- **timezone** (*string*) –
- **type** (*string*) – Type indicating where the site (pixstor) is deployed
- **url** (*string*) – (read only)

`GET /sites/{id}/settings/`

API endpoint for managing sites.

Parameters:
- **id** (*string*) –

Query Parameters:
- **flat** (*boolean*) – Return flat dict

Status Codes:
- 200 OK –

Response JSON Object:
- **[].desc** (*string*) – (read only)
- **[].id** (*integer*) – (read only)
- **[].key** (*string*) – (required)
- **[].value** (*object*) –

`PATCH /sites/{id}/settings/`

API to set a setting or a group of settings associated with a site. A group of settings should be sent in key-value schema. Terminology Setting - An individual setting e.g. sambda:realm

Parameters:
- **id** (*string*) –

Request JSON Object:
- **values[].desc** (*string*) – (read only)
- **values[].id** (*integer*) – (read only)
- **values[].key** (*string*) – (required)
- **values[].value** (*object*) –

Status Codes:
- 202 Accepted – Task id of the task fired for the associated request.

`POST /sites/{id}/settings/refresh/`

Refresh settings for a given site under management. This API will also create settings in the hub for this site if they don’t already exist.

Parameters:

- **id** (*string*) –

Status Codes:

- 202 Accepted – Task id of the task fired for the associated request.

GET /spaces/

Model View Set for filebrowser's Space model.

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.
- **site_id** (*integer*) – Site ID
- **search_keyword** (*string*) – Keyword to filter by

Status Codes:

- 200 OK –

Response JSON Object:

- **count** (*integer*) – (required)
- **next** (*string*) –
- **previous** (*string*) –
- **results[].color[]** (*integer*) –
- **results[].created** (*string*) – Time of the job creation (read only)
- **results[].editable** (*string*) – (read only)
- **results[].id** (*integer*) – (read only)
- **results[].immutable** (*boolean*) – Space that once created cannot be deleted or updated. Usually the default space.
- **results[].is_ready** (*string*) – (read only)
- **results[].mountpoint** (*string*) – Mount point (required)
- **results[].name** (*string*) – Space Name (required)
- **results[].nested_shares** (*string*) – (read only)
- **results[].on_sites** (*object*) –
- **results[].permission_mode** (*string*) – Ability to set the permission mode on a space
- **results[].relationships** (*string*) – (read only)
- **results[].sites** (*string*) – (read only)
- **results[].size** (*integer*) – Space size quota in bytes. No value specified, means as large as your system allows.
- **results[].snapshot_schedule.duration** (*string*) – (required)
- **results[].snapshot_schedule.frequency** (*string*) – (required)
- **results[].snapshot_schedule.id** (*integer*) – (read only)
- **results[].snapshot_schedule.space** (*integer*) –
- **results[].snapshot_schedule.time** (*string*) – (required)
- **results[].url** (*string*) – (read only)
- **results[].uuid** (*string*) – (read only)

POST /spaces/

Model View Set for filebrowser's Space model.

Request JSON Object:

- **color[]** (*integer*) –
- **created** (*string*) – Time of the job creation (read only)
- **mountpoint** (*string*) – (required)
- **name** (*string*) – (required)
- **permission_mode** (*string*) – Ability to set the permission mode on a space
- **sites[].filesystem** (*string*) – (read only)
- **sites[].id** (*string*) – (read only)
- **sites[].iris** (*string*) – (read only)
- **sites[].label** (*string*) – (required)
- **sites[].name** (*string*) – (required)
- **sites[].pool** (*string*) – (read only)
- **sites[].usage[]** (*integer*) –
- **size** (*integer*) – Space size quota in bytes. No value specified, means as large as your system allows.
- **snapshot_schedule.duration** (*string*) – (required)
- **snapshot_schedule.frequency** (*string*) – (required)
- **snapshot_schedule.id** (*integer*) – (read only)
- **snapshot_schedule.space** (*integer*) –
- **snapshot_schedule.time** (*string*) – (required)

Status Codes:

- 201 Created –

Response JSON Object:

- **color[]** (*integer*) –
- **created** (*string*) – Time of the job creation (read only)
- **editable** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **immutable** (*boolean*) – Space that once created cannot be deleted or updated. Usually the default space.
- **is_ready** (*string*) – (read only)
- **mountpoint** (*string*) – Mount point (required)
- **name** (*string*) – Space Name (required)
- **nested_shares** (*string*) – (read only)
- **on_sites** (*object*) –
- **permission_mode** (*string*) – Ability to set the permission mode on a space
- **relationships** (*string*) – (read only)
- **sites** (*string*) – (read only)
- **size** (*integer*) – Space size quota in bytes. No value specified, means as large as your system allows.
- **snapshot_schedule.duration** (*string*) – (required)
- **snapshot_schedule.frequency** (*string*) – (required)
- **snapshot_schedule.id** (*integer*) – (read only)
- **snapshot_schedule.space** (*integer*) –
- **snapshot_schedule.time** (*string*) – (required)
- **url** (*string*) – (read only)
- **uuid** (*string*) – (read only)

`GET /spaces/{id}/`

Model View Set for filebrowser's Space model.

Parameters:

- **id** (*string*) –

Query Parameters:

- **site_id** (*integer*) – Site ID
- **search_keyword** (*string*) – Keyword to filter by

Status Codes:

- 200 OK –

Response JSON Object:

- **color[]** (*integer*) –
- **created** (*string*) – Time of the job creation (read only)
- **editable** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **immutable** (*boolean*) – Space that once created cannot be deleted or updated. Usually the default space.
- **is_ready** (*string*) – (read only)
- **mountpoint** (*string*) – Mount point (required)
- **name** (*string*) – Space Name (required)
- **nested_shares** (*string*) – (read only)
- **on_sites** (*object*) –
- **permission_mode** (*string*) – Ability to set the permission mode on a space
- **relationships** (*string*) – (read only)
- **sites** (*string*) – (read only)
- **size** (*integer*) – Space size quota in bytes. No value specified, means as large as your system allows.
- **snapshot_schedule.duration** (*string*) – (required)
- **snapshot_schedule.frequency** (*string*) – (required)
- **snapshot_schedule.id** (*integer*) – (read only)
- **snapshot_schedule.space** (*integer*) –
- **snapshot_schedule.time** (*string*) – (required)
- **url** (*string*) – (read only)
- **uuid** (*string*) – (read only)

`PATCH /spaces/{id}/`

Model View Set for filebrowser's Space model.

Parameters:

- **id** (*string*) –

Request JSON Object:

- **color[]** (*integer*) –
- **created** (*string*) – Time of the job creation (read only)
- **mountpoint** (*string*) – Mount point (read only)
- **name** (*string*) –
- **permission_mode** (*string*) – Ability to set the permission mode on a space
- **sites[].filesystem** (*string*) – (read only)
- **sites[].id** (*string*) – (read only)
- **sites[].iris** (*string*) – (read only)
- **sites[].label** (*string*) – (required)
- **sites[].name** (*string*) – (required)

- **sites[].pool** (*string*) – (read only)
- **sites[].usage[]** (*integer*) –
- **size** (*integer*) – Space size quota in bytes. No value specified, means as large as your system allows.
- **snapshot_schedule.duration** (*string*) – (required)
- **snapshot_schedule.frequency** (*string*) – (required)
- **snapshot_schedule.id** (*integer*) – (read only)
- **snapshot_schedule.space** (*integer*) –
- **snapshot_schedule.time** (*string*) – (required)

Status Codes:

- 200 OK –

Response JSON Object:

- **color[]** (*integer*) –
- **created** (*string*) – Time of the job creation (read only)
- **editable** (*string*) – (read only)
- **id** (*integer*) – (read only)
- **immutable** (*boolean*) – Space that once created cannot be deleted or updated. Usually the default space.
- **is_ready** (*string*) – (read only)
- **mountpoint** (*string*) – Mount point (required)
- **name** (*string*) – Space Name (required)
- **nested_shares** (*string*) – (read only)
- **on_sites** (*object*) –
- **permission_mode** (*string*) – Ability to set the permission mode on a space
- **relationships** (*string*) – (read only)
- **sites** (*string*) – (read only)
- **size** (*integer*) – Space size quota in bytes. No value specified, means as large as your system allows.
- **snapshot_schedule.duration** (*string*) – (required)
- **snapshot_schedule.frequency** (*string*) – (required)
- **snapshot_schedule.id** (*integer*) – (read only)
- **snapshot_schedule.space** (*integer*) –
- **snapshot_schedule.time** (*string*) – (required)
- **url** (*string*) – (read only)
- **uuid** (*string*) – (read only)

`DELETE /spaces/{id}/`

Model View Set for filebrowser's Space model.

Parameters:

- **id** (*string*) –

Status Codes:

- 204 No Content –

`GET /spaces/{id}/files/`

Model View Set for filebrowser's Space model.

Parameters:

- **id** (*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.
- **site** (*string*) – Site to run the stat operation on if provided, otherwise all space's sites
- **path** (*string*) – Path to retrieve stat
- **details** (*boolean*) – Retrieve all the details of the children files
- **restricted** (*boolean*) – Include restricted objects
- **check_available_sites** (*boolean*) – Check only available sites
- **custom_refresh_period** (*integer*) – Custom time in seconds to invalidate the DB cache
- **custom_timeout** (*integer*) – Custom time in seconds to invalidate the task
- **raise_exception** (*boolean*) – Raise exception if timeout
- **type** (*string*) – item type

Status Codes:

- 200 OK –

Response JSON Object:

- **atime** (*string*) –
- **children** (*string*) – (read only)
- **ctime** (*string*) –
- **data_size** (*integer*) –
- **deleted** (*boolean*) –
- **disk_usage** (*integer*) –
- **error** (*string*) – Failed to fetch file data error
- **image_preview** (*string*) – Relative URL of an image preview
- **metadata** (*object*) –
- **mtime** (*string*) –
- **name** (*string*) – Name of of the fs object
- **path** (*string*) – Path of the object (required)
- **path_depth** (*integer*) –
- **remote_locations** (*object*) –
- **restricted** (*boolean*) –
- **site** (*string*) – (read only)
- **size** (*integer*) –
- **status** (*string*) – Migration status
- **thumbnail** (*string*) – Relative URL of an object thumbnail
- **time_refreshed** (*string*) –
- **total_files** (*integer*) – Only applies to directories
- **type** (*string*) –
- **url** (*string*) – (read only)
- **video_preview** (*string*) – Relative URL of a video preview

`GET /storagepools/`

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[].capacity** (*integer*) – Capacity of storage pool (required)
- **results[].filesystem** (*string*) – Storage pool filesystem (required)
- **results[].free_space** (*integer*) – Free space of storage pool (required)
- **results[].is_default** (*boolean*) – Check if default pool
- **results[].name** (*string*) – Name storage pool (required)
- **results[].site** (*integer*) – Site correlated with storage object (required)
- **results[].used_space** (*integer*) – Used space of storage pool (required)

`GET /storagepools/{id}/`

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **capacity** (*integer*) – Capacity of storage pool (required)
- **filesystem** (*string*) – Storage pool filesystem (required)
- **free_space** (*integer*) – Free space of storage pool (required)
- **is_default** (*boolean*) – Check if default pool
- **name** (*string*) – Name storage pool (required)
- **site** (*integer*) – Site correlated with storage object (required)
- **used_space** (*integer*) – Used space of storage pool (required)

`GET /tags/`

API endpoint for search tags

Query Parameters:

- **filter** (*string*) – substring that returned tags must contain

Status Codes:

- 200 OK –

Response JSON Object:

- **[].name** (*string*) – (required)

`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
- **task_id** (*string*) – Task Id
- **state** (*array*) – Task states
- **job_id** (*integer*) – Job ID
- **internal** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **count** (*integer*) – (required)
- **next** (*string*) –
- **previous** (*string*) –
- **results[].dynamic_parents[]** (*integer*) –
- **results[].id** (*integer*) – (read only)
- **results[].job** (*integer*) – (required)
- **results[].parents** (*string*) – (read only)
- **results[].queue** (*string*) – (required)
- **results[].site** (*string*) – (required)
- **results[].started** (*string*) – Time that the task started running
- **results[].state** (*string*) –
- **results[].task_id** (*string*) – Celery task ID
- **results[].tasktype** (*string*) – (required)
- **results[].url** (*string*) – (read only)

`GET /tasks/{id}/`

API endpoint for viewing tasks.

Parameters:

- **id** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **completed** (*string*) – Time of the task completion
- **dynamic_parents[]** (*integer*) –
- **friendly_name** (*string*) – Text to be used to describe what this task is doing.
- **id** (*integer*) – (read only)
- **job** (*integer*) – (required)
- **parents** (*string*) – (read only)
- **paths** (*string*) – (read only)
- **queue** (*string*) – (required)
- **result** (*object*) –
- **runtime** (*string*) – (read only)
- **site** (*string*) – (required)

- **started** (*string*) – Time that the task started running
- **state** (*string*) –
- **task_id** (*string*) – Celery task ID
- **tasktype** (*string*) – (required)
- **url** (*string*) – (read only)

`GET /tasks/{id}/files/`

API endpoint for viewing tasks.

Parameters:

- **id** (*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.
- **state** (*string*) –
- **type** (*string*) –
- **site** (*string*) –

Status Codes:

- 200 OK –

Response JSON Object:

- **completed** (*string*) – Time of the task completion
- **dynamic_parents[]** (*integer*) –
- **friendly_name** (*string*) – Text to be used to describe what this task is doing.
- **id** (*integer*) – (read only)
- **job** (*integer*) – (required)
- **parents** (*string*) – (read only)
- **paths** (*string*) – (read only)
- **queue** (*string*) – (required)
- **result** (*object*) –
- **runtime** (*string*) – (read only)
- **site** (*string*) – (required)
- **started** (*string*) – Time that the task started running
- **state** (*string*) –
- **task_id** (*string*) – Celery task ID
- **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[].nas_user.allow_ssh** (*boolean*) –
- **results[].nas_user.primary_group** (*string*) – (required)
- **results[].nas_user.uid** (*integer*) – UID number for the user. This is automatically generated. And cannot be changed (read only)
- **results[].profile._keycloak_refresh** (*string*) –
- **results[].profile.country** (*string*) – User's country
- **results[].profile.default_site** (*integer*) –
- **results[].profile.department** (*string*) – Name of the department user is working in
- **results[].profile.home_site** (*integer*) –
- **results[].profile.line_manager** (*string*) – Name of the line manager of the user
- **results[].profile.phone** (*string*) – User's phone number
- **results[].profile.timezone** (*string*) – Timezone in a string format e.g. 'Europe/London'
- **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*) –
- **nas_user.allow_ssh** (*boolean*) –
- **nas_user.primary_group** (*string*) – (required)
- **password** (*string*) – (required)

- **profile._keycloak_refresh** (*string*) –
- **profile.country** (*string*) – User's country
- **profile.default_site** (*integer*) –
- **profile.department** (*string*) – Name of the department user is working in
- **profile.home_site** (*integer*) –
- **profile.line_manager** (*string*) – Name of the line manager of the user
- **profile.phone** (*string*) – User's phone number
- **profile.timezone** (*string*) – Timezone in a string format e.g. 'Europe/London'
- **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*) –
- **nas_user.allow_ssh** (*boolean*) –
- **nas_user.primary_group** (*string*) – (required)
- **password** (*string*) – (required)
- **profile._keycloak_refresh** (*string*) –
- **profile.country** (*string*) – User's country
- **profile.default_site** (*integer*) –
- **profile.department** (*string*) – Name of the department user is working in
- **profile.home_site** (*integer*) –
- **profile.line_manager** (*string*) – Name of the line manager of the user
- **profile.phone** (*string*) – User's phone number
- **profile.timezone** (*string*) – Timezone in a string format e.g. 'Europe/London'
- **username** (*string*) – Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. (required)

`POST /users/change_password/`

API endpoint for managing users.

Request JSON Object:

- **new_password** (*string*) – (required)
- **old_password** (*string*) – (required)

Status Codes:

- 201 Created –

Response JSON Object:

- **new_password** (*string*) – (required)
- **old_password** (*string*) – (required)

`POST /users/sync/`

API to sync nas users on each site

Status Codes:

- 201 Created –

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*) –
- **nas_user.allow_ssh** (*boolean*) –
- **nas_user.primary_group** (*string*) – (required)
- **nas_user.uid** (*integer*) – UID number for the user. This is automatically generated. And cannot be changed (read only)
- **profile._keycloak_refresh** (*string*) –
- **profile.country** (*string*) – User's country
- **profile.default_site** (*integer*) –
- **profile.department** (*string*) – Name of the department user is working in
- **profile.home_site** (*integer*) –
- **profile.line_manager** (*string*) – Name of the line manager of the user
- **profile.phone** (*string*) – User's phone number
- **profile.timezone** (*string*) – Timezone in a string format e.g. 'Europe/London'
- **url** (*string*) – (read only)
- **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*) –
- **nas_user.allow_ssh** (*boolean*) –
- **nas_user.primary_group** (*string*) – (required)
- **nas_user.uid** (*integer*) – UID number for the user. This is automatically generated. And cannot be changed (read only)
- **profile._keycloak_refresh** (*string*) –

- **profile.country** (*string*) – User's country
- **profile.default_site** (*integer*) –
- **profile.department** (*string*) – Name of the department user is working in
- **profile.home_site** (*integer*) –
- **profile.line_manager** (*string*) – Name of the line manager of the user
- **profile.phone** (*string*) – User's phone number
- **profile.timezone** (*string*) – Timezone in a string format e.g. 'Europe/London'
- **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*) –
- **nas_user.allow_ssh** (*boolean*) –
- **nas_user.primary_group** (*string*) – (required)
- **password** (*string*) –
- **profile._keycloak_refresh** (*string*) –
- **profile.country** (*string*) – User's country
- **profile.default_site** (*integer*) –
- **profile.department** (*string*) – Name of the department user is working in
- **profile.home_site** (*integer*) –
- **profile.line_manager** (*string*) – Name of the line manager of the user
- **profile.phone** (*string*) – User's phone number
- **profile.timezone** (*string*) – Timezone in a string format e.g. 'Europe/London'

Status Codes:
- 200 OK –

Response JSON Object:
- **email** (*string*) –
- **first_name** (*string*) –
- **groups[]** (*string*) –
- **last_name** (*string*) –
- **nas_user.allow_ssh** (*boolean*) –
- **nas_user.primary_group** (*string*) – (required)
- **password** (*string*) –
- **profile._keycloak_refresh** (*string*) –
- **profile.country** (*string*) – User's country
- **profile.default_site** (*integer*) –
- **profile.department** (*string*) – Name of the department user is working in
- **profile.home_site** (*integer*) –
- **profile.line_manager** (*string*) – Name of the line manager of the user
- **profile.phone** (*string*) – User's phone number
- **profile.timezone** (*string*) – Timezone in a string format e.g. 'Europe/London'

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[].discovery_options** (*object*) –
- **results[].enabled** (*boolean*) – Is the workflow available for use?
- **results[].external_only** (*boolean*) – Signifies that the workflow is only allowed to operate on external targets
- **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[].post_action** (*object*) –
- **results[].pre_action** (*object*) –
- **results[].preferred_queue** (*string*) – Preferred queue for this workflow
- **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*) –
- **discovery_options** (*object*) –
- **enabled** (*boolean*) – Is the workflow available for use?
- **external_only** (*boolean*) – Signifies that the workflow is only allowed to operate on external targets
- **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)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **preferred_queue** (*string*) – Preferred queue for this workflow
- **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*) –
- **discovery_options** (*object*) –
- **enabled** (*boolean*) – Is the workflow available for use?
- **external_only** (*boolean*) – Signifies that the workflow is only allowed to operate on external targets
- **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)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **preferred_queue** (*string*) – Preferred queue for this workflow
- **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*) –
- **discovery_options** (*object*) –
- **enabled** (*boolean*) – Is the workflow available for use?
- **external_only** (*boolean*) – Signifies that the workflow is only allowed to operate on external targets
- **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)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **preferred_queue** (*string*) – Preferred queue for this workflow
- **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*) –
- **discovery_options** (*object*) –
- **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)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **preferred_queue** (*string*) – Preferred queue for this workflow
- **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*) –
- **discovery_options** (*object*) –
- **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)
- **post_action** (*object*) –
- **pre_action** (*object*) –
- **preferred_queue** (*string*) – Preferred queue for this workflow
- **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

ngeneahubctl

The `ngeneahubctl` command is the main command line administration tool for Ngenea Hub

database

The `ngeneahubctl database` sub-commands are used for managing the Ngenea Hub database.

The database these commands will affect is the one Ngenea Hub is currently configured to use. These commands will work regardless of whether the target database is 'internal' or external, and the behaviour will be equivalent.

database wipe

Delete all data from the Ngenea Hub database.

```
ngeneahubctl database wipe
```

Warning: This operation is not reversable.

You should **always create a database backup** before running 'wipe'. See the Backup guide for more details on creating backups.

Ngenea Hub must be running to perform a database wipe. After executing the ‘wipe’ command, perform a full restart of Ngenea Hub to re-initialise the now empty database.

After performing ‘wipe’, it will be necessary to re-join all Ngenea workers.

To execute this command, you will be prompted to verify that the command is being run on the correct machine by typing in the hostname of the machine. You will also be prompted to confirm that the correct database is being wiped - either ‘internal’ or ‘external’.

If the target database is external, and that database contains data other than that used by Ngenea Hub, only the Ngenea Hub data will be deleted. Any other data in the database will not be affected.

This command differs from `ngeneahubctl wipe`, which drops all Ngenea Hub container volumes. Dropping container volumes will clear database data as a side-effect, and only if Ngenea Hub is configured to use an internal database. `ngeneahubctl wipe` cannot remove data from an external database.

wipe

Drop all Ngenea Hub internal container volumes.

```
ngeneahubctl wipe
```

Warning: This operation is not reversable.

Dropping volumes will clear all Ngenea Hub internal container data, including

- database data (if internal)
- persistent Redis data (if internal)
- persistent RabbitMQ data (if used and internal)
- SSL certificates for internal services

`ngeneahubctl wipe` cannot remove data or volumes for external services.

To delete only database data, or to delete data from an external database, use `ngeneahubctl database wipe` instead.

You should **always create a database backup** before running ‘wipe’. See the Backup guide for more details on creating backups.

After performing ‘wipe’, it will be necessary to re-join all Ngenea workers.

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
-D, --no-verify                    Disable TLS verification

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

-q, --queue QUEUE                  Queue to perform the workflow
on
-Q, --target-queue TARGETQUEUE     Queue 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.

- **-D**, **–no-verify**

  Disables TLS Verification

  By default, **ngclient** will verify TLS certificate at the remote end.

  With this flag, requests made by **ngclient** will accept any TLS certificate presented by the server, and will ignore hostname mismatches and/or expired certificates.

- **–client-key**

  Ngenea Hub authentication client key.

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

- **-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

- **-q**, **–queue**

  Queue to use for workflows. By default, source queue is set as *default*

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

- **-Q**, **–target-queue**

  Target queue for the *send* workflow

  If the target queue is not provided, it will use the source queue. If the source queue is also not provided, the queue will be set to *default*.

- **-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
```

NOTE: The `authenticate` command doesn't read config values from the config file, so the *–no-verify* and *–base-url* arguments should be passed as cmd arguments if default values aren't required

RECALL A FILE

```
ngclient recall /mmfs1/data/hello.txt -s site1 --client-key jDBh2
cRk6.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

2021 ArcaPix Limited

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the 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 defi
nition

    --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)** *aut henticate*

- **-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":
["hurricane-01"], "discovery": "recursive", "enabled": true, "vis
ible": true, "fields": [], "filter_rules": [{"type": "all", "state
": "all", "action": [{"name": "dynamo.tasks.migrate"}], "descript
ion": "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", "ico
n_classes": [], "filter_rules": [{"type": "all", "state": "all", "
action": [{"name": "dynamo.tasks.reverse_stub", "site": "*destina
tionsite", "overwrite": true}]}], "fields": [{"name": "destinatio
```

```
nsite", "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

2021 ArcaPix Limited

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the 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 Ngenea Hub.

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)** *auth enticate* 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 disab
le

    --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)** *aut henticate*

- **-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

2021 ArcaPix Limited

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

ngclient-settings

NGCLIENT-SETTINGS

SYNOPSIS

ngclient settings refresh [options...]

DESCRIPTION

The *settings refresh* command is used to trigger a refresh of site settings in Ngenea Hub.

This means loading the latest settings from PixStor on one or all sites into Ngenea Hub. This may be useful if a setting has been manually changed on a site, or if the settings displayed in Ngenea Hub are otherwise wrong or out-of-date.

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)** *auth enticate* command can be used to generate a client key from a username or access token.

OPTION SUMMARY

```
    --site                  Name of a specific site to refresh set
tings from

    --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

- **–site**

  Name of a specific site to refresh settings on.

  If this flag isn't specified, settings will be refreshed for all sites registered in Ngenea Hub

- **–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)** *aut henticate*

- **-h**, **–help**

  Prints the help message.

EXAMPLES

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

REFRESH SETTINGS ON ONE SITE

```
$ ngclient settings refresh --site london
```

REFRESH SETTINGS ON ALL SITES

```
ngclient settings refresh
```

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

2023 ArcaPix Limited

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

ngclient-sites

NGCLIENT-SITES

SYNOPSIS

ngclient sites refresh [options...]

DESCRIPTION

The *sites refresh* command is used to trigger a refresh of site-specific metadata in Ngenea Hub.

Site metadata includes things like storage pools, filesystems, analytics, etc. This metadata is typically refreshed periodically, but if the data currently returned from Ngenea Hub seems wrong or out of date, this command can be used to force a refresh.

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)** *auth enticate* command can be used to generate a client key from a username or access token.

OPTION SUMMARY

```
    --site                  Name of a specific site to refresh set
tings from

    --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

- **–site**

  Name of a specific site to refresh metadata on.

  If this flag isn’t specified, metadata will be refreshed for all sites registered in Ngenea Hub

- **–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)** *aut henticate*

- **-h**, **–help**

  Prints the help message.

EXAMPLES

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

REFRESH METADATA ON ONE SITE

```
$ ngclient sites refresh --site london
```

REFRESH METADATA ON ALL SITES

```
ngclient sites refresh
```

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

2023 ArcaPix Limited

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the 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
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)** *aut henticate*

- **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.

- **queue**

  The default queue to use for workflows.By default, source queue is set as *defa ult*

  For *migrate* and *recall*, this is the queue where the workflows execute. For *send*, this is the source queue, 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.

- **api_secure_verify**

  Whether to enable/disable TLS Verification when communicating with Ngenea Hub REST API.

  This is particularly useful when Ngenea Hub REST API is behind a self-signed certificate.

  If false, TLS verification will be disabled.

  If not specified, the default is *true*.

EXAMPLE

```
[settings]
base_url = http://mypixserver:8000
client_key = ...
site = mysite
queue = standard
wait = true
timeout = 180
hydrate = true
api_secure_verify = true
```

SEE ALSO

ngclient(1), transparent_recall(1)

LICENSE

2021 ArcaPix Limited

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the 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.

- **–queue**

  Queue to perform the transparent recall on.

- **-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.

The *queue* can be specified in the configuration; otherwise, it will be set to the default option, *default*.

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.

```
/* BEGIN: NGENEA TRANSPARENT RECALL 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 LENGTH(XATTR('dmapi.APXguuid'))>0 AND
```

```
CountSubstr(MISC_ATTRIBUTES,'V')>0

RULE FileOpen_else EVENT 'OPEN' DIRECTORIES_PLUS

RULE FileData EVENT 'DATA'
    ACTION( system_lw('/usr/bin/python3 /usr/bin/
transparent_recall :' ||
        VARCHAR(FS_ID) || ':' ||VARCHAR(INODE) || ':' ||
VARCHAR(GENERATION) ||
        ' ' || getDetail('path_name') )=0)
    WHERE getDetail('open_flags')!='0' AND
LENGTH(XATTR('dmapi.APXguuid'))>0 AND
CountSubstr(MISC_ATTRIBUTES,'V')>0

RULE FileData_else EVENT 'DATA' DIRECTORIES_PLUS

/* END: NGENEA TRANSPARENT RECALL POLICY */
```

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

Adjusting the transparent recall queue

If an alternative queue has been established, for example `transparent_recall` it can be used for all transparent recall operations by using this policy alternative:

```
RULE FileData EVENT 'DATA'
    ACTION( system_lw('/usr/bin/python3 /usr/bin/
transparent_recall --queue transparent_recall :' ||
        VARCHAR(FS_ID) || ':' ||VARCHAR(INODE) || ':' ||
VARCHAR(GENERATION) ||
        ' ' || getDetail('path_name') )=0)
    WHERE getDetail('open_flags')!='0' AND
LENGTH(XATTR('dmapi.APXguuid'))>0 AND
CountSubstr(MISC_ATTRIBUTES,'V')>0
```

This substitutes the main `ACTION` to allow for all transparent recall operations to go through the custom queue exclusively. This means transparent recalls will not be effected by any other operations happening on the main

For more details on how to create and enable these custom queues, refer to the configuration section of the ngeneahub documentation.

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

2021 ArcaPix Limited

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Backup

This section provides detailed instructions for backing up and restoring static configurations and selected database tables in Ngenea Hub. It also describes how to prune older backup archives based on a configured retention policy. It is intended for administrators responsible for maintaining the stability and recoverability of the Hub environment.

Ngenea Hub's backup and restore functionality allows administrators to preserve essential configuration data, such as users, schedules, workflows, external targets, spaces, and system settings.

By default, backups include configuration-related database tables and static configuration files.

When the `--full` flag is used, Hub Job history and task-related data is included in the backup.

The guide outlines the commands and procedures required to:

- Export configuration data and database records into a secure archive
- Optionally include Job history data using `--full`
- Restore the system to a previously saved state
- Prune older backups based on the configured retention policy
- Configure automatic and scheduled backups

By following this document, administrators can ensure minimal downtime, maintain system integrity across updates or unexpected disruptions, and manage backup storage effectively.

Backing Up HUB Static Configurations

Warning: By default, the backup process includes only static configurations and selected database tables. Job and task-related data are **not included** unless a **Full** backup is explicitly requested.

Ngenea Hub provides a set of commands to efficiently back up and restore critical static configurations and selected database tables. It also provides a command to prune older backup archives according to a defined retention policy. This ensures quick recovery and minimal downtime during system upgrades or failures at a cost of reduced time to retain records.

These commands allow administrators to safeguard essential configuration data such as **schedules**, **users**, **workflows**, **external targets**, and **spaces**.

Two backup modes are supported:

- **ConfigOnly** (default) – Backs up static configurations and selected configuration. Job and task history are excluded.
- **Full** – Includes everything in ConfigOnly, plus job and task history in an additional archive.

Warning: Full backups include job history data, which can significantly increase the size of the backup files.

In environments with large job datasets, this may result in longer backup and restore times and increased storage usage.

```
ngeneahubctl backup --help

Usage: ngeneahubctl backup [OPTIONS] COMMAND [ARGS]...

  Backup and restore ngeneahub static configurations

Options:
  -T, --disable-tty       Disable pseudo-tty allocation
  --container-mount TEXT  Container mount path for DB
  --db-container TEXT     Name of the DB container
```

```
  --external-db           Use external database
  --help                  Show this message and exit.

Commands:
  auto     Export and Prune ngeneahub static configurations
  export   Export ngeneahub static configurations to a file
  prune    Prune ngeneahub static configurations
  restore  Restore ngeneahub static configurations from a file
```

Export

The backup process captures key Ngenea Hub configurations and selected database tables, providing a snapshot of key static configurations.

By default, the backup includes static configuration data only. If a Full backup is requested, job and task history data are also included.

To perform a backup, the Ngenea Hub database must be accessible. If using the `internal` database, ensure the database container is running. The container name is `ngeneahub-db-1`.

Service status can be validated with: `ngeneahubctl status`.

If using an external database, ensure the external postgres database service is running and reachable before performing the backup.

Command

```
ngeneahubctl backup export --help

Usage: ngeneahubctl backup export [OPTIONS]

  Export ngeneahub static configurations to a file

Options:
  -o, --output DIRECTORY  Location to store generated archive.
Default
                          '$(PWD)'
  -f, --full              Create an additional file containing
only the job history.
  --help                  Show this message and exit.
```

This command performs the following actions:

- **Database Dump** (ConfigOnly – Default)

  Exports selected configuration-related tables into a SQL file, including:

  - Schedules
  - Users
  - Workflows
  - External targets

    - Global configurations
    - Spaces

  This mode excludes job history.

- **Full Backup**

  When the `--full` flag is provided

  ```
  ngeneahubctl backup export --full <backup_path>
  ```

  In addition to the archive created using the ConfigOnly parameter, an additional archive is created containing job and task history data which includes:

    - Workflow Jobs and tasks
    - Settings Jobs
    - Job results and statistics

Configuration backup

Copies static configuration files from key directories **into the ConfigOnly archive**:

- `/etc/ngenea/postgres` - PostgreSQL settings
- `/etc/ngenea/redis` - Redis configurations
- `/etc/ngenea/rabbitmq` - RabbitMQ configurations
- `/etc/ngenea/ngenea-client.conf` - Hub client configuration
- `/etc/sysconfig/ngeneahub` - Hub configuration

Archive creation

Packages the SQL dump and configuration files into a compressed archive. - `ngeneahub_backup_<timestamp>.tar.gz` - Static configurations and configuration database tables - `ngeneahub_backup_jobhistory_<timestamp>.tar.gz` - Job history tables

These archives are saved to the current working directory or to a directory specified by the `-o` or `--output` command-line option.

> **Note:** Backups can also be created automatically:
>
> - On Hub startup (see Backup on Startup)
> - On a scheduled basis (see Auto Backup)

Backup on Startup

When the Ngenea Hub starts up, a database backup is automatically created using the backup tool.

Backup type: Controlled via the ngeneahubctl sysconfig file `/etc/sysconfig/ngeneahub` using:

`BACKUP_ON_START = Full | ConfigOnly | None`

Full – Full database backup (including Job History)

ConfigOnly – Backup only configuration tables (default)

None – Skip backup on startup

> **Note:** If `BACKUP_ON_START=Full`, the backup process generates two separate archives
>
> 1. `ngeneahub_backup_<timestamp>.tar.gz` – Standard database and configuration backup
> 2. `ngeneahub_backup_jobhistory_<timestamp>.tar.gz` – Job history backup

Backup location: Controlled via `postgres_spool_dir` in the sysconfig file. This defines where the backup files will be stored.

> **Note:** The startup backup occurs before the backend applies database updates to ensure a consistent database snapshot.

Restore

> **Note:** Depending on the scenario, a wipe **may be** required before restoring a backup.

The restore process reinstates the backed-up database tables and configuration files, returning the Hub to its operational state following a failure or upgrade.

To perform a restoration, Ngenea Hub must be fully operational.

> **Note:** After performing a restoration, all services need to be restarted.

Command

```
ngeneahubctl backup restore --help

Usage: ngeneahubctl backup restore [OPTIONS] PATH

  Restore ngeneahub static configurations from a file

Options:
  -f, --full  Add an additional flag which restores job history
  --help  Show this message and exit.
```

Restore Modes

- **ConfigOnly Restore**

Restores selected configuration-related database tables and static configuration files including:

- Schedules
- Users
- Workflows
- External targets
- Global configurations
- Spaces

This mode excludes job history.

- **Full Restore**

  When the `--full` flag is provided

  ```
  ngeneahubctl backup restore --full <backup_path>
  ```

  This restores Job history related data which includes:

  - Workflow Jobs and tasks
  - Settings Jobs
  - Job results and statistics

This command performs the following:

- **Unpack the backup archive**: Extracts the contents of the backup file.
- **Restore configurations**: Copies the saved configuration files back into their respective directories, re-establishing the static system setup.
- **Truncate and reset tables**: Clears the selected database tables, resetting them to ensure no conflicts with the incoming data.
- **Import the SQL dump**: Inserts the backed-up data into the database, restoring Configuration tables (default) and Job history tables (when `--full` is used)

Prune

The prune command is used to remove old database backups based on the retention policy.

Retention setting: `db_backup_retain_count`

If not set (None), all backups are kept and pruning does not run.

Pruning process

1. Removes backups exceeding the retention count
2. Deletes the oldest backups first
3. Only affects files in the configured `db_backup_dir`
4. Logs: Output is sent to the systemd logs

Command

```
ngeneahubctl backup prune --help

Usage: ngeneahubctl backup prune [OPTIONS]

  Prune ngeneahub static configurations

Options:
  --help    Show this message and exit.
```

Note: The prune operation can be executed manually or automatically as part of the auto backup process. See Auto Backup for details.

Auto Backup

Ngenea Hub provides a management command for automated backups

Command

```
ngeneahubctl backup auto --help

Usage: ngeneahubctl backup auto [OPTIONS]

  Export and Prune ngeneahub static configurations

Options:
  --help    Show this message and exit.
```

This command performs a 3-step process:

Export - Calls `ngeneahubctl backup export` to create a database backup

Prune - Calls `ngeneahubctl backup prune` to remove old backups according to retention settings

Report - Sends a report of the entire operation to the configured report target (if enabled)

Each auto backup creates a job in the Hub UI for the corresponding backup run. The job includes tasks for both the backup and prune steps, with detailed task results showing created and pruned backups.

Note: These Jobs cannot be re-submitted.

Configuration Settings

These are the supported backup configurations, which can be set via the /configurations API in the backend or through the Global Settings page in the UI.

Setting	Possible Values	Description
`db_backup_enabled`	True / False	Enable or disable database backups
`db_backup_type`	Full / ConfigOnly	Defines the backup type (default: ConfigOnly)
`db_backup_dir`	File path	Directory where backups will be stored
`db_backup_schedule`	Cron expression	Schedule for periodic backups (e.g., `22 2 * * *`)
`db_backup_retain_count`	≥1	Maximum number of backups to retain
`db_backup_report`	True / False	Whether to generate and send a backup report

Scheduled Backups

Auto backups can also be scheduled using systemd service and timer. Database backups run as an external systemd service included with the ngenea-hub RPM.

Service config: /usr/lib/systemd/system/ngenea-hub-database-backup.service

Timer config: /usr/lib/systemd/system/ngenea-hub-database-backup.timer

The timer schedule is updated from the `db_backup_schedule` configuration which can be set via configurations API or through the UI. At the scheduled time, the service automatically executes `ngeneahubctl backup auto` command.

Alerts

Ngenea Hub allows you to configure job alerts to monitor the status of both `scheduled` and `non-scheduled` jobs. Alerts can be triggered when jobs have not started, are running but not progressing, failing, failed with task failures, completely failed or completed successfully.

Alerts are useful for ensuring that issues are detected and communicated promptly via email to the configured recipients. You can configure alerts either via the `Global Settings page` in the UI or via the `/configurations` API in the backend.

For more details on how to configure alerts in the UI, see Job Alerts.

Configuration Settings

The following backend configuration fields are supported for job alerts. These fields can be updated via the `/configurations` API.

Setting	Type	Default	Description
`alert_scheduled`	Boolean	`False`	Enable alerts for **scheduled jobs**.
`alert_unscheduled`	Boolean	`False`	Enable alerts for **unscheduled jobs**.

Setting	Type	Default	Description
`alert_notstarted_jobs`	Boolean	`False`	Enable alerts for jobs that **have not started**.
`notstarted_job_threshold`	Integer (seconds)	`7200`	Time in seconds before a job is considered not started.
`alert_notprogressed_jobs`	Boolean	`False`	Enable alerts for jobs that are **running but not progressed**.
`notprogressed_job_threshold`	Integer (seconds)	`7200`	Time in seconds before a job is considered not progressed.
`alert_inprogress_failing_jobs`	Boolean	`False`	Enable alerts for jobs that are **failing while in progress**.
`alert_completed_has_failures_jobs`	Boolean	`False`	Enable alerts for jobs that **completed with task failures**.
`alert_completed_failed_jobs`	Boolean	`False`	Enable alerts for jobs that **completed with job failure**.
`alert_completed_success_jobs`	Boolean	`False`	Enable alerts for jobs that **completed successfully**.
`alert_email_address`	Array of EmailField	`[]`	List of email addresses that will receive alert notifications.
`alerting_started_at`	Datetime	`null`	Timestamp indicating when alerting was activated.

Programming Guide

This guide is intended for users who work with, build, or integrate with the Hub APIs. It provides a clear overview of how the API is structured, how to interact with it effectively, and the standards followed in its design and implementation. You'll find information on authentication, endpoint conventions, request/response formats, error handling, versioning, and best practices.

REST API

To better understand APIs in the context of **Ngenea Hub**, let's start with a quick overview of what an API is and its essential concepts.

An **API (Application Programming Interface)** is a set of rules and protocols that allows software applications to communicate with each other. APIs define how to request and send data, which functions are available, and what responses can be expected.

What are Remote APIs?

Remote APIs are APIs where the software making the request and the software providing the data are on different machines or different networks. These APIs work over the internet or a network, using tools like `HTTP`.

What are Web-Based APIs?

Web-based APIs are **remote APIs** that use web protocols like `HTTP` or `HTTPS` to communicate. They often send and receive data in formats like `JSON`. These APIs are widely used in websites and mobile apps to get or send data.

What are REST APIs?

`REST` (Representational State Transfer) APIs are a special kind of web-based API that follow a specific set of design rules. They use URLs to identify resources (like a user or a file) and `HTTP` methods like `GET` or `POST` to perform actions on those resources.

When is an API called RESTful?

An API is called `RESTful` when it follows the rules (constraints) of `REST`. It must behave in a certain predictable way, such as using `HTTP` properly and not storing client information on the server between requests.

What Are the Constraints that make an API REST?

- **Stateless**: Each request from a client to the server must contain all the information needed. The server doesn't **remember** past requests.
- **Client-Server Architecture**: The system is split into clients (apps, browsers) and servers (data providers), and each focuses only on its role.
- **Cacheable**: Responses can be stored by the client to avoid repeating the same request.
- **Uniform Interface**: There is a standard way to access resources (like using `GET` to read data, `POST` to create).
- **Layered System**: The system can have layers (like security, load balancers) that the client doesn't need to know about.
- **Code on Demand** (optional): Servers can send code (like `JavaScript`) to the client to run if needed.

How is JSON Related to API?

`JSON` (JavaScript Object Notation) is a simple format for organizing and exchanging data. When an app talks to an API, the API often replies with data in `JSON` format.

What is an OAuth Token?

An `OAuth` token is like a digital key that lets someone use a service without logging in every time.

HTTP Verbs vs CRUD Operations

HTTP Verb	CRUD Operation	Example Action
GET	Read	Get user details
POST	Create	Add a new blog post
PATCH	Update (partial)	Change only the email address
DELETE	Delete	Remove a user

Now, we'll focus on understanding the APIs specifically for Ngenea Hub.

Authentication Methods

Overview

This document outlines the two supported authentication methods for accessing the Ngenea Hub API: `JWT` (JSON Web Tokens) and `Client Keys` (API Keys).

It provides clear instructions on how to obtain, use, and manage these tokens and keys, including examples for both **command-line usage** and **Swagger UI**.

API Reference Location

All endpoints, parameters, and schemas can be explored through the interactive API documentation available at:

```
http(s)://<your-hub-address>/api/docs/
```

> **Note:** Replace `your-hub-address` with the actual IP address or domain of your Ngenea Hub instance.

JWT Authentication - Using Curl

A `JWT` (JSON Web Token) is a compact, secure way to represent user identity. It is commonly used for authentication. After logging in with your credentials, the server provides a token that confirms your identity, allowing you to make future API requests without resending your username and password each time.

To obtain a JWT, send a `POST` request to the authentication endpoint using curl:.

Example Format

```
curl -X 'POST' \
  'http://<your-hub-address>/api/auth/token/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "<username>",
  "password": "<password>"
}' |jq
```

Note: The `curl` command works across different operating systems, but with slight differences in usage.

On `Linux` and `macOS`, curl is typically pre-installed and can be run directly in the terminal without any setup.

On `Windows 10` and later, curl is also built into Command Prompt (CMD) and PowerShell, you can use GIT BASH as well.

If you're using an older version of Windows, you may need to download curl from the official site: curl for Windows .

Note: `jq` is a command-line tool used to easily parse and extract specific data from `JSON` responses, making it simpler to handle `API` outputs like `JWT` tokens.

It comes pre-installed on many Linux distributions and macOS, otherwise, it must be installed manually as per Download jq , while on Windows you can install it via package managers like Chocolatey, Winget, or use it in WSL - check your installed version anytime with `jq -version`.

The server responds with two tokens:

- An **access token**, which is short-lived (1 hour) and should included in each request to authenticate the user.
- A **refresh token**, which lasts longer (1 day) and is used to obtain a new access token when the current one expires.

These tokens are returned in a JSON format like this:

```
{
  "access": "your_access_token_here",
  "refresh": "your_refresh_token_here"
}
```

Using JWT Access Token for API Requests

When making authenticated requests using the JWT token, include it in the `Authorization` header using the `Bearer` scheme.

```
Bearer <JWT_Access_Token>
```

Example: Hub Health

```
curl -X 'GET' \
  'http://<your-hub-address>/api/health/' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer <JWT_Access_Token> '|jq
```

Example Response:


```
{
  "overall_health": "ok",
  "hub_status": {
    "health": "ok"
  },
  "site_status": [
    {
      "site": "siteA",
      "health": "ok",
      "nodes": [
        {
          "name": "siteA",
          "health": "ok",
          "online": true,
          "has_default_queue": true,
          "last_heartbeat": "2025-06-02T15:25:18.562813+00:00",
          "worker_version": "2.6.0",
          "ngenea_version": "1.31.0",
          "analytics_version": "2.7.1",
          "search_version": "1.2.0"
        }
      ],
      "queues": [
        {
          "name": "default",
          "health": "ok",
          "online": true,
          "last_heartbeat": "2025-06-02T15:25:18.562813+00:00"
        }
      ]
    },
}
```


Using Refresh Tokens to Obtain New Access Tokens

As mentioned earlier, **access tokens expire after 1 hour**, while refresh **tokens expire after 1 day**. When you use an expired access token, you'll receive an `HTTP 401 Unauthorized` or `HTTP 403 Forbidden error`.

In this situation, you should request a new access token using the `/api/auth/token/refresh/` endpoint with your refresh token.

Example request using curl:

```
$ 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
```

Response :

```
{
  "access": <new_access_token>,
}
```

Client Key Authentication

A client key (or API key) is a long-lived, unique token used to authenticate and authorize applications or scripts to access an API. Unlike short-lived tokens like JWTs, client keys typically do not expire unless manually revoked, making them ideal for automated processes, integrations, and services that require persistent access without frequent re-authentication.

To generate a client key, send a `POST` request to the `/api/auth/clientkeys/` endpoint with a descriptive name for the key, including your **JWT access token** in the Authorization header:

You can either use the access token obtained above or generate a new one using the following command:

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

echo $JWT_TOKEN
```

Note: Why is this different ?

The difference lies in improving user flow and convenience:

- `export JWT_TOKEN=...` stores the access token in an environment variable called `JWT_TOKEN`. This makes the token easily accessible within the current shell session and any child processes (such as scripts or subsequent commands).
- The use of `jq -r .access` extracts only the raw access token (without quotes) from the JSON response.
- `echo $JWT_TOKEN` prints the token to the terminal, confirming that it was successfully retrieved and saved.

This approach simplifies usage by allowing the token to be reused easily in subsequent commands. (These commands are for Linux based systems)

Example Format - Client Key Creation

```
curl -s -X POST 'http://10.201.2.224:8000/api/auth/clientkeys/'
-H 'Accept: application/json'
-H 'Content-Type: application/json'
-H "Authorization: Bearer $JWT_TOKEN"
-d '{"name": "your_key"}' | jq '.'
```

Example Response

```
{
  "url": "http://localhost:8000/api/auth/clientkeys/10/",
  "id": 10,
  "name": "your_key",
  "api_key": "YOUR_API_KEY"
}
```

Using Client Key for API Requests

Once you have the client key, you can use it for API access by including it in the `Authorization` header using the `Api-Key` scheme.

```
Api-Key <api_key>
```

Example-Jobs

```
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
```

Note: Replace `YOUR_API_KEY` with the actual API key string returned during creation.

Example Response

```
{
  "count": 6,
  "next": null,
  "previous": null,
  "results": [
    {
      "url": "http://my-hub:8000/api/jobs/1/",
      "id": 1,
      "workflow": null,
      "fields": null,
      "created": "2025-05-14T13:00:40.153027Z",
      "started": "2025-05-14T13:00:48.166288Z",
      "completed": "2025-05-14T13:00:49.829041Z",
      "runtime": 1.662753,
      "state": "SUCCESS",
```

```
      "site": "siteA",
      "queue": null,
      "owner": {
        "type": "user",
        "id": 1,
        "name": "pixit"
      },
      "discovery": null,
      "dir_walk_complete": true,
      "is_settings_job": true,
      "friendly_name": "Populating/Updating settings from site
siteA",
      "progress": null
    },
    {
      "url": "http://my-hub:8000/api/jobs/2/",
      "id": 2,
      "workflow": null,
      "fields": null,
      "created": "2025-05-14T13:02:14.395013Z",
      "started": "2025-05-14T13:02:18.294664Z",
      "completed": "2025-05-14T13:02:19.611492Z",
      "runtime": 1.316828,
      "state": "SUCCESS",
      "site": "siteB",
      "queue": null,
      "owner": {
        "type": "user",
        "id": 1,
        "name": "pixit"
      },
      ],
  "owners": [
    {
      "type": "user",
      "name": "hubadmin",
      "id": 2
    },
    {
      "type": "user",
      "name": "pixit",
      "id": 1
    },
    {
      "type": "system",
      "name": "System",
      "id": -2
    },
    {
      "type": "unknown",
      "name": "Unknown",
      "id": -1
    }
```

```
  ]
}
```

Revoking a Client Key

Client keys can be revoked at any time. To do this, you must send a `DELETE` request to the URL of the specific client key, using a valid `JWT` for authentication.

Example Format (Revoking API Key)

```
curl -X DELETE
'http(s)://<your-hub-address>/api/auth/clientkeys/<key-id>/' \
-H "Authorization: Bearer $JWT_TOKEN"
```

> **Note:** Replace `key-id` with the numerical ID of the client key you wish to delete.

Once deleted, the API key will no longer grant access to any API resources.

Key Differences Between JWT and API Keys

- `JWT tokens` consist of an access token that expires after 1 hour and a refresh token that allows you to obtain a new access token without re-entering user credentials. The refresh token itself expires after 1 day.
- `API keys` do not expire automatically and do not require refreshing. They remain valid until manually revoked or deleted.

JWT Authentication - Using Swagger UI

Swagger UI provides a visual interface to test API endpoints easily, including authentication flows.

1. **Open Swagger UI**: Go to your Swagger API documentation, typically available at:

```
http(s)://<your-ngenea-hub-host>/api/docs/
```

This opens the interactive Swagger UI interface for exploring and testing the API.

1. **Authenticate via /auth/token/ Endpoint**: This is the endpoint to obtain your access and refresh JWT tokens.

Look for: `POST`**/api/auth/token/**

Execute the request:

- Click the `POST`**/auth/token/** to expand it.
- Click **“Try it out”**.
- In the `Request body`, input your credentials in this format:

```
{
  "username": "your_username",
```

```
  "password": "your_password"
}
```

- Click **“Execute”**.

Response: If successful, you’ll receive a `JSON` response like:

```
{
  "access": "your_access_token",
  "refresh": "your_refresh_token"
}
```

Authorize the Access Token

To use the access token for other endpoints: Click the “Authorize” button (top right in Swagger UI).

- In the popup, enter your token in this format:

```
Bearer your_access_token
```

- Click **“Authorize”**, then **“Close”**.

Now, all subsequent API requests in Swagger UI will include this token.

Refreshing Expired Access Tokens

Access tokens expire after 1 hour. Use the **refresh token** to get a new access token.

Find the endpoint: `POST`**/api/auth/token/refresh/**

Execute the request:

- Expand the endpoint.
- Click **“Try it out”**.

Input:

```
{
  "refresh": "your_refresh_token"
}
```

- Click **“Execute”**.

Response: You’ll receive a new access token:

```
{
  "access": "new_access_token"
}
```

Use the **Authorize** button again to replace the old access token with this new one.

Generate a Client Key - Using Swagger UI

> **Note:** Make sure you've already completed JWT token generation and authorization via the `/auth/token/` endpoint and used the **Authorize** button in **S wagger UI** to authenticate.
>
> This is required before accessing protected endpoints like `/auth/clientkeys/`.

Locate the `/auth/clientkeys/` Endpoint.

- In Swagger UI, search or scroll down to find: `POST` **/api/auth/clientkeys/**
- Click it to expand its details.
- Click **"Try it out"** button to make the request body editable.
- **Fill in the Request Body**: Enter the necessary parameters.
- A basic example:

```
{
  "name": "my-client-key"
}
```

- Once your request is ready, click **Execute** to send it.
- If successful, you'll receive a response similar to:

```
{
  "url": "https://<your-ngenea-hub-host>/api/auth/clientkeys/0/",
  "id": 0,
  "name": "my-client-key",
  "api_key": "generated_client_api_key"
}
```

> **Note:** Store the `api_key` securely, as it is not retrievable again through the API.

Authorize the API Key

- To use the API key for other endpoints: Click the **"Authorize"** button (top right in Swagger UI).
- In the popup, enter your `API key` in the required format.

Example format:

```
Api-Key <your_api_key>
```

- Click **"Authorize"**, then **"Close"**.
- Now, all subsequent API requests in Swagger UI will include this API key automatically.

API Endpoints - User and Access Management

Introduction

This section provides detailed documentation for all available Ngenea Hub `API` endpoints. Each endpoint is a specific `URL` path designed to perform a distinct function, such as retrieving analytics, managing client keys, or interacting with job data. Whether you're listing resources, updating configurations, or triggering actions, each endpoint follows standard `HTTP` conventions and returns structured JSON responses.

The purpose of this guide is to help you understand exactly how each API endpoint works, including:

- What the endpoint does and when to use it
- Required and optional parameters (path, query, or body)
- Example `curl` requests for real-world usage
- Sample responses with explanations of each field

Each endpoint is broken down into clearly labeled sections to help both beginners and experienced developers quickly understand how to use it effectively.

> **Note:** While the Ngenea API is intended for internal tools and authorized automation within secured environments.
>
> **All API calls require proper authentication using either JWT access tokens or Client API Keys**.
>
> Refer to the link to know more about the authentication methods.

API Reference Location

All endpoints, parameters, and schemas can be explored through the interactive API documentation available at:

```
http(s)://<your-hub-address>/api/docs/
```

> **Note:** Replace `your-hub-address` with the actual IP address or domain of your Ngenea Hub instance.

Auth - Authentication

GET /auth/clientkeys/

Description

Retrieves a paginated list of client API keys. This endpoint is typically used by administrators or authorized users to view all API keys that have been generated for client-side access to the hub. It is useful in scenarios where API access needs to be managed, audited, or reviewed for security purposes.

For example, developers or administrators might use this endpoint to list keys tied to specific users, monitor which keys are active, or integrate this data into dashboards or automated scripts. Each key entry includes its name, the associated user, and a URL that can be used to perform further operations such as updating or deleting the key.

This endpoint does not expose actual client keys, and there is no way to retrieve a client key once it has been created.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X GET \
  'http://<your-server>/api/auth/clientkeys/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "count": 12,
  "next": null,
  "previous": null,
  "results": [
    {
      "url": "http://<your-server>/api/auth/clientkeys/5/",
      "id": 5,
      "name": "client_key_1",
      "user": "hubadmin"
    },
    {
      "url": "http://<your-server>/api/auth/clientkeys/11/",
```

```
      "id": 11,
      "name": "key_2",
      "user": "hubadmin"
    },
    ...
  ]
}
```

Response Fields

Field	Type	Description
`count`	integer	Total number of client keys available.
`next`	string/null	URL to the next page of results (if any).
`previous`	string/null	URL to the previous page of results (if any).
`results`	array	List of client key objects. Each object includes:
`results[].url`	string	API URL to access this specific client key.
`results[].id`	integer	Unique identifier of the client key.
`results[].name`	string	User-defined name for the client key.
`results[].user`	string	Username associated with this client key.

GET /auth/clientkeys/{id}/

Description

Retrieves the details of a specific client API key by its `unique ID`. This endpoint is useful for viewing the metadata of an individual client key such as its name, associated user, and API URL for further operations.

Request Parameter

Name	Type	Required	Description
`id`	string	Yes	The unique identifier (ID) of the client key to retrieve.

Example Request

```
curl -X GET \
  'http://<your-server>/api/auth/clientkeys/7/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "url": "http://<your-server>/api/auth/clientkeys/7/",
  "id": 7,
```

```
  "name": "new_key_0206",
  "user": "hubadmin"
}
```

Response Fields

Field	Type	Description
`url`	string	Full API endpoint URL for this client key.
`id`	integer	Unique ID of the client key.
`name`	string	User-defined name for the client key.
`user`	string	Username associated with the key (e.g., the creator or owner).

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

Description

This endpoint permanently deletes a specific client API key by its unique ID. Use this operation to remove keys that are no longer needed or have been compromised.

Deleted keys cannot be recovered, so use with caution.

Request Parameter

Name	Type	Required	Description
`id`	string	Yes	The unique identifier (ID) of the client key to retrieve.

Example Request

```
curl -X DELETE \
  'http://<your-server>/api/auth/clientkeys/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `204 No Content`

This means the key was successfully deleted.

There is no response body when deletion is successful.

`GET /api/auth/token/publickey/`

Description

This API endpoint gives you the **public key** that is used to verify the **signature** on a `JWT` (JSON Web Token).

When a user logs in, the server creates a `JWT` and signs it using a **private key**. This signed token is then sent to the client (like a web app or mobile app).

Other systems or services (like your backend or microservices) can then use this **public key** to check:

- That the token really came from the authentication server, and
- That it has not been changed or tampered with.

This process is called **token verification**.

Using the public key like this is especially helpful in systems where different parts (like microservices) need to validate tokens **independently**, without needing to contact the main authentication server every time.

Request Parameter

This endpoint does not require any query or path parameters.

Example Request

```
curl -X GET \
  'http://<your-server>/api/auth/token/publickey/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "keys": [
    {
      "alg": "RS256",
      "e": "AQAB",
      "kid": "<key-id>",
      "kty": "RSA",
      "n": "<modulus-value>",
      "use": "sig"
    }
  ]
}
```

Response Fields

Field	Type	Description
`alg`	string	The algorithm used for the key (e.g., RS256).
`e`	string	Public exponent of the RSA key, encoded in Base64URL.
`kid`	string	Key ID — a unique identifier for the key.
`kty`	string	Key type (usually RSA).
`n`	string	The RSA public key modulus, Base64URL-encoded.

Field	Type	Description
`use`	string	How the key is used (e.g., `sig` for signature verification).

`POST /api/auth/token/verify/`

Description

This API endpoint is used to verify the validity of a JWT (JSON Web Token).

It checks three main things:

- The token has not expired.
- The user linked to the token exists in the system.
- The user is active (i.e., not disabled or deleted).

This is useful for confirming that a token should still be trusted before allowing access to a protected resource.

Resquest Parameters

Field	Type	Required	Description
`type`	string	Yes	The type of token (e.g., `"access"` or `"refresh"`).
`token`	string	Yes	The JWT you want to verify.

Example Request

```
curl -X POST \
  'http://<your-server>/api/auth/token/verify/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>' \
  -H 'Content-Type: application/json' \
  -d '{
    "type": "access",
    "token": "<your-jwt-token>"
  }'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "id": 2,
  "username": "exampleuser",
  "first_name": "",
  "last_name": "",
  "email": "",
  "is_active": true,
  "is_superuser": false,
  "is_staff": false,
```

```
  "last_login": "2025-06-11T12:17:53.724142Z",
  "date_joined": "2025-05-23T10:57:26.219710Z",
  "groups": [1],
  "user_permissions": []
}
```

Response Fields

Field	Type	Description
`id`	integer	Unique identifier of the user in the system.
`username`	string	The username associated with the token.
`first_name`	string	User's first name (empty if not provided).
`last_name`	string	User's last name (empty if not provided).
`email`	string	Email address of the user (empty if not provided).
`is_active`	boolean	Indicates whether the user account is currently active.
`is_superuser`	boolean	Indicates if the user has superuser (full admin) privileges.
`is_staff`	boolean	Indicates whether the user can log into the admin site.
`last_login`	string (ISO 8601)	Timestamp of the last time the user logged in.
`date_joined`	string (ISO 8601)	Timestamp of when the user account was created.
`groups`	array of integers	List of group IDs that the user belongs to.
`user_permissions`	array	List of permission identifiers assigned to the user (empty if none).
`password`	string	Hashed version of the user's password (internal use – do not expose).

Users

In the Hub system, **users** represent individuals who have access to the platform through a secure login. Each user has a profile containing basic information such as their name, email, and group membership. The system defines two distinct user types:

- **Admins**: These users have full access to all system features and administrative functions. They can manage other users, configure settings, and generate or revoke API keys.
- **Read-Only Users**: These users have restricted access and can only view data within the Hub. They cannot modify settings, manage users, or perform administrative tasks.

Understanding user roles is essential for managing access control and integrating with the Users API. Only Admins are permitted to perform operations that alter user data or system configuration.

- `username`: A required field (max 150 characters) that can include alphanumeric characters, underscores, and some symbols like @, +, ., and -.
- `first_name` / `last_name`: Optional fields for the user's first and last name (max 150 characters).
- `email`: Optional field for the user's email address.
- `password`: Required field, with restrictions on which characters can be used.

`GET /users/`

Description

The `/users/` endpoint retrieves a list of users registered in the Hub system. This endpoint is primarily accessible to Admin users, who can use it to manage or monitor users within the platform. It supports pagination and returns detailed user information, including group membership.

Request Parameters

Refer to the **Request Parameters** under `GET` **/auth/clientkeys/** to learn more.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/users/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 2,
      "url": "http://your-hub-address:8000/api/users/hubadmin/",
      "username": "hubadmin",
      "first_name": "",
      "last_name": "",
      "email": "",
      "is_active": true,
      "last_login": "2025-06-24T12:59:29.823039Z",
```

```
      "date_joined": "2025-05-23T10:57:26.219710Z",
      "groups": [
        {
          "id": 1,
          "url": "http://your-hub-address:8000/api/groups/1/",
          "name": "Administrators"
        }
      ],
      "profile": null,
      "nas_user": null
    },
    {
      "id": 1,
      "url": "http://your-hub-address:8000/api/users/pixit/",
      "username": "pixit",
      "first_name": "",
      "last_name": "",
      "email": "jenkins@example.com",
      "is_active": true,
      "last_login": "2025-05-14T13:07:57.656518Z",
      "date_joined": "2025-05-14T12:06:59.927211Z",
      "groups": [
        {
          "id": 1,
          "url": "http://your-hub-address:8000/api/groups/1/",
          "name": "Administrators"
        }
      ],
      "profile": null,
      "nas_user": null
    }
  ]
}
```


Response Fields

Field	Type	Description
`count`	integer	Total number of users in the system.
`next`	string/ null	URL to the next page of results, or `null` if there are no more pages.
`previous`	string/ null	URL to the previous page of results, or `null` if on the first page.
`results`	array	List of user objects. Each object contains detailed information about a user.

Each **user object** in `results` includes:

Field	Type	Description
`id`	integer	Unique identifier for the user.
`url`	string	API endpoint URL for the individual user.
`username`	string	User’s login name.

Field	Type	Description
`first_name`	string	User’s first name (can be empty).
`last_name`	string	User’s last name (can be empty).
`email`	string	User’s email address.
`is_active`	boolean	Indicates whether the user account is active.
`last_login`	datetime	Timestamp of the user’s last login.
`date_joined`	datetime	Timestamp of when the user account was created.
`groups`	array	List of groups the user belongs to. Typically used to determine user roles.
`profile`	object/ null	Reserved for extended profile information. Currently `null`.
`nas_user`	object/ null	Reserved for NAS-related user info. Reserved implies the field is unused, which means `null` unless the user is NAS enabled.

POST /users/

Description

This endpoint creates a new user in the Hub system. Only **Admin users** are authorized to create new user accounts.

Users can be assigned to one or more groups and **optionally provided with extended profile details**. By default, the new user is active. The `nas_user` field is included in the response and can be left `null` unless your system integrates with a NAS (Network-Attached Storage) component.

Request Parameters

Name	Type	Required	Description
`username`	string	Yes	The unique username for the new user.
`password`	string	Yes	The password for the user. It will be securely hashed in the response.
`first_name`	string	No	User’s first name.
`last_name`	string	No	User’s last name.
`email`	string	No	User’s email address.
`groups`	array	Yes	List of group names the user should belong to (e.g., `["Users"]`, `["Administrators"]`).
`profile`	object	No	Optional user profile with extended info. See fields below.

Profile Subfields

Field	Type	Description
`phone`	string	User’s phone number.
`timezone`	string	User’s preferred timezone. e.g. `Europe/London`
`country`	string	User’s country.
`department`	string	Department to which the user belongs.

Field	Type	Description
`line_manager`	string	Name of the user’s line manager.
`default_site`	integer	ID of the default site assigned to the user.
`home_site`	integer	ID of the user’s home site.

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/users/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
    "username": "Example2",
    "password": "*********",
    "first_name": "Example",
    "last_name": "User",
    "email": "user@gmail.com",
    "groups": ["Users"],
    "profile": {
      "phone": "12345678",
      "timezone": "string",
      "country": "string",
      "department": "string",
      "line_manager": "string",
      "default_site": 1,
      "home_site": 1
    }
  }'
```

Successful Response

HTTP Status Code: `201 Created`

Content-Type: `application/json`

Example Response

```
{
  "username": "Example2",
  "password": "********",
  "first_name": "Example",
  "last_name": "User",
  "email": "user@gmail.com",
  "groups": ["Users"],
  "profile": {
    "phone": "12345678",
    "timezone": "string",
    "country": "string",
    "department": "string",
    "line_manager": "string",
    "default_site": 1,
    "home_site": 1
```

```
  },
  "nas_user": null
}
```

Response Fields

Note: The response fields are described under the request parameters section above. Please refer to that section for details.

POST /users/change_password/

Description

This endpoint allows an authenticated **user to securely change their password**. The request must include the user's current (old) password and the new password they wish to set.

Only the currently logged-in user can change their own password using this endpoint. Admin users must use the user management interface or a different admin-level endpoint to reset other users' passwords.

Request Parameters

Name	Type	Required	Description
old_password	string	Yes	The user's current password.
new_password	string	Yes	The new password to set. Must meet password policies.

Note: When creating or changing a password, the following rules must be met:

- Must be at least **8 characters** long
- **Cannot be too similar** to the username
- Must include **at least one digit**
- Must include **at least one uppercase letter**
- Must include **at least one special character** (e.g., !@#$%^&*)

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/users/change_password/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
    "old_password": "*********",
    "new_password": "**********"
  }'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "message": "password successfully changed"
}
```

Error Response

HTTP Code	Description	Example Message
400	Invalid old password or weak new password	`{"old_password": ["Incorrect password."]}` or `{"new_password": ["This password is too short."]}`
401	Unauthorized (missing or invalid token)	`{"detail": "Authentication credentials were not provided."}`

GET /users/{username}/

This endpoint retrieves detailed information for a specific user identified by their **username**. It returns the user's basic info, group memberships, profile details, and NAS user reference (if applicable).

This endpoint is accessible to Admin users for reviewing user configurations, or by users retrieving their own account data (depending on permissions).

Request Parameters

Name	Type	Required	Description
username	string	Yes	The username of the user to retrieve. Must match pattern: `[\w\.\+\-_@]+`

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/users/Example/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "id": 4,
```

```
  "url": "http://your-hub-address:8000/api/users/Example/",
  "username": "Example",
  "first_name": "Example",
  "last_name": "User",
  "email": "user@gmail.com",
  "is_active": true,
  "last_login": null,
  "date_joined": "2025-06-25T11:53:37.572849Z",
  "groups": [
    {
      "id": 2,
      "url": "http://your-hub-address:8000/api/groups/2/",
      "name": "Users"
    }
  ],
  "profile": {
    "phone": "string",
    "timezone": "string",
    "country": "string",
    "department": "string",
    "line_manager": "string",
    "default_site": 1,
    "home_site": 1
  },
  "nas_user": null
}
```


Response Fields

To view the information related to response fields and profile subfields , refer to `GET` & `POST` **Users** endpoint.

`PATCH /users/{username}/`

Description

This endpoint allows partial updates to an existing user's information using their username. You can update any combination of the following fields: basic user details, group membership, password, and extended profile information. This endpoint also allows you to `PATCH` `NAS` user settings, such as enabling or disabling `NAS` access.

Only Admin users or authorized users (e.g., self-edit under permissions) can perform this action.

Request Path Parameters

Name	Type	Required	Description
username	string	Yes	The username of the user to update.

Request Body Parameters (Partial Update)

Name	Type	Description
password	string	(Optional) New password. Must comply with Password Requirements.
first_name	string	(Optional) Updated first name.
last_name	string	(Optional) Updated last name.
email	string	(Optional) Updated email address.
groups	array	(Optional) List of group names to which the user should belong.
profile	object	(Optional) User's profile data. See subfields below.

Note: To view the information related to profile subfields , refer to `POST Users` endpoint.

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/users/Example2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
    "password": "*******",
    "first_name": "Updated",
    "last_name": "Info",
    "email": "user@example.com",
    "groups": ["Users"],
    "profile": {
      "phone": "string",
      "timezone": "string",
      "country": "string",
      "department": "string",
      "line_manager": "string",
      "default_site": 1,
      "home_site": 1
    }
  }'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "first_name": "Updated",
  "last_name": "Info",
  "email": "user@example.com",
  "groups": [
    "Users"
```

```
  ],
  "profile": {
    "phone": "string",
    "timezone": "string",
    "country": "string",
    "department": "string",
    "line_manager": "string",
    "default_site": 1,
    "home_site": 1
  },
  "nas_user": null
}
```

Response Fields

The response fields are the same as the request parameters. Please refer to the section above for a better understanding.

`DELETE /users/{username}/`

Description

This endpoint permanently deletes a user from the system based on their **username**.

Only users with **Administrator privileges** can perform this action. Once deleted, the user account **cannot be recovered** unless the system provides a separate restore or archival mechanism.

Request Parameters

Name	Type	Required	Description
username	string	Yes	The username of the user to delete. Must match pattern: `[\w\.\+\-_@]+`

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/users/Example2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `204 No Content`

Content-Type: `None`

Body: Empty

This status code indicates that the user was successfully deleted. No response body is returned.

Note:

- This action **cannot be undone**. Make sure the user data is no longer needed or has been backed up before deletion.
- If the specified username does not exist, the API may return a `404 Not Found` error.
- System-protected users (like default admin accounts) might be restricted from deletion, depending on implementation.

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

Description

This endpoint activates a user account identified by the provided username. Activation sets the user's `is_active` flag to `true`, allowing them to log in and access the system, assuming they meet other authentication requirements.

Request Parameters

Name	Type	Required	Description
username	string	Yes	The username of the user to delete. Must match pattern: `[\w\.\+\-_@]+`

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/users/Example3/activate/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -d ''
```

Successful Response

HTTP Status Code: `201 Created`

Content-Type: `application/json`

Description: Returns the full user object with `is_active` set to `true`.

Example Response

```
{
  "id": 6,
  "url": "http://your-hub-address:8000/api/users/Example3/",
  "username": "Example3",
  "first_name": "Example",
  "last_name": "User",
  "email": "",
  "is_active": true,
  "last_login": null,
  "date_joined": "2025-06-25T12:01:34.447630Z",
```

```
  "groups": [
    {
      "id": 2,
      "url": "http://your-hub-address:8000/api/groups/2/",
      "name": "Users"
    }
}
```

Response Fields

The response fields are identical to those returned by the `GET` **/users/** endpoint. Please refer to the section above for more details.

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

Description

This endpoint deactivates a user account identified by the `username` parameter. Deactivation sets the `is_active` field of the user to `false`, preventing login or API access.

Useful for managing user access without permanently deleting accounts.

Request Parameters

Name	Type	Required	Description
username	string	Yes	The username of the user to delete. Must match pattern: `[\w\.\+\-_@]+`

Example Request

```
curl -X 'POST' \
  'http://10.201.2.224:8000/api/users/Example3/deactivate/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -d ''
```

Successful Response

HTTP Status Code: `201 Created`

Content-Type: `application/json`

Description: Returns the full user object with `is_active` set to `false`.

Example Response

```
{
  "id": 6,
  "url": "http://your-hub-address:8000/api/users/Example3/",
  "username": "Example3",
  "first_name": "Example",
  "last_name": "User",
```

```
  "email": "",
  "is_active": false,
  "last_login": null,
  "date_joined": "2025-06-25T12:01:34.447630Z",
  "groups": [
    {
      "id": 2,
      "url": "http://your-hub-address:8000/api/groups/2/",
      "name": "Users"
    }
  ],
  "profile": {
    "phone": "12345678",
    "timezone": "string",
    "country": "string",
    "department": "string",
    "line_manager": "string",
    "default_site": 1,
    "home_site": 1
  },
  "nas_user": null
}
```


Response Fields

Please refer to `PATCH` **/users/{username}/** response fields.

My Permissions

The `/my-permissions/` endpoint allows an authenticated user to view all permissions assigned to them within the Ngenea Hub. These permissions may come from group memberships, direct assignments, or object-level grants.

The response includes details such as permission ID, name, application label, codename, and the associated model. This endpoint is strictly limited to the requesting user's own permissions and does not expose permissions of other users.

GET /my-permissions/

Description

This endpoint allows the `authenticated user` to view only their own permissions in the Ngenea Hub.

It returns all permissions the user has through:

- Group membership (`group_permissions`)
- Direct assignment (`user_permissions`)
- Object-level permissions (`group_object_permissions`, `user_object_permissions`)

> **Note:** Users cannot view permissions of other users via this endpoint.

Request Parameters: `None`

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/my-permissions/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "group_permissions": [
    {
      "id": 290,
      "name": "Can add log entry",
      "app_label": "admin",
      "codename": "add_logentry",
      "model": "logentry"
    },
    {
      "id": 2,
      "name": "Can add alert",
      "app_label": "alerts",
      "codename": "add_alert",
      "model": "alert"
    }
  ],
  "user_permissions": [],
  "group_object_permissions": [],
  "user_object_permissions": []
}
```

Response Fields

Field	Type	Description
`group_permissions`	array of objects	Permissions inherited through the user's group(s)
`user_permissions`	array of objects	Permissions assigned directly to the user
`group_object_permissions`	array	Object-level permissions via group (if any)
`user_object_permissions`	array	Object-level permissions assigned directly (if any)

Each permission object contains:

Subfield	Type	Description
`id`	integer	Unique identifier of the permission
`name`	string	Descriptive name of the permission
`app_label`	string	Application label (e.g., `"admin"`)
`codename`	string	Permission codename used internally
`model`	string	Model associated with the permission

Permissions

GET /permissions/

Description

The `/permissions/` endpoint in Ngenea Hub provides a comprehensive list of **all system-wide permissions** configured in the backend. This endpoint is typically used by administrators, integrators, or role-management interfaces to list available permissions that can be assigned to users or groups within Ngenea Hub.

> **Note:** This API returns all possible permissions in Ngenea Hub, not just those granted to the requesting user. To see permissions specific to the logged-in user, use the `/my-permissions/` endpoint.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/permissions/?
page=1&page_size=20' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "count": 349,
```

```
  "next": "http://your-hub-address:8000/api/permissions/?page=2",
  "previous": null,
  "results": [
    {
      "id": 1,
      "name": "Can view health endpoint",
      "app_label": "filebrowser",
      "codename": "view_health",
      "model": "node"
    },
    {
      "id": 2,
      "name": "Can add alert",
      "app_label": "alerts",
      "codename": "add_alert",
      "model": "alert"
    },
    {
      "id": 3,
      "name": "Can change alert",
      "app_label": "alerts",
      "codename": "change_alert",
      "model": "alert"
    }
    // ... more permissions
  ]
}
```

Response Fields

Field	Type	Description
`count`	integer	Total number of available permissions
`next`	string / null	URL to the next page of results, if applicable
`previous`	string / null	URL to the previous page of results, if applicable
`results`	array of objects	List of permission objects for the current page or result set

Each permission object contains:

Subfield	Type	Description
`id`	integer	Unique ID of the permission
`name`	string	Descriptive, human-readable name
`app_label`	string	Application/module the permission belongs to
`codename`	string	Internal codename used for logic checks
`model`	string	Model name the permission applies to

`GET /permissions/{id}/`

Description

This endpoint retrieves **detailed information** about a **specific permission** in Ngenea Hub by its unique ID.

Request Parameters

Name	Type	Location	Required	Description
`id`	string	path	Yes	The unique ID of the permission to retrieve

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/permissions/7/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "id": 7,
  "name": "Can change bucket",
  "app_label": "external_targets",
  "codename": "change_bucket",
  "model": "bucket"
}
```

Response Fields

Please refer to the response fields section under `GET` **/permissions/**.

Groups

In Ngenea Hub, **groups** are fundamental for organizing users and managing their access to various **Sites** and **Spaces**. Groups allow administrators to define roles and permissions for multiple users at once, ensuring efficient access control across the system. By categorizing users into different groups, Hub administrators can easily assign **read-only access**, **administrative rights**, or specific permissions based on the group a user belongs to.

Ngenea Hub comes with default groups such as **Administrators** and **Users**, with predefined roles. However, custom groups can also be created to meet specific organizational needs. When integrated with LDAP or Active Directory (AD), group memberships can be automatically synced, allowing external directory users to inherit group-based permissions in Ngenea Hub.

> **Note:** Refer to Groups and Users to learn more about groups.

GET /groups/

Description

The `GET /groups/` endpoint allows users to retrieve a paginated list of groups in the Ngenea Hub. This endpoint provides information about each group, including the group name, description, associated users, and the permissions assigned to the group.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/groups/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

This request retrieves the first page of groups, with the default page size of 20.

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "name": "Administrators",
      "users": [
        {
          "username": "pixit",
          "first_name": "",
          "last_name": "",
          "email": "jenkins@example.com",
          "date_joined": "2025-05-14T12:06:59.927211Z",
          "last_login": "2025-05-14T13:07:57.656518Z"
        },
```

```
        {
          "username": "hubadmin",
          "first_name": "",
          "last_name": "",
          "email": "",
          "date_joined": "2025-05-23T10:57:26.219710Z",
          "last_login": "2025-07-04T14:17:23.449154Z"
        }
      ],
      "description": "Default group for users with administrative
access",
      "permissions": [290, 291, 292, 293,],
      "object_permissions": [],
      "nas_group": null
    },
    {
      "id": 2,
      "name": "Users",
      "users": [
        {
          "username": "Test",
          "first_name": "Test",
          "last_name": "User",
          "email": "test@gmail.com",
          "date_joined": "2025-06-25T11:43:39.651419Z",
          "last_login": null
        },
        {
          "username": "Example",
          "first_name": "Example",
          "last_name": "User",
          "email": "user@gmail.com",
          "date_joined": "2025-06-25T11:53:37.572849Z",
          "last_login": null
        }
      ],
      "description": "Default group for users with read-only
access",
      "permissions": [305, 17, 109, 117,],
      "object_permissions": [],
      "nas_group": null
    },
    {
      "id": 3,
      "name": "Test",
      "users": [],
      "description": null,
      "permissions": [305, 17, 109, 117, 113, 26, 29,],
      "object_permissions": [],
      "nas_group": {"gid": 100000}
    }
  ]
}
```

Response Fields

Group List Response Fields

Field Name	Description	Type	Example
`count`	The total number of groups available.	Integer	3
`next`	URL to the next set of results (if any).	String or null	null
`previous`	URL to the previous set of results (if any).	String or null	null
`results`	A list of group objects returned by the API.	Array of Objects	See below for details

Group Object Fields

Field Name	Description	Type	Example
`id`	Unique identifier for the group.	Integer	1
`name`	The name of the group.	String	“Administrators”
`users`	A list of user objects associated with the group. Each user object contains user information.	Array of Objects	See below for details
`description`	A description of the group. If not provided, this will be null.	String or null	“Default group for users with administrative access”
`permissions`	A list of permission IDs assigned to the group.	Array of Integers	[290, 291, 292, 293, ...]
`object_permissions`	A list of object-level permissions (if any).	Array of Objects	[] (empty array)
`nas_group`	Information related to the group’s NAS configuration. If not applicable, this will be null.	Object or null	null or `{ "gid": 100000 }`

User Object Fields - For each user in the `users` array:

Field Name	Description	Type	Example
`username`	Username of the user.	String	“pixit”
`first_name`	The user’s first name.	String	“” (empty string)
`last_name`	The user’s last name.	String	“” (empty string)

Field Name	Description	Type	Example
`email`	The user's email address.	String	"jenkins@example.com"
`date_joined`	The date when the user joined the system.	DateTime	"2025-05-14T12:06:59.927211Z"
`last_login`	The last time the user logged in.	DateTime or null	"2025-05-14T13:07:57.656518Z"

`POST /groups/`

Description

This API endpoint allows you to create a new group. By sending a `POST` request to this endpoint with the necessary details, a new group is created in the system. This includes specifying the group's name, users, description, and permissions.

Request Parameters

Name	Description	Required
`name`	The name of the group being created (e.g., "Test Group").	Yes
`users`	A list of user usernames to add to this group. For example: `["Test", "Example", "Example3"]`.	Yes
`description`	A brief description of the group (e.g., "A group for testing purposes").	No
`permissions`	A list of permission IDs assigned to this group (e.g., `[305, 17, 109]`).	Yes

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/groups/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Test Group",
  "users": [
    "Test",
    "Example",
    "Example3"
  ],
  "description": "A group for testing purposes",
  "permissions": [
    305,
    17,
    109
  ]
}'
```

Successful Response

HTTP Status Code: `201 Created`

Content-Type: `application/json`

The group has been successfully created.

Example Response


```
{
  "id": 5,
  "name": "Test Group",
  "users": [
    "Test",
    "Example",
    "Example3"
  ],
  "description": "A group for testing purposes"
  "permissions": [
    305,
    17,
    109
  ],
  "object_permissions": [],
  "nas_group": null
}
```


Response Fields

Name	Description
`id`	The unique identifier for the newly created group.
`name`	The name of the group (e.g., “Test Group”).
`users`	A list of usernames added to the group (e.g., `["Test", "Example", "Example3"]`).
`description`	The description of the group (e.g., “A group for testing purposes”).
`permissions`	A list of permission IDs associated with this group (e.g., `[305, 17, 109]`).
`object_permissions`	A list of object-level permissions. In this case, it is an empty list, meaning no object-level permissions are assigned.
`nas_group`	The NAS (Network Attached Storage) group associated with this group. In this case, it is `null` as no NAS group is provided.

`GET /groups/{id}/`

Description

This API endpoint retrieves detailed information about a specific group using its `ID`. It provides data such as:

- The `name` of the group
- A list of `users` assigned to the group (with user details like usernames, emails, join date, last login)
- The `permissions` granted to the group
- Any `object permissions` for the group
- The `description` of the group (if available)

Request Parameters

Name	Description	Required	Type	Example
`id`	The unique identifier of the group you want to retrieve. This is a path parameter and should be included in the URL.	Yes	String	4

Example Request

```
curl -X GET 'http://your-hub-address:8000/api/groups/4/' \
-H 'accept: application/json' \
-H 'Authorization: Api-Key YOUR_API_KEY'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "id": 4,
  "name": "API Test Group",
  "users": [
    {
      "username": "pixit",
      "first_name": "",
      "last_name": "",
      "email": "jenkins@example.com",
      "date_joined": "2025-05-14T12:06:59.927211Z",
      "last_login": "2025-05-14T13:07:57.656518Z"
    },
    {
      "username": "hubadmin",
      "first_name": "",
      "last_name": "",
      "email": "",
      "date_joined": "2025-05-23T10:57:26.219710Z",
      "last_login": "2025-07-04T14:17:23.449154Z"
    },
    {
      "username": "Test",
```

```
      "first_name": "Test",
      "last_name": "User",
      "email": "test@gmail.com",
      "date_joined": "2025-06-25T11:43:39.651419Z",
      "last_login": null
    },
    {
      "username": "Example",
      "first_name": "Example",
      "last_name": "User",
      "email": "user@gmail.com",
      "date_joined": "2025-06-25T11:53:37.572849Z",
      "last_login": null
    },
    {
      "username": "Example3",
      "first_name": "Example",
      "last_name": "User",
      "email": "",
      "date_joined": "2025-06-25T12:01:34.447630Z",
      "last_login": null
    }
  ],
  "description": "Group for API testing users",
  "permissions": [
    305,
    17,
    109,
    117,
    113
  ],
  "object_permissions": [],
  "nas_group": null
}
```


Response Fields

Refer to response fields under `POST` **/Groups**.

`PATCH /groups/{id}/`

Description

The **PATCH /groups/{id}/** endpoint is used to update an existing group. You can modify the group's name, users, permissions, and other attributes. This request is typically used to make partial updates to a group without changing the entire resource.

In this case, we are updating a group with the `ID 5` by:

- Modifying its name to `"Test Group"`.
- Adding/Removing `users` in the users array (in this case, users "Test" and "Example3").

- Modifying the `permissions` array (adding permission IDs 305, 17, and 109).

Request Parameters

Name	Description	Type	Required	Example
`id`	The unique identifier of the group that you want to update (path parameter). This is the ID of the group.	String	Yes	`5`
`data` (body)	An object containing the data to update the group with. The fields that can be included are: `name`, `users`, `permissions`.	Object	Yes	`{ "name": "Test Group", "users": ["Test", "Example3"], "permissions": [305, 17, 109] }`

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/groups/5/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Test Group",
  "users": [
    "Test",
    "Example3"
  ],
  "permissions": [
    305,
    17,
    109
  ]
}'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "id": 5,
  "name": "Test Group",
  "users": [
    "Test",
    "Example3"
  ],
  "description": null,
```

```
  "permissions": [
    305,
    17,
    109
  ],
  "object_permissions": [],
  "nas_group": null
}
```

Response Fields

Field Name	Description	Type	Example
`id`	The unique identifier of the group.	Integer	`5`
`name`	The updated name of the group.	String	`"Test Group"`
`users`	A list of the usernames associated with this group.	Array of Strings	`["Test", "Example3"]`
`description`	The description of the group. If no description is provided, this will be `null`.	String or null	`null`
`permissions`	A list of permission IDs assigned to this group.	Array of Integers	`[305, 17, 109]`
`object_permissions`	A list of object-level permissions assigned to the group (if any).	Array of Objects	`[]` (empty array)
`nas_group`	Information about the group's NAS configuration (if any). If not applicable, this will be `null`.	Object or null	`null`

DELETE /groups/{id}/

Description

This API endpoint is used to delete a specific group from the system based on its ID.

Request Parameters

Name	Description
`id` (path parameter)	The unique identifier of the group you wish to delete. It is provided in the URL path (e.g., `4`).

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/groups/4/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

No Request Body

Since this is a `DELETE` request, no body is required in the request. Only the ID is used to identify the group for deletion.

Successful Response

HTTP Status Code: `204 (No Content)` – This status code indicates that the group was successfully deleted, and there is no additional content to return in the response body.

Content-Type: The response is in `JSON` format (application/json), but there is no body content as the request was successful (since it's a DELETE operation).

API Endpoint - Policies

A policy (specifically a `policy-based tiering` workflow) is an advanced data orchestration tool that automates the movement of data between storage pools or from pools to Ngenea targets. Policies are driven by conditions such as file age, size, access time, or storage pool utilization levels, enabling dynamic data management.

Difference from Workflow:

- Policy: Operates vertically, automating data movement within storage tiers based on predefined conditions.
- **Workflow:** Operates horizontally, focusing on moving data into or out of a site.

API Reference Location

All endpoints, parameters, and schemas can be explored through the interactive API documentation available at:

```
http(s)://<your-hub-address>/api/docs/
```

> **Note:** Replace `your-hub-address` with the actual IP address or domain of your Ngenea Hub instance.

GET /policies/

Description

This endpoint allows you to retrieve a list of policies, with support for pagination. It returns a collection of policies configured in the system, including their details such as schedule, triggers, conditions, and other related configurations.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.

Name	Type	Required	Description
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/policies/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "schedule": {
        "id": 1,
        "timezone": "UTC",
        "minute": "0",
        "hour": "0",
        "day_of_week": "1",
        "day_of_month": "*",
        "month_of_year": "*",
        "time": "00:00:00",
        "first_occur": false,
        "start_date_time": null
      },
      "triggers": [
        {
          "id": 1,
          "pool_name": "sas1",
          "max_utilisation": null,
          "upper_threshold": null,
          "lower_threshold": null,
          "premigrate_threshold": null,
          "is_cloud": false
        }
      ],
      "order": {
        "id": 1,
```

```
        "by": "file_size",
        "reverse": false
      },
      "condition_groups": [],
      "name": "Example",
      "policy_type": "MIGRATION",
      "filesystem": "mmfs1",
      "created": "2025-06-27T14:14:31.272573Z",
      "enabled": true,
      "threads": 4,
      "site": 1,
      "spaces": [
        2
      ]
    }
  ]
}
```


Response Fields

Field	Description
`count`	The total number of policies available in the system (e.g., `1` policy).
`next`	URL of the next page of results (`null` if there are no more pages).
`previous`	URL of the previous page of results (`null` if there is no previous page).
`results`	An array of policy objects containing the policy details.

Policy Object (Inside results Array) - Each policy object contains the following fields:

Field	Description
`id`	The unique identifier for the policy (e.g., `1`).
`schedule`	The schedule for when the policy should run, including timing and frequency.
`triggers`	The conditions that trigger the policy, such as storage pool settings or thresholds.
`order`	Defines how the data should be processed (e.g., by file size).
`condition_groups`	Groups of conditions that must be met for the policy to be executed (e.g., `[]`).
`name`	The name of the policy (e.g., `"Example"`).
`policy_type`	The type of policy (e.g., `"MIGRATION"` for moving data).
`filesystem`	The filesystem where the policy is applied (e.g., `"mmfs1"`).
`created`	Timestamp of when the policy was created.
`enabled`	Whether the policy is enabled (`true` means enabled).
`threads`	The number of threads used for policy execution (e.g., `4`).
`site`	The site to which the policy is applied (e.g., `1`).
`spaces`	The storage spaces involved in the policy (e.g., `2`).

POST /policies/

Description

This API endpoint is used to create a new policy, specifying how data should be managed, processed, or migrated between storage locations.

The policy can be scheduled to run at a specific time and trigger based on conditions like storage pool utilization, file size, and other predefined conditions.

The policy can migrate data between storage tiers or to different cloud targets based on these rules.

Request Parameters

Name	Description
`data` (required)	The main body of the request. This contains the configuration for the new policy.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/policies/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```


```
{
  "schedule": {
    "timezone": "UTC",
    "minute": "0",
    "hour": "0",
    "day_of_week": "1",
    "day_of_month": "*",
    "month_of_year": "*",
    "time": "00:00:00",
    "first_occur": false,
    "start_date_time": null
  },
  "triggers": [
    {
      "pool_name": "sas1",
      "max_utilisation": 0,
      "lower_threshold": 0,
      "upper_threshold": 0,
      "premigrate_threshold": 0,
      "is_cloud": true
    }
  ],
  "order": {
    "by": "file_size",
    "reverse": false
  },
  "name": "Test Policy",
```

```
  "policy_type": "MIGRATION",
  "filesystem": "mmfs1",
  "created": "2025-06-27T14:14:31.272573Z",
  "enabled": true,
  "threads": 4,
  "site": 1,
  "spaces": [
    2
  ]
}
```


Request Body Fields

`schedule`: Defines when the policy should execute.

Field	Description
`timezone`	The timezone to use for scheduling (set to `"UTC"`).
`minute`	Minute of the hour when the policy triggers (set to `0`).
`hour`	Hour of the day for policy execution (set to `0`, i.e., midnight).
`day_of_week`	Day of the week the policy runs (set to `1`, i.e., Monday).
`day_of_month`	Day of the month (set to `"*"`, meaning any day).
`month_of_year`	Month of the year (set to `"*"`, meaning any month).
`time`	Exact time the policy will trigger (set to `00:00:00`).
`first_occur`	Boolean flag for immediate first run (set to `false`).
`start_date_time`	When the policy becomes active (set to `null`).

> **Note:** This configuration schedules the policy to run at **midnight every Monday (UTC time)**.

`triggers`: Specifies the conditions that activate the policy.

Field	Description
`pool_name`	The name of the storage pool (e.g., `"sas1"`).
`max_utilisation`	Max storage pool utilization (set to `0`, no limit).
`lower_threshold`	Lower utilization threshold (set to `0`).
`upper_threshold`	Upper utilization threshold (set to `0`).
`premigrate_threshold`	Threshold for pre-migration actions (set to `0`).
`is_cloud`	Indicates if the pool is cloud-based (set to `true`).

> **Note:** This configuration **triggers the policy for the cloud-based pool `sas1`** regardless of utilization.

`order`: Defines how files should be processed.

Field	Description
`by`	The file order criteria (e.g., `"file_size"` to prioritize larger files).
`reverse`	Boolean flag to reverse order (set to `false`, meaning no reversal).

> **Note:** Files will be processed from **largest to smallest**.

General Policy Fields

Field	Description
`name`	The name of the policy (e.g., `"Test Policy"`).
`policy_type`	The type of policy (e.g., `"MIGRATION"`).
`filesystem`	Filesystem where the policy applies (e.g., `"mmfs1"`).
`created`	Timestamp of policy creation (e.g., `"2025-06-27T14:14:31.272573Z"`) .
`enabled`	Whether the policy is active (`true` means enabled).
`threads`	Number of threads for processing (e.g., `4`).
`site`	ID of the associated site (e.g., `1`).
`spaces`	List of storage space IDs (e.g., `[2]`).

Successful Response

HTTP Status Code: `201 (Created)`

Content-Type: `application/json`

This indicates that the policy was successfully created on the server.

Example Response


```
{
  "id": 1,
  "schedule": {
    "timezone": "UTC",
    "minute": "0",
    "hour": "0",
    "day_of_week": "1",
    "day_of_month": "*",
    "month_of_year": "*",
    "time": "00:00:00",
    "first_occur": false,
    "start_date_time": null
  },
  "triggers": [
    {
      "pool_name": "sas1",
      "max_utilisation": 0,
      "lower_threshold": 0,
      "upper_threshold": 0,
      "premigrate_threshold": 0,
      "is_cloud": true
    }
  ],
  "order": {
    "by": "file_size",
    "reverse": false
  },
  "name": "Test Policy",
  "policy_type": "MIGRATION",
```

```
  "filesystem": "mmfs1",
  "created": "2025-06-27T14:14:31.272573Z",
  "enabled": true,
  "threads": 4,
  "site": 1,
  "spaces": [2]
}
```

Response Fields

The response fields are the same as the request parameters. Please refer to the section above for a better understanding.

GET /policies/{id}/

Description

This API endpoint retrieves detailed information about a specific policy identified by the `id` parameter. It returns the policy's configuration, including its schedule, triggers, order, and other associated details.

Request Parameters

Name	Description
`id`	**(Required)** The unique identifier of the policy you want to retrieve. It is a string value and must be provided as part of the URL path.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/policies/1/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "id": 1,
  "schedule": {
    "id": 1,
    "timezone": "UTC",
    "minute": "0",
    "hour": "0",
    "day_of_week": "1",
    "day_of_month": "*",
    "month_of_year": "*",
    "time": "00:00:00",
```

```
    "first_occur": false,
    "start_date_time": null
  },
  "triggers": [
    {
      "id": 1,
      "pool_name": "sas1",
      "max_utilisation": null,
      "upper_threshold": null,
      "lower_threshold": null,
      "premigrate_threshold": null,
      "is_cloud": false
    }
  ],
  "order": {
    "id": 1,
    "by": "file_size",
    "reverse": false
  },
  "condition_groups": [],
  "name": "Example",
  "policy_type": "MIGRATION",
  "filesystem": "mmfs1",
  "created": "2025-06-27T14:14:31.272573Z",
  "enabled": true,
  "threads": 4,
  "site": 1,
  "spaces": [
    2
  ]
}
```


Response Fields

Field Name	Description
`id`	The unique identifier for the policy. In this case, the policy ID is `1`.
`schedule`	Defines when the policy should execute based on the provided time parameters.
`triggers`	Defines the conditions that will trigger the policy.
`order`	Defines how the files will be processed during policy execution.
`condition_groups`	An empty list indicating that no condition groups are defined.
`name`	The name of the policy (e.g., `"Example"`).
`policy_type`	The type of policy (e.g., `"MIGRATION"` for data migration).
`filesystem`	The filesystem to which the policy applies (e.g., `"mmfs1"`).
`created`	The timestamp indicating when the policy was created (e.g., `2025-06-27T14:14:31.272573Z`).
`enabled`	Boolean indicating if the policy is currently active (`true`).
`threads`	Number of threads used to process files (e.g., `4`).
`site`	Site ID where the policy applies (e.g., `1`).

Field Name	Description
`spaces`	List of space IDs associated with this policy (e.g., `[2]`).

Nested Field: `schedule`

Field Name	Description
`schedule.id`	The unique identifier of the schedule configuration.
`schedule.timezone`	The timezone in which the policy is scheduled (e.g., `"UTC"`).
`schedule.minute`	The minute of the hour when the policy triggers (e.g., `0`).
`schedule.hour`	The hour of the day when the policy triggers (e.g., `0` for midnight).
`schedule.day_of_week`	Day of the week the policy runs (e.g., `1` for Monday).
`schedule.day_of_month`	Day of the month (set to `"*"`, meaning any day).
`schedule.month_of_year`	Month of the year (set to `"*"`, meaning any month).
`schedule.time`	Exact time the policy will trigger (e.g., `00:00:00`).
`schedule.first_occur`	Whether to start immediately on the first occurrence (`false`).
`schedule.start_date_time`	Start date and time when the policy becomes active (set to `null`).

Nested Field: `triggers`

Field Name	Description
`triggers.id`	The unique identifier of the trigger configuration.
`triggers.pool_name`	Storage pool name for the trigger (e.g., `"sas1"`).
`triggers.max_utilisation`	Maximum utilization threshold (set to `null`).
`triggers.upper_threshold`	Upper utilization threshold (set to `null`).
`triggers.lower_threshold`	Lower utilization threshold (set to `null`).
`triggers.premigrate_threshold`	Pre-migration threshold (set to `null`).
`triggers.is_cloud`	Whether the pool is cloud-based (`false`).

Nested Field: `order`

Field Name	Description
`order.id`	The unique identifier of the order configuration.
`order.by`	Criterion used for ordering files (e.g., `"file_size"`).
`order.reverse`	Whether to reverse the order (`false`).

PATCH /policies/{id}/

Description

This API endpoint allows updating an existing policy identified by its `id`. The `PATCH` request allows partial updates, meaning you can modify only the fields that need to be changed, such as the policy name, schedule, triggers, or other attributes.

Request Parameters

Name	Description
`id`	**(Required)** The unique identifier of the policy you want to update. It should be provided as part of the URL path.
`data`	**(Required)** A JSON object containing the updated fields of the policy.

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/policies/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json'
```


```
  "schedule": {
    "timezone": "UTC",
    "minute": "0",
    "hour": "0",
    "day_of_week": "1",
    "day_of_month": "*",
    "month_of_year": "*",
    "time": "00:00:00",
    "first_occur": false,
    "start_date_time": null
  },
  "triggers": [
    {
      "pool_name": "sas1",
      "max_utilisation": 0,
      "lower_threshold": 0,
      "upper_threshold": 0,
      "premigrate_threshold": 0,
      "is_cloud": true
    }
  ],
  "order": {
    "by": "file_size",
    "reverse": false
  },
  "name": "Test Policy Updated",
  "policy_type": "MIGRATION",
  "filesystem": "mmfs1",
  "created": "2025-06-27T14:14:31.272573Z",
  "enabled": true,
  "threads": 4,
  "site": 1,
  "spaces": [
```

```
    2
  ]
```

Note: For example purpose, we have updated only the name of the policy to `Test Policy Updated` and the creation timestamp to reflect the update time (“2025-07-07T16:42:36.971472Z”).

You can modify other parameters such as the schedule, triggers, and other settings as per your specific requirements. All other configurations, including the filesystem (`mmfs1`) and storage pool trigger (`sas1`), remain the same or can be adjusted based on your needs.

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

The server successfully processes the request, and the policy is updated with the new values.

Response Fields

Note: Refer to response fields under `GET` **/POLICES/ID**.

DELETE /policies/{id}/

Description

This API endpoint is used to delete a policy identified by its `id`. The `DELETE` request permanently removes the policy from the system. The deletion is typically irreversible, and once deleted, the policy cannot be restored.

Request Parameters

Name	Description
`id`	**(Required)** The unique identifier of the policy that you want to delete. It should be provided as part of the URL path.

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/policies/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `204 (No Content)`

The server successfully processes the request to delete the policy. A `204 No Content` status code means that the request was successful, but there is no content to return in the response body. It indicates that the policy has been deleted.

Content-Type: `application/json`

`POST /policies/{id}/run/`

Description

The `POST` **/policies/{id}/run/** endpoint allows you to manually trigger the execution of a policy identified by its `id`. This action forces the policy to run immediately, bypassing its regular schedule. The policy will execute based on the predefined conditions and triggers that were set during its creation.

When you make this request, a unique `task_id` is returned, which can be used to track the progress and status of the task.

This endpoint doesn't require any additional parameters; the policy runs according to its current configuration.

It is particularly useful for scenarios where immediate execution is necessary, such as during urgent data migration tasks or testing, without waiting for the next scheduled trigger.

Request Parameters

Name	Description
`id`	**(Required)** The ID of the policy to run. It should be provided in the URL path.

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/policies/1/run/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
```

This response indicates that the task has been successfully triggered. You can now track the task using the provided `task_id`.

Example Response

```
{
  "task_id": "ba14329f-7f77-4c60-9301-7bdfe36199b5"
}
```

Response Fields

Name	Description
`task_id`	A unique identifier for the task initiated by this request. This ID can be used to track the progress or status of the policy execution.

API Endpoints - File Management and Storage

File

GET /file/

Description

The `/file/` API endpoint not only lists the files and folders under a specified path for a given site but also returns important metadata (details) about each item. These details provide valuable context about the nature, state, and attributes of each file, enabling more informed decisions and automated actions in file management systems.

Each file or folder returned by the endpoint includes a structured set of fields such as its name, type, size, creation/modification timestamps, access status, and more.

Additionally, each item is associated with a specific site. This ensures that files are clearly scoped to their respective locations, enabling secure, site-aware management and control.

Request Parameters

Parameter	Type	Required	Description
`path`	string	Yes	The target directory path to browse. (e.g., a path or virtual root).
`site`	string	Yes	The name of the site.
`page`	integer	No	Page number for paginated results. Defaults to 1.
`page_size`	integer	No	Number of items per page (20–100). Use 0 to disable pagination.
`details`	boolean	No	If true, returns detailed metadata for child files/folders. Defaults to false.
`restricted`	boolean	No	If true, includes restricted (e.g., hidden) objects. Defaults to false.
`cache_ttl`	integer	No	Duration (in seconds) for caching the result.
`type`	string	No	Filter by item type (e.g., `directory`, `file`).

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/file/?
path=mmfs1&site=siteA&details=false&restricted=false' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "url": "http://your-hub-address:8000/api/file/?
path=mmfs1&site=siteA&details=false&restricted=false",
  "site": "siteA",
  "path": "mmfs1",
  "type": "directory",
  "size": 262144,
  "ctime": "2025-05-14T12:01:11.995371Z",
  "mtime": "2025-05-14T12:01:11.995176Z",
  "atime": "2025-06-23T14:13:11.940276Z",
  "children": {
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
      {
        "site": "siteA",
        "path": "mmfs1/data",
        "name": "data",
        "type": "directory",
        "size": 0,
        "deleted": false,
        "restricted": false,
        "status": "online"
      }
    ]
  }
}
```


Response Fields

Field	Type	Description
`url`	string	URL of the current directory view.
`site`	string	Site name the data belongs to.
`path`	string	Current directory path.
`type`	string	Always `"directory"` in this context.
`size`	integer	Size of the directory (in bytes).
`ctime`, `mtime`, `atime`	string	Creation, modification, and access timestamps (ISO format).
`children.count`	integer	Number of child items in this directory.
`children.results[]`	array	List of file or folder objects under this path.

POST /file/workflow/

Description

The `/file/workflow/` API endpoint initiates a specified Workflow Operation on a list of files or directories within a given site.

Workflows can perform automated actions such as migration, deletion, or metadata updates on targeted file paths. This operation supports recursive discovery and accepts additional workflow-specific parameters.

Request Parameters

Parameter	Type	Required	Description
`paths`	array	Yes	List of target file or directory paths to apply the workflow to.
`site`	string	Yes	The **primary site** on which the workflow is executed.
`discovery`	string	Yes	How to discover files: `"recursive"` or `"flat"` (non-recursive).
`workflow`	string	Yes	Name of the workflow to execute (e.g., `migrate`, `delete`, `tag`).
`fields`	object	No	Optional workflow-specific fields. Example: `{ "lock_level": "implicit" }`

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/file/workflow/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
  "paths": [{"path": "/mmfs1/data/"}],
  "site": "siteA",
  "discovery": "recursive",
  "workflow": "migrate",
  "fields": {
    "lock_level": "implicit"
  }
}'
```

Note: In this example, the `/file/workflow/` API is used to trigger a migrate workflow on the directory `/mmfs1/data/` within the given site.

The `"discovery"`: `"recursive"` parameter means the workflow applies to the specified directory and all of its contents, including files and nested subdirectories.

The `"fields"` section sets `"lock_level"`: `"implicit"`, which controls how files are locked or reserved during the migration process.

The response confirms that a migration job has started, providing a job ID, status (STARTED), and a list of task IDs representing the operations in progress.

This kind of workflow is useful for moving data across storage systems, preparing content for processing, or archiving large data sets within a site managed by the Hub.

Successful Response

HTTP Status Code: `201 Created`

Content-Type: `application/json`

Example Response


```
{
  "url": "http://your-hub-address:8000/api/jobs/38/",
  "id": 38,
  "workflow": "migrate",
  "fields": {
    "lock_level": "implicit"
  },
  "created": "2025-06-24T13:17:09.771857Z",
  "started": "2025-06-24T13:17:09.818646Z",
  "completed": null,
  "runtime": 0.088984,
  "state": "STARTED",
  "site": "siteA",
  "queue": "default",
  "owner": {
    "type": "user",
    "id": 2,
    "name": "hubadmin"
  },
  "discovery": "recursive",
  "is_settings_job": false,
  "friendly_name": "Dehydrate of '/mmfs1/data/' on 'siteA'",
  "progress": null,
  "tasks": [266, 265, 267]
}
```


Response Fields

Field	Type	Description
`id`	integer	Job ID representing the submitted workflow operation.
`url`	string	API link to check job status and details.
`workflow`	string	Name of the executed workflow.
`fields`	object	Workflow-specific parameters used in this execution.
`created`	datetime	Timestamp when the job was created.
`started`	datetime	Timestamp when execution started.
`completed`	datetime	Timestamp when execution completed (`null` if still running).
`runtime`	float	Time (in seconds) taken so far.
`state`	string	Current job state: `STARTED`, `COMPLETED`, `FAILED`, etc.

Field	Type	Description
`site`	string	Site where the workflow was executed.
`queue`	string	Job queue used, usually `"default"`.
`owner`	object	User who triggered the job (includes `id`, `name`, `type`).
`discovery`	string	Discovery mode used (`recursive` or `flat`).
`friendly_name`	string	Human-readable job description.
`progress`	object/ null	Progress tracking info (`null` if not yet tracked).
`tasks`	array	List of task IDs spawned by this job.

Filesystems

GET /filesystems/

Description

The `/filesystems/` API endpoint retrieves a list of **available file systems** across all configured sites. Each file system is associated with a specific site and includes basic metadata such as its name and mount point.

This endpoint is particularly useful, where each site may host its own instance of a distributed file system (e.g., GPFS/Spectrum Scale), and operations need to be scoped accordingly.

> **Note:** A filesystem is a structured method of storing, organizing, and accessing data on a storage device.
>
> In multi-site platforms like Ngenea Hub, a filesystem refers to a mounted storage volume, such as mmfs1 that exists within a specific site. Each filesystem is uniquely identified by its name and mount point, and it serves as the root for all file and directory operations within that site.
>
> This logical storage environment serves as the root for files, directories, and filesets within the associated site.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/filesystems/?
```

```
page=1&page_size=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "name": "mmfs1",
      "site": "siteA",
      "mountpoint": "/mmfs1"
    },
    {
      "name": "mmfs1",
      "site": "siteB",
      "mountpoint": "/mmfs1"
    }
  ]
}
```


Response Fields

Top-Level Fields

Field	Type	Description
`count`	integer	Total number of file systems across all pages.
`next`	string	URL to the next page of results (if any).
`previous`	string	URL to the previous page of results (if any).
`results`	array	List of file systems with their metadata.

Each item in the `results` array contains:

Field	Type	Description
`name`	string	Name of the file system (e.g., `mmfs1`).
`site`	string	Name of the site where this file system is mounted.
`mountpoint`	string	Mount path of the file system on that site.

GET /filesystems/{id}/

Description

The `GET /filesystems/{id}/` endpoint retrieves detailed information about a **specific filesystem space** by its unique ID. This is useful for querying metadata about a filesystem space associated with a particular site—such as its name and mount location.

This endpoint is typically used when you already know the filesystem's ID (from a `/filesystems/` list call) and want to fetch more focused data for display or internal logic.

Request Parameters

Parameter	Type	Required	Description
`id`	string	Yes	The unique identifier of the filesystem.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/filesystems/1/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "name": "mmfs1",
  "site": "siteA",
  "mountpoint": "/mmfs1"
}
```

Response Fields

Please check the above endpoint to understand the response field.

Filesets

A fileset is a logical subdivision of a filesystem, used to organize, manage, and apply policies to files in a more granular way. *In Ngenea Hub, a fileset is automatically created when a space is created*.

It provides an isolated namespace within a filesystem, allowing for features such as:

- Independent data management (e.g., quota, snapshots)
- Enhanced access control
- Simplified backup and archiving

Note: The term “fileset” comes from its role: a set of files grouped by policy, location, or purpose within a larger filesystem. This abstraction helps manage large-scale data more efficiently.

GET /filesets/

Description

This API endpoint allows users to retrieve all filesets for a specific site.

Request Parameters

Name	Type	Location	Required	Description
`site`	string	query	Yes	Name of the site (minimum 1 character) where file sets are defined.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/filesets/?site=siteA' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
[
  {
    "path": "/mmfs1/data/Test_Space",
    "site": "qatest-pdevan-testplugin-siteA",
    "name": "sas1-Test_Space",
    "label": "sas1-Test_Space",
    "filesystem": "mmfs1"
  }
]
```

Response Fields

Field	Type	Description
`path`	string	Full path to the fileset within the filesystem.
`site`	string	The name of the site the fileset belongs to.
`name`	string	The internal name of the fileset.
`label`	string	The display label of the fileset.
`filesystem`	string	The parent filesystem this fileset resides in.

Sites

In Ngenea Hub, a site represents a distinct location or system within the Ngenea Hub ecosystem where data and events are managed and synchronized. Each site is identified by a unique name, which is crucial for tracking, managing, and reflecting events (such as file updates or deletions) across the distributed environment.

GET /sites/

Description

The `GET /sites/` endpoint allows you to retrieve a list of all sites registered in your Ngenea Hub system. Each site represents a physical or cloud-based server managed by the Hub. The endpoint supports pagination and provides detailed metadata about each site, such as its name, type, associated storage spaces, and file statistics.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/sites/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "url": "http://your-hub-address:8000/api/sites/1/",
      "id": 1,
      "name": "siteA",
      "shortcode": "CXL",
      "label": "siteA",
```

```
      "am_i_configured": true,
      "bandwidth": 0,
      "spaces": ["mmfs1", "Test_Space"],
      "type": "CLOUD",
      "directories_count": 317,
      "files_count": 977,
      "files_hydrated_count": 977,
      "files_stubbed_count": 0,
      "editable": true,
      "public_url": "https://your-hub-address",
      "color": [170, 232, 242],
      "ready_to_configure": true
    },
    {
      "url": "http://your-hub-address:8000/api/sites/2/",
      "id": 2,
      "name": "siteB",
      "shortcode": "I17",
      "label": "siteB",
      "am_i_configured": true,
      "bandwidth": 0,
      "spaces": ["mmfs1"],
      "type": "CLOUD",
      "directories_count": 317,
      "files_count": 1001,
      "files_hydrated_count": 1001,
      "files_stubbed_count": 0,
      "editable": true,
      "public_url": "https://10.201.2.185",
      "color": [227, 170, 242],
      "ready_to_configure": true
    }
  ]
}
```


Response Fields

Top-Level Fields

Field	Type	Description
`count`	integer	Total number of sites available.
`next`	string/null	URL to the next page of results, or `null` if none.
`previous`	string/null	URL to the previous page of results, or `null` if none.
`results`	array	List of site objects returned for the current page.

`results[]` Object Fields

Field	Type	Description
`url`	string	API URL for the specific site resource.
`id`	integer	Unique identifier of the site.
`name`	string	Name of the site.
`shortcode`	string	Short code representing the site.

Field	Type	Description
`label`	string	Label for the site.
`am_i_configured`	boolean	Indicates if the site is configured.
`bandwidth`	integer	Bandwidth limit for the site.
`spaces`	array of strings	List of associated spaces for the site.
`type`	string	Type of the site (e.g., `"CLOUD"`).
`directories_count`	integer	Number of directories in the site.
`files_count`	integer	Total number of files in the site.
`files_hydrated_count`	integer	Number of hydrated files.
`files_stubbed_count`	integer	Number of stubbed files.
`editable`	boolean	Whether the site is editable.
`public_url`	string	Public URL to access the site.
`color`	array of integers	RGB color values representing the site color.
`ready_to_configure`	boolean	Indicates if the site is ready to be configured.

POST /sites/

Description

Creates a new site with the specified properties, including spaces, name, configuration parameters, and metadata.

Request Parameters

Name	Type	Location	Description
`data`	object	body	JSON object containing the site details and configuration settings.

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/sites/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
    "spaces": ["mmfs1"],
    "name": "testexample",
    "shortcode": "str",
    "file_batch_size": 40,
    "file_batch_gb": 1,
    "enable_auto_file_batch_sizing": true,
    "lock_threshold": 86400,
    "bandwidth": 0,
    "elasticsearch_url": "localhost:9200",
    "pixstor_search_url": "https://localhost",
    "public_url": "https://pixstor-mn-001",
    "include": [],
```

```
    "exclude": [],
    "label": "string",
    "am_i_configured": true,
    "type": "CLOUD",
    "color": [242, 170, 235],
    "files_count": 2147483647,
    "directories_count": 2147483647,
    "files_stubbed_count": 2147483647,
    "files_hydrated_count": 2147483647,
    "gpfs_iscan_buckets": 2147483647,
    "gpfs_iscan_threads": 2147483647,
    "ready_to_configure": true
  }'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "id": 3,
  "url": "http://10.201.2.224:8000/api/sites/3/",
  "name": "testexample",
  "shortcode": "str",
  "label": "string",
  "spaces": ["mmfs1"],
  "file_batch_size": 40,
  "file_batch_gb": 1,
  "enable_auto_file_batch_sizing": true,
  "lock_threshold": 86400,
  "bandwidth": 0,
  "elasticsearch_url": "localhost:9200",
  "pixstor_search_url": "https://localhost",
  "public_url": "https://pixstor-mn-001",
  "include": [],
  "exclude": [],
  "am_i_configured": true,
  "type": "CLOUD",
  "color": [242, 170, 235],
  "files_count": 2147483647,
  "directories_count": 2147483647,
  "files_stubbed_count": 2147483647,
  "files_hydrated_count": 2147483647,
  "gpfs_iscan_buckets": 2147483647,
  "gpfs_iscan_threads": 2147483647,
  "ready_to_configure": true,
  "editable": true
}
```


Response Fields

Refer to the requests body fields under `GET/SITES`.

GET /sites/{id}/

Description

Retrieves detailed information about a specific site by its unique ID.

Request Parameters

Name	Type	Location	Description
`id`	string	path	The unique ID of the site.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/sites/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "url": "http://your-hub-address:8000/api/sites/2/",
  "spaces": [
    {
      "url": "http://your-hub-address:8000/api/spaces/1/",
      "id": 1,
      "name": "mmfs1"
    }
  ],
  "editable": true,
  "id": 2,
  "name": "siteB",
  "shortcode": "I17",
  "file_batch_size": 40,
  "file_batch_gb": 1,
  "enable_auto_file_batch_sizing": true,
  "lock_threshold": 86400,
  "bandwidth": 0,
  "elasticsearch_url": "localhost:19200",
  "pixstor_search_url": null,
  "public_url": "https://10.201.2.185",
  "include": null,
  "exclude": null,
  "label": "siteB",
  "am_i_configured": true,
```

```
  "type": "CLOUD",
  "color": [227, 170, 242],
  "files_count": 1001,
  "directories_count": 317,
  "files_stubbed_count": 0,
  "files_hydrated_count": 1001,
  "gpfs_iscan_buckets": null,
  "gpfs_iscan_threads": null,
  "ready_to_configure": true
}
```

Response Fields

Refer to the requests body fields under `GET/SITES`.

PATCH /sites/{id}/

Description

Updates an existing site's properties using partial or full data. Only the fields included in the request will be updated.

Request Parameters

Name	Type	Location	Description
`id`	string	path	The unique ID of the site to update.
`data`	object	body	JSON object containing the fields to update.

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/sites/3/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
    "spaces": ["mmfs1"],
    "name": "testexample",
    "shortcode": "Te",
    "label": "string"
  }'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "url": "http://your-hub-address:8000/api/sites/3/",
  "spaces": [
```

```
    {
      "url": "http://your-hub-address:8000/api/spaces/1/",
      "id": 1,
      "name": "mmfs1"
    }
  ],
  "editable": true,
  "id": 3,
  "name": "testexample",
  "shortcode": "Te",
  "label": "string",
  "type": "CLOUD",
  "ready_to_configure": true
}
```

Note: The site with ID `3` was successfully updated with example values (e.g., shortcode: “Te”, label: “TestEx”). Any subset of parameters can be provided in the PATCH request to update specific fields as per user requirements.

Response Fields

The response includes all updated site fields. These are the same as those returned in **GET /sites/{id}/**.

DELETE /sites/{id}/

Description

Deletes a specific site identified by its unique ID.

Request Parameters

Name	Type	Location	Description
`id`	string	path	The unique ID of the site.

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/sites/4/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `204 No Content` — No response body returned.

Note: Site with ID 4 was deleted. This operation is irreversible. Ensure the correct ID is passed, as all associated metadata and configuration will be permanently removed.

GET /sites/{id}/health/

Description Retrieves the health status of a specific site, including the health of its nodes and queues.

Health represents the overall operational status of the site and its components, indicating whether they are functioning correctly, online, and responsive. A status of "ok" means the site and its nodes and queues are running normally without issues.

Request Parameters

Name	Type	Location	Description
`id`	string	path	The unique ID of the site.

Example Request

```
curl -X 'GET' \
  'http://10.201.2.224:8000/api/sites/2/health/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key
xxxxxxxx.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "site": "qatest-pdevan-testplugin-siteB",
  "health": "ok",
  "nodes": [
    {
      "name": "qatest-pdevan-testplugin-siteB-1",
      "health": "ok",
      "online": true,
      "has_default_queue": true,
      "last_heartbeat": "2025-07-03T15:34:59.540149+00:00",
      "worker_version": "2.6.0",
      "ngenea_version": "1.31.0",
      "analytics_version": "2.7.1",
      "search_version": "1.2.0"
    }
  ],
  "queues": [
    {
      "name": "default",
      "health": "ok",
      "online": true,
      "last_heartbeat": "2025-07-03T15:35:09.552068+00:00"
    }
```

```
  ]
}
```

Response Fields

Site Health Response Fields

Field	Type	Description
`site`	string	Name of the site.
`health`	string	Overall health status of the site (e.g., `ok`).
`nodes`	array	List of node health objects.

`nodes[]` Object Fields

Field	Type	Description
`nodes[].name`	string	Name of the node.
`nodes[].health`	string	Health status of the node.
`nodes[].online`	boolean	Whether the node is online.
`nodes[].has_default_queue`	boolean	Whether the node has a default queue configured.
`nodes[].last_heartbeat`	string	Timestamp of the last heartbeat received from the node.
`nodes[].worker_version`	string	Version of the worker component (if available).
`nodes[].ngenea_version`	string	Version of the ngenea component (if available).
`nodes[].analytics_version`	string	Version of the analytics module (if available).
`nodes[].search_version`	string	Version of the search module (if available).
`nodes[].version_details`	string	Version fetch status message (if any errors).

`queues[]` Object Fields

Field	Type	Description
`queues[].name`	string	Queue name (e.g., `default`).
`queues[].health`	string	Health status of the queue.
`queues[].online`	boolean	Whether the queue is online.
`queues[].last_heartbeat`	string	Timestamp of the last heartbeat from the queue.

POST /sites/{id}/refresh/

Description

Triggers a refresh operation for the specified site, which may include updating analytics, fetching remote servers, loading storage pools, and checking space/site quotas asynchronously.

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/sites/2/refresh/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -d ''
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`


```
{
  "dynamocore.tasks.site.fetch_site_analytics": "4fb5b60a-630e-4adb-b946-2c555184f0a4",
  "dynamocore.tasks.server.get_remote_servers": "900d2b0e-1c78-422e-82c7-c6ac84a9693c",
  "dynamocore.tasks.admin.load_storagepools": "7edadbc1-c400-4d19-b0b5-6af348085e5c",
  "dynamocore.tasks.space_site.send_check_space_site_quota": "34d94c8b-d4e4-4357-bf65-d589f920dbd6"
}
```


Response Fields

Field	Type	Description
Task names as keys	string	Identifiers for the triggered asynchronous tasks.
Task IDs as values	string	UUIDs representing each task instance.

GET /sites/{id}/settings/

Description

This API endpoint retrieves the configuration settings for a specific site identified by its id. These settings control various operational parameters such as backup schedules, logging, monitoring, network configurations, and other system behaviors related to the site.

Accessing site settings is essential for administrators or automated systems to review, audit, or adjust the configurations without direct access to the server. It allows monitoring of the current configuration state and supports troubleshooting, compliance checks, and dynamic adjustments to the site behavior.

Request Parameters

Name	Type	Location	Description
`id`	string	path	The unique identifier of the site.
`flat`	boolean	query	Return settings as a flat dictionary (`true` or `false`, default: `false`).

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/sites/2/settings/?flat=false'
\
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
[
  {
    "id": 32,
    "key": "apbackup:schedule",
    "value": "*-*-* 01:00:00",
    "desc": "The systemd calendar event for configuring the
systemd timer service that runs apbackup."
  },
  {
    "id": 36,
    "key": "email:arcapix_alert_address",
    "value": "alerts@pixitmedia.com",
    "desc": "Description not specified"
  },
  {
    "id": 37,
    "key": "monitoring:auth:monitoring_internal",
    "value": "**********",
    "desc": "Description not specified"
  },
  {
    "id": 42,
    "key": "ntp:servers",
    "value": [
      "0.centos.pool.ntp.org",
      "1.centos.pool.ntp.org",
      "2.centos.pool.ntp.org",
      "3.centos.pool.ntp.org"
    ],
    "desc": "list of NTP servers"
  }
  // more settings ...
]
```


Response Fields

Field	Description
`id`	Setting unique identifier.
`key`	The setting key name.
`value`	The setting value (may be string, bool, array, etc.).
`desc`	Description of the setting (may be `null` or unspecified).

POST /sites/{id}/settings/refresh/

Description

The `POST /sites/{id}/settings/refresh/` endpoint is used to refresh the settings for a specific site identified by id. This action can be used to update existing settings or create settings in the Ngenea Hub if none exist for the given site. It's an asynchronous process, meaning the operation will be carried out in the background, and you can track its progress via a task ID.

Request Parameters

Name	Type	Location	Description
`id`	string	path	**Required.** Unique identifier of the site to refresh settings for. This is part of the URL path. Example: `/sites/2/settings/refresh/`

Example Request

```
curl -X 'POST' \
  'http://10.201.2.224:8000/api/sites/2/settings/refresh/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -d ''
```

Successful Response

HTTP Status Code: `202 Accepted`

This indicates that the request has been accepted and is now being processed, but the operation has not completed yet. The task is running asynchronously, so you can track its progress with the task_id.

Content-Type: `application/json`

Example Response

```
{
  "task_id": "201ef094-5e55-42b7-818b-1e586c52bda0"
}
```

Response Fields

Field	Type	Description
`task_id`	string	A unique identifier for the refresh task. Use this ID to track its progress or retrieve status.

Shares

Shares are designated directories or access points within the Pixstor storage system that are made available to external systems over a network. These *shares allow remote clients to access, read, write, or manage data stored in the Pixstor environment* as if it were part of their local file system.

These shares can be accessed using various protocols like **NFS** (for Linux/Unix-based clients) and **Samba** (for Windows and other SMB-compatible clients).

NFS and Samba are two common protocols for sharing files, with NFS being primarily used for Linux-based environments and Samba providing cross-platform compatibility for Windows and Linux systems.

GET /shares/nfs/

The `GET /shares/nfs/` endpoint is used to retrieve a list of **NFS shares** from your storage system. NFS (Network File System) shares allow remote clients to access the file system over a network. This API call provides details of all NFS shares currently configured, including their associated settings, clients, and more.

Request Parameters

No parameters - This endpoint does not require any parameters in the request query. The response will return a list of all NFS shares.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/shares/nfs/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
[
  {
    "id": 1,
    "uuid": "a4255625-e9d9-4a72-817a-719cfd69d73e",
    "space": 2,
    "clients": [
      {
```

```
        "id": 1,
        "hosts": "*",
        "force_root_user_id": false,
        "force_all_user_id": false,
        "asynchronous": false,
        "write_delay": true,
        "insecure_ports": false,
        "subtree_check": true,
        "advanced_settings": null,
        "read_only": false,
        "anonymous_user_id": 65534,
        "anonymous_group_id": 65534,
        "uuid": "4edbd055-7ec6-4a3f-9b80-cde5d6f0d55d",
        "nfs_share": 1
      }
    ],
    "path": "/mmfs1/data/Test_Space",
    "name": "test2",
    "fsid": 55102
  },
  {
    "id": 3,
    "uuid": "67aedd4b-60ee-462d-8673-0d8f662fae7b",
    "space": 1,
    "clients": [
      {
        "id": 2,
        "hosts": "string",
        "force_root_user_id": null,
        "force_all_user_id": true,
        "asynchronous": true,
        "write_delay": true,
        "insecure_ports": true,
        "subtree_check": true,
        "advanced_settings": {},
        "read_only": true,
        "anonymous_user_id": 2147483647,
        "anonymous_group_id": 2147483647,
        "uuid": "28a181ef-2c56-4e03-b64d-49f5082fda4a",
        "nfs_share": 3
      }
    ],
    "path": "/mmfs1/data/API_Example",
    "name": "test5",
    "fsid": 65535
  }
]
```


Response Fields

Field	Description
`id`	The unique identifier for the NFS share.
`uuid`	A unique identifier for the NFS share.

Field	Description
`space`	The storage pool associated with this NFS share. Corresponds to the space identifier.
`clients`	A list of client configurations (who can access the share). Each includes access settings.
`path`	The file system path for the NFS share (e.g., `/mmfs1/data/Test_Space`).
`name`	The name of the NFS share.
`fsid`	The file system identifier (FSID) representing the file system for this share.

POST /shares/nfs/

Description

This request is used to create a new **NFS** share. The share can be accessed remotely by clients that support the NFS protocol.

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/shares/nfs/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
    "path": "/mmfs1",
    "name": "Test1",
    "space": 1,
    "clients": [
      {
        "hosts": "*",
        "read_only": false
      }
    ]
  }'
```

Request Body Fields

NFS Share Configuration Fields

Field	Type	Description
`path`	string	Directory path of the NFS share (e.g., `/mmfs1`).
`name`	string	Name of the NFS share (e.g., `"Test1"`).
`space`	integer	Space ID where the share is created (system managed).
`clients[]`	object	List of client configurations defining access permissions.
`hosts`	string	Hosts that can access the share (e.g., `*` for all hosts).
`read_only`	boolean	If `true`, makes the share read-only for clients (default: `false`) .

Optional Fields

Field	Type	Description
`force_root_user_id`	boolean	If `true`, map requests from uid/gid 0 to the anonymous uid/gid.
`force_all_user_id`	boolean	If `true`, maps all users to a specific user ID when accessing the share.
`asynchronous`	boolean	If `true`, enables async writes (faster but risks data loss on failure).
`write_delay`	boolean	If `true`, enables write delay to enhance performance.
`insecure_ports`	boolean	If `true`, allows access from insecure (non-privileged) ports.
`subtree_check`	boolean	If `true`, enables subtree integrity checks when moving/renaming files.
`advanced_settings`	object	Placeholder for advanced or custom configurations.
`anonymous_user_id`	integer	User ID for anonymous access (e.g., `65534` for “nobody”).
`anonymous_group_id`	integer	Group ID for anonymous access (e.g., `65534` for “nobody”).
`uuid`	string	UUID of the client configuration (system generated).
`nfs_share`	integer	ID of the NFS share associated with this client config (auto-assigned).

Successful Response

HTTP Status Code: `201 CREATED`

Content-Type: `application/json`

Example Response


```
{
  "id": 5,
  "uuid": "3eae699e-c148-4179-ac36-b94a4d5f4dc0",
  "space": 1,
  "clients": [
    {
      "id": 4,
      "hosts": "*",
      "force_root_user_id": null,
      "force_all_user_id": null,
      "asynchronous": null,
      "write_delay": null,
      "insecure_ports": null,
      "subtree_check": null,
      "advanced_settings": null,
      "read_only": false,
      "anonymous_user_id": null,
      "anonymous_group_id": null,
```

```
      "uuid": "948bed6b-a87c-4b2d-8d38-3fd88ee5599a",
      "nfs_share": 5
    }
  ],
  "path": "/mmfs1",
  "name": "Test1",
  "fsid": 19904
}
```

Response Fields

Field	Type	Description
`id`	integer	The unique identifier for the newly created NFS share.
`uuid`	string	The UUID of the NFS share.
`space`	integer	The space (ID) where the share was created.
`clients`	array	List of client settings for the share, including host and permission info.
`clients[]`	object	Client-specific settings. Each includes `id`, `hosts`, `read_only`, etc.
`path`	string	The path of the directory being shared (e.g., `/mmfs1`).
`name`	string	The name of the share (e.g., `"Test1"`).
`fsid`	integer	The filesystem ID that represents the share's filesystem.

GET /shares/nfs/{id}/

Description

This endpoint retrieves detailed information about a specific NFS share, identified by its *id*. The data returned includes the NFS share's properties such as the share's path, name, associated space, and client configurations.

Request Parameters

Name	Description
`id`	**Required.** The unique integer identifier of the NFS share. Used as a path parameter in API requests.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/shares/nfs/1/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "id": 1,
  "uuid": "a4255625-e9d9-4a72-817a-719cfd69d73e",
  "space": 2,
  "clients": [
    {
      "id": 1,
      "hosts": "*",
      "force_root_user_id": false,
      "force_all_user_id": false,
      "asynchronous": false,
      "write_delay": true,
      "insecure_ports": false,
      "subtree_check": true,
      "advanced_settings": null,
      "read_only": false,
      "anonymous_user_id": 65534,
      "anonymous_group_id": 65534,
      "uuid": "4edbd055-7ec6-4a3f-9b80-cde5d6f0d55d",
      "nfs_share": 1
    }
  ],
  "path": "/mmfs1/data/Test_Space",
  "name": "test2",
  "fsid": 55102
}
```


Response Fields

Refer to the response fields under `GET/Shares/NFS/`.

GET /shares/samba/

Description

This endpoint retrieves a **list of all Samba shares** available in the system, providing details like share name, access settings, and visibility.

Request Parameters : No parameters required for this request.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/shares/samba/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
[
  {
    "id": 1,
    "path": "/mmfs1",
    "uuid": "1406dc83-124c-4028-8b9d-8a0ef8c3b375",
    "hosts_allow": [],
    "hosts_deny": [],
    "admin_users": [],
    "allowed_users": [],
    "name": "Test1",
    "read_only": false,
    "visible": true,
    "create_mask": null,
    "directory_mask": null,
    "advanced_settings": {},
    "multi_thread_reads": true,
    "multi_thread_writes": true,
    "root_share_locking": true,
    "hsm_support": true,
    "guest_ok": false,
    "force_create_mode": null,
    "force_directory_mode": null,
    "space": 1
  },
  ////////////////////////////////// - Sites info
]
```

Response Fields

Field	Description
`id`	A unique identifier for the samba share.
`path`	The path to the shared directory in the file system (e.g., `/mmfs1`).
`uuid`	A universally unique identifier (UUID) for the samba share.
`hosts_allow`	List of allowed client hosts for accessing the share. Empty if no specific hosts are allowed.
`hosts_deny`	List of denied client hosts for accessing the share. Empty if no specific hosts are denied.
`admin_users`	List of users with administrative access to the share.
`allowed_users`	List of users permitted to access the share.
`name`	The name of the samba share (e.g., `Test1`).
`read_only`	Boolean indicating if the share is read-only. `true` means read-only; users can only view or read data but cannot modify it.
`visible`	Boolean indicating if the share is visible to clients.
	- `true`: Clients can see and browse the share.
	- `false`: Clients cannot see the share during normal browsing, but access is still possible if the path is known.

Field	Description
`create_mask`	Permissions mask for creating new files in the share.
`directory_mask`	Permissions mask for directories in the share.
`advanced_settings`	Advanced settings for the samba share (if any). Can be an empty object if no advanced settings exist.
`multi_thread_reads`	Boolean indicating whether multi-threaded reads are enabled.
`multi_thread_writes`	Boolean indicating whether multi-threaded writes are enabled.
`root_share_locking`	Boolean indicating if root share locking is enabled.
`hsm_support`	Boolean indicating if HSM (Hierarchical Storage Management) is supported.
`guest_ok`	Boolean indicating if guest access is allowed to the share.
`force_create_mode`	File creation mode enforced on the share, if any.
`force_directory_mode`	Directory creation mode enforced on the share, if any.
`space`	The storage space identifier associated with the share.

POST /shares/samba/

Description

The **POST /shares/samba/** endpoint is used to create a new Samba share on the server. You must send a request with details like the share's name, path, and visibility settings.

Additionally, you can specify which clients are allowed access to the share, along with specific permissions (read-only or full access).

Example Request

```
curl -X POST \
  http://your-hub-address:8000/api/shares/samba/ \
  -H "accept: application/json" \
  -H "Authorization: Api-Key your-api-key" \
  -H "Content-Type: application/json" \
  -d '{
    "path": "/mmfs1",
    "name": "Test9",
    "space": 1,
    "read_only": true,
    "visible": true,
    "clients": [
      {
        "hosts": "*",
        "read_only": false
      }
    ]
  }'
```

Request Body Fields

Field	Description	Example
`path`	The directory path to be shared via Samba.	`/mmfs1`
`name`	The name of the Samba share.	`Test9`
`space`	The storage space ID that this share will be linked to.	`1`
`read_only`	If set to `true`, the share is read-only for all users.	`true`
`visible`	If set to `true`, the share is visible to clients.	`true`
`clients`	A list of client configurations specifying which clients can access the share.	`[{"hosts": "*", "read_only": false}]`

Successful Response

HTTP Status Code: `201 CREATED`

Content-Type: `application/json`

Example Response


```
{
  "id": 2,
  "path": "/mmfs1",
  "uuid": "bb4f28ae-996d-4e68-b973-e9c36238cb43",
  "hosts_allow": [],
  "hosts_deny": [],
  "admin_users": [],
  "allowed_users": [],
  "name": "Test9",
  "read_only": true,
  "visible": true,
  "create_mask": null,
  "directory_mask": null,
  "advanced_settings": {},
  "multi_thread_reads": false,
  "multi_thread_writes": false,
  "root_share_locking": false,
  "hsm_support": false,
  "guest_ok": false,
  "sites": {},
  "force_create_mode": null,
  "force_directory_mode": null,
  "space": 1
}
```


Response Fields

Refer to the response fields under `GET`**/Shares/Samba**.

GET /shares/samba/{id}/

Description

This API endpoint retrieves the details of a specific Samba share based on its unique ID. It returns metadata, configuration settings, and site-specific share information.

Request Parameters

Name	Type	In	Required	Description
`id`	integer	path	Yes	A unique integer value identifying the Samba share.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/shares/samba/3/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "id": 3,
  "path": "/mmfs1",
  "uuid": "f01ab517-6b70-4bd8-8d60-9f7e9b3a7a58",
  "hosts_allow": {},
  "hosts_deny": {},
  "admin_users": {},
  "allowed_users": {},
  "name": "Samba Test",
  "read_only": true,
  "visible": true,
  "create_mask": "null",
  "directory_mask": "null",
  "advanced_settings": {},
  "multi_thread_reads": true,
  "multi_thread_writes": true,
  "root_share_locking": true,
  "hsm_support": true,
  "guest_ok": true,
  "sites": {
    "siteA": {
      "name": "Samba Test",
      "path": "/mmfs1",
      "present": true,
      "visible": true,
      "guest_ok": true,
      "read_only": true,
      "hosts_deny": {},
      "admin_users": {},
```

```
      "create_mask": "null",
      "hosts_allow": {},
      "hsm_support": true,
      "allowed_users": {},
      "directory_mask": "null",
      "advanced_settings": {},
      "multi_thread_reads": true,
      "root_share_locking": true,
      "multi_thread_writes": true
    },
    "siteB": {
      "name": "Samba Test",
      "path": "/mmfs1",
      "present": true,
      "visible": true,
      "guest_ok": true,
      "read_only": true,
      "hosts_deny": {},
      "admin_users": {},
      "create_mask": "null",
      "hosts_allow": {},
      "hsm_support": true,
      "allowed_users": {},
      "directory_mask": "null",
      "advanced_settings": {},
      "multi_thread_reads": true,
      "root_share_locking": true,
      "multi_thread_writes": true
    }
  },
  "force_create_mode": null,
  "force_directory_mode": null,
  "space": 1
}
```


Response Fields

Field	Type	Description
`id`	integer	Unique ID of the Samba share
`path`	string	Path of the share on the filesystem
`uuid`	string	Unique identifier (UUID) for the Samba share
`hosts_allow`	object	List of allowed hosts (empty if unrestricted)
`hosts_deny`	object	List of denied hosts (empty if unrestricted)
`admin_users`	object	Admin users for the share (empty if none)
`allowed_users`	object	Allowed users for the share (empty if unrestricted)
`name`	string	Human-readable name of the share
`read_only`	boolean	Indicates if the share is read-only
`visible`	boolean	Indicates if the share is visible
`create_mask`	string	File permission mask on file creation (often null or a string like `"0777"`)

Field	Type	Description
`directory_mask`	string	Directory permission mask on directory creation
`advanced_settings`	object	Advanced Samba configuration settings (if any)
`multi_thread_reads`	boolean	Whether multi-threaded reads are enabled
`multi_thread_writes`	boolean	Whether multi-threaded writes are enabled
`root_share_locking`	boolean	Indicates if root share locking is enabled
`hsm_support`	boolean	Whether HSM (Hierarchical Storage Management) support is enabled
`guest_ok`	boolean	Whether guest access is allowed
`sites`	object	Site-specific configuration and presence info (keyed by site name)
`force_create_mode`	string	Optional enforced file create mode (null if not set)
`force_directory_mode`	string	Optional enforced directory create mode (null if not set)
`space`	integer	Logical identifier or grouping reference (may represent tenant/project)

PATCH /shares/samba/{id}/

Description

This endpoint allows partial updates to an existing Samba share identified by its `id`. Only specific fields are allowed to be updated as listed below. Any attempt to update other fields may result in validation errors or be ignored by the backend.

Request Parameters

Name	Type	In	Required	Description
`id`	integer	path	Yes	A unique integer value identifying the Samba share.

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/shares/samba/3/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
  "path": "/mmfs1",
  "name": "Samba Test Updated",
  "read_only": true,
  "visible": true,
  "advanced_settings": {},
  "multi_thread_reads": true,
  "multi_thread_writes": true,
  "root_share_locking": true,
  "hsm_support": true,
  "guest_ok": true,
```

```
  "sites": {
    "siteA": {
      "name": "Samba Test Updated",
      "path": "/mmfs1",
      "present": true,
      "visible": true,
      "guest_ok": true,
      "read_only": true,
      "hosts_deny": {},
      "admin_users": {},
      "create_mask": "null",
      "hosts_allow": {},
      "hsm_support": true,
      "allowed_users": {},
      "directory_mask": "null",
      "advanced_settings": {},
      "multi_thread_reads": true,
      "root_share_locking": true,
      "multi_thread_writes": true
    }
  },
  "space": 1
}
'
```

Request Body Fields

Field	Type	Description
`hosts_allow`	object	Hosts allowed to access the share
`hosts_deny`	object	Hosts denied from accessing the share
`admin_users`	object	Admin users with elevated privileges
`allowed_users`	object	Regular users allowed to access the share
`name`	string	Display name of the Samba share
`read_only`	boolean	If `true`, the share is read-only
`visible`	boolean	If `true`, the share is visible to users
`create_mask`	string	File creation permission mask (e.g., `"0755"`)
`directory_mask`	string	Directory creation permission mask (e.g., `"0755"`)
`advanced_settings`	object	Additional advanced Samba settings
`multi_thread_reads`	boolean	Enable multi-threaded read operations
`multi_thread_writes`	boolean	Enable multi-threaded write operations
`root_share_locking`	boolean	Enable locking at the root share level
`hsm_support`	boolean	Enable Hierarchical Storage Management (HSM) support
`guest_ok`	boolean	If `true`, guest access is allowed
`force_create_mode`	string	Enforced file create mode (e.g., `"0755"`)
`force_directory_mode`	string	Enforced directory create mode (e.g., `"0755"`)

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "id": 3,
  "path": "/mmfs1",
  "uuid": "f01ab517-6b70-4bd8-8d60-9f7e9b3a7a58",
  "hosts_allow": {},
  "hosts_deny": {},
  "admin_users": {},
  "allowed_users": {},
  "name": "Samba Test Updated",
  "read_only": true,
  "visible": true,
  "create_mask": "null",
  "directory_mask": "null",
  "advanced_settings": {},
  "multi_thread_reads": true,
  "multi_thread_writes": true,
  "root_share_locking": true,
  "hsm_support": true,
  "guest_ok": true,
  "on_sites": {
    "siteA": {
      "name": "Samba Test Updated",
      "path": "/mmfs1",
      "present": true,
      "visible": true,
      "guest_ok": true,
      "read_only": true,
      "hosts_deny": {},
      "admin_users": {},
      "create_mask": "null",
      "hosts_allow": {},
      "hsm_support": true,
      "allowed_users": {},
      "directory_mask": "null",
      "advanced_settings": {},
      "multi_thread_reads": true,
      "root_share_locking": true,
      "multi_thread_writes": true
    }
  },
  "force_create_mode": null,
  "force_directory_mode": null,
  "space": 1
}
```


Response Fields

Same as request fields above.

DELETE /shares/samba/{id}/

Description

This endpoint deletes an existing Samba share identified by its unique id. Once deleted, the Samba share and its configuration will be permanently removed from the system.

Request Parameters

Name	Type	In	Required	Description
`id`	integer	path	Yes	A unique integer value identifying the Samba share.

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/shares/samba/3/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Request Body Fields

This request does **not** require a request body. Only the id parameter in the URL path is needed.

Successful Response

Status Code: `204` **No Content**

Content Type: `application/json`

The Samba share was successfully deleted. No content is returned in the response body.

Spaces

A **Space** represents a logical storage area within the PixStor environment that spans one or more storage systems. Each Space is defined by key attributes such as name, location, size, data protection level, performance profile, and file share accessibility.

Spaces allow for centralized or site-specific access to storage resources and support both global and local views. Access to Spaces is governed by user roles and group permissions, enabling flexible control for administrators, users, and restricted users.

This endpoint allows interaction with Spaces for purposes such as retrieving details, listing available Spaces, or performing administrative actions depending on access level.

Description

The `GET /spaces/` API endpoint retrieves a list of **space** objects from the file browser. Each space represents a storage area (mount point) that may span across multiple sites with metadata such as snapshot schedule, shares, nested shares, and more.

This endpoint supports **pagination** and provides detailed information per space including IDs, names, sizes, site associations, usage stats, snapshot configurations, and relationships.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/spaces/?page=1&page_size=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
    {
      "url": "http://your-hub-address:8000/api/spaces/3/",
      "id": 3,
      "uuid": "6526e45d-5d5e-41c8-8c49-d68efdaab7d5",
      "name": "API_Example",
      "mountpoint": "/mmfs1/data/API_Example",
      "sites": [
        {
          "id": 1,
```

```
          "name": "siteA",
          "label": "siteA",
          "pool": "sas1",
          "usage": [0, 1099511627776]
        }
      ],
      "created": "2025-07-04T14:18:07.227156Z",
      "size": 1099511627776,
      "snapshot_schedule": {
        "id": 2,
        "frequency": "1 00:00:00",
        "time": "00:13:00",
        "duration": "7 00:00:00",
        "space": 3
      },
      "on_sites": {
        "qatest-pdevan-testplugin-siteA": {
          "name": "API_Example",
          "size": 1099511627776,
          "present": true,
          "syncing": true
        }
      },
      "immutable": false,
      "editable": true,
      "color": [111, 64, 191],
      "permission_mode": null,
      "relationships": {
        "shares": [],
        "policies": [],
        "schedules": [
          {
            "id": 3,
            "name": "Test4",
            "discovery": "recursive"
          }
        ]
      },
      "nested_shares": {
        "samba": [],
        "nfs": []
      },
      "is_ready": true
    }
  ]
}
```


Response Fields

Field	Description
`id`	Internal unique ID of the space.
`uuid`	Universally unique identifier.
`name`	Name of the space.

Field	Description
`mountpoint`	Filesystem mount path of the space.
`sites`	List of site objects the space is linked to, each with pool and usage data.
`created`	Creation timestamp of the space.
`size`	Total size in bytes.
`snapshot_schedule`	Optional object defining backup/snapshot timing and frequency.
`on_sites`	Map of site names to presence, size, and sync status of the space.
`immutable`	Boolean. Whether the space can be modified.
`editable`	Boolean. Indicates whether the space settings can be modified. This depends on the permissions of the user making the request.
`color`	RGB color array for display purposes.
`permission_mode`	(Optional) Permission configuration.
`relationships`	Contains related shares, policies, and scan schedules.
`nested_shares`	Samba and NFS shares nested within the space.
`is_ready`	Boolean. Indicates if the space is ready to be used.

POST / spaces/

Description

The `POST /spaces/` endpoint allows you to create a new space in the system. When posting, you define metadata like: Name & mount path, Associated sites, Size, Snapshot schedule (optional), Visual settings like color, Permission handling.

Request Parameters

Name	Location	Type	Required	Description
`data`	body	object	Yes	JSON object representing the space

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/spaces/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "post_example",
    "mountpoint": "/mmfs2",
    "sites": [
      {
        "name": "siteA"
      }
    ],
    "size": 922337203685477,
```

```
    "snapshot_schedule": {
      "frequency": "1 00:00:00",
      "time": "00:00:00",
      "duration": "7 00:00:00"
    },
    "color": [255, 255, 255],
    "permission_mode": "chmodOnly"
  }'
```

Request Body Fields

Field	Type	Required	Description
`name`	string	Yes	Name of the space. Must be unique.
`mountpoint`	string	Yes	Must start with `/`. Filesystem path to mount.
`sites`	list	Yes	List of sites (by name) to associate the space with.
`size`	integer	Yes	Space size in bytes.
`snapshot_schedule`	object	Optional	Defines snapshot automation.
├─ `frequency`	string	Optional	Format like `"1 00:00:00"` (e.g., every 1 day).
├─ `time`	string	Optional	Time of day to run snapshots (e.g., `"00:00:00"`).
├─ `duration`	string	Optional	Duration to retain snapshots (e.g., `"7 00:00:00"`).
`color`	array	Optional	RGB array for UI display (e.g., `[255, 255, 255]`).
`permission_mode`	string	Optional	Permission handling mode (e.g., `"chmodOnly"`).

Successful Response

HTTP Status Code: `201 CREATED`

Content-Type: `application/json`

Example Response


```
{
  "url": "http://your-hub-address:8000/api/spaces/7/",
  "id": 7,
  "uuid": "1d485bb0-4680-4a87-b64b-cb757bc82ccf",
  "name": "post_example",
  "mountpoint": "/mmfs2",
  "sites": [
    {
      "id": 1,
      "name": "siteA",
      "label": "siteA",
      "pool": null,
      "usage": null
```

```
    }
  ],
  "created": "2025-08-11T16:03:18.974929Z",
  "size": 922337203685477,
  "snapshot_schedule": {
    "id": 3,
    "frequency": "1 00:00:00",
    "time": "00:00:00",
    "duration": "7 00:00:00",
    "space": 7
  },
  "on_sites": {},
  "immutable": false,
  "editable": true,
  "color": [255, 255, 255],
  "permission_mode": "chmodOnly",
  "relationships": {
    "shares": [],
    "policies": [],
    "schedules": []
  },
  "nested_shares": {
    "samba": [],
    "nfs": []
  },
  "is_ready": false
}
```


Response Fields

Field	Description
`url`	API URL to access this space.
`id`	Unique ID of the space.
`uuid`	Universally unique ID.
`name`	Name of the created space.
`mountpoint`	Mount path starting with `/`.
`sites`	List of site objects associated with this space.
├─ `id`, `name`, `label`, etc.	Site metadata.
`created`	Timestamp when the space was created.
`size`	Size of the space in bytes.
`snapshot_schedule`	Object showing snapshot config and assigned ID.
├─ `id`, `frequency`, etc.	Snapshot metadata.
`on_sites`	Mapping of space presence/sync on each site (may be empty initially).
`immutable`	Indicates if space is read-only.
`editable`	Indicates if settings can be changed.
`color`	RGB color for UI (e.g., `[255, 255, 255]`).
`permission_mode`	Permission mode applied (e.g., `"chmodOnly"`).

Field	Description
`relationships`	Linked resources: shares, policies, schedules.
`nested_shares`	Nested NFS/Samba shares inside this space.
`is_ready`	Indicates if space is ready for use (typically `false` immediately after creation).

GET /spaces/{id}/

Description

The **GET /spaces/{id}/** endpoint retrieves detailed information about a specific space, identified by its `ID`. It returns metadata about the space, such as: Mount path and name, Associated sites, Size and usage, Snapshot schedule, Synchronization status, Shares, policies, schedules, Readiness and permissions.

Request Parameters

Name	Location	Type	Required	Description
`id`	path	string	Yes	The numeric ID of the space to retrieve (e.g., `3`).

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/spaces/3/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "url": "http://your-hub-address:8000/api/spaces/3/",
  "id": 3,
  "uuid": "6526e45d-5d5e-41c8-8c49-d68efdaab7d5",
  "name": "API_Example",
  "mountpoint": "/mmfs1/data/API_Example",
  "sites": [
    {
      "id": 1,
      "name": "siteA",
      "label": "siteA",
      "pool": "sas1",
      "usage": [0, 1099511627776]
    }
  ],
  "created": "2025-07-04T14:18:07.227156Z",
```

```
  "size": 1099511627776,
  "snapshot_schedule": {
    "id": 2,
    "frequency": "1 00:00:00",
    "time": "00:13:00",
    "duration": "7 00:00:00",
    "space": 3
  },
  "on_sites": {
    "siteA": {
      "name": "API_Example",
      "size": 1099511627776,
      "present": true,
      "syncing": true
    }
  },
  "immutable": false,
  "editable": true,
  "color": [111, 64, 191],
  "permission_mode": null,
  "relationships": {
    "shares": [],
    "policies": [],
    "schedules": [
      {
        "id": 3,
        "name": "Test4",
        "discovery": "recursive"
      }
    ]
  },
  "nested_shares": {
    "samba": [],
    "nfs": []
  },
  "is_ready": true
}
```


Response Fields

Field	Description
`url`	API endpoint for the specific space.
`id`	Space ID.
`uuid`	Universally unique identifier for the space.
`name`	Display name of the space.
`mountpoint`	Filesystem path to the space.
`sites`	List of site objects where this space exists.
└─ `id`, `name`, `label`	Site metadata.
└─ `pool`, `usage`	Pool name and disk usage in bytes (`[used, total]`).
`created`	Timestamp of creation (ISO 8601 format).
`size`	Total size allocated for the space.

Field	Description
`snapshot_schedule`	Snapshot configuration and associated space ID.
`└─ frequency`, `time`, etc.	Snapshot timing and duration settings.
`on_sites`	Dictionary of site names with current sync status.
`└─ present`, `syncing`	Booleans showing sync and availability.
`immutable`	`true` if read-only; otherwise `false`.
`editable`	Whether the space settings can be changed.
`color`	RGB color code used for UI display (e.g., `[255, 255, 255]`).
`permission_mode`	Permissions mode (can be `null`).
`relationships`	Nested relationships with other objects.
`└─ shares`, `policies`	Empty lists or linked share/policy objects.
`└─ schedules`	Discovery or snapshot schedules attached.
`nested_shares`	Samba and NFS shares created within this space, including shares from any spaces nested under it. For example, a top-level space like `mmfs1` will list all shares, as all other spaces are nested beneath it.
`is_ready`	`true` if the space is initialized and ready.

DELETE /spaces/{id}/

Description

Deletes a specific Space identified by its `ID`. This permanently removes the space from the system.

Request Parameters

Name	Description	Type	Required	Location
`id`	The unique identifier of the Space to delete	string	Yes	Path

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/spaces/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `204 No Content`

No response body is returned. The 204 status confirms successful deletion.

GET /spaces/{id}/files/

Description

Retrieves the file or directory structure inside a specific space, identified by its `ID`. This includes metadata such as file size, type, path, access times, and status across different sites. Useful for browsing contents within the space mount point.

Request Parameters

Name	Type	Required	Description
`id`	string	Yes	ID of the space to browse.
`path`	string	Yes	Path within the space to retrieve files from. Must start with `/`.
`site`	string	No	Specific site to target. If omitted, checks all available sites.
`details`	boolean	No	Whether to retrieve all details of child files. Default: `false`.
`restricted`	boolean	No	Whether to include restricted objects. Default: `false`.
`check_available_sites`	boolean	No	Whether to only check available sites. Default: `true`.
`raise_exception`	boolean	No	Whether to raise an exception if a timeout occurs. Default: `true`.
`custom_refresh_period`	integer	No	Custom DB cache invalidation time in seconds.
`custom_timeout`	integer	No	Custom timeout period for the operation in seconds.
`type`	string	No	Filter by item type (e.g., `file`, `directory`).
`page`	integer	No	Page number for pagination.
`page_size`	integer	No	Number of results per page. Can be `0` (no pagination), or between `20`–`100`.

Example Request

```
curl -X 'GET' \
  'http://10.201.2.224:8000/api/spaces/1/files/?
path=%2Fmmfs1%2Fdata&details=false&restricted=false&check_availab
le_sites=true&raise_exception=true' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
[
  {
```

```
    "url": "http://your-hub-address:8000/api/file/?
path=%2Fmmfs1%2Fdata&site=qatest-pdevan-testplugin-siteB",
    "children": {
      "count": 2,
      "results": [
        {
          "site": "siteB",
          "path": "/mmfs1/data/sample_data",
          "name": "sample_data",
          "type": "directory",
          "size": 0,
          "status": "online"
        },
        {
          "site": "siteB",
          "path": "/mmfs1/data/sample_data.tgz",
          "name": "sample_data.tgz",
          "type": "file",
          "size": 0,
          "status": "online"
        }
      ]
    },
    "site": "siteB",
    "path": "/mmfs1/data",
    "name": "data",
    "type": "directory",
    "size": 512,
    "status": "online"
  },
  {
    "url": "http://your-hub-address:8000/api/file/?
path=%2Fmmfs1%2Fdata&site=siteA",
    "children": {
      "count": 5,
      "results": [
        {
          "site": "siteA",
          "path": "/mmfs1/data/API_Example",
          "name": "API_Example",
          "type": "directory",
          "size": 0,
          "status": "online"
        },
        {
          "site": "siteA",
          "path": "/mmfs1/data/sample_data.tgz",
          "name": "sample_data.tgz",
          "type": "file",
          "size": 36375247,
          "status": "online",
          "disk_usage": 184025088,
          "total_files": 94,
          "data_size": 178229989
```

```
        }
      ]
    },
    "site": "siteA",
    "path": "/mmfs1/data",
    "name": "data",
    "type": "directory",
    "size": 512,
    "status": "online"
  }
]
```

Response Fields

Field	Type	Description
`url`	string	Link to file/directory API for this site.
`children`	object	Contains count, pagination, and results for sub-items.
`site`	string	Site this structure refers to.
`path`	string	Full path to the current folder.
`name`	string	Folder name.
`type`	string	Either `"file"` or `"directory"`.
`size`	integer	Size in bytes.
`status`	string	Status of the object, e.g., `"online"`.
`ctime` / `mtime` / `atime`	datetime	Creation, modification, and access timestamps (nullable).
`disk_usage`	integer	Disk usage in bytes (optional; present for files/directories).
`total_files`	integer	Total number of files under the directory (if available).
`data_size`	integer	Actual data size in bytes.
`deleted`	boolean	Whether it's deleted (soft-deleted).

Buckets

GET /api/buckets/

Description

This API endpoint retrieves a list of *storage buckets* configured in the system. Buckets represent logical containers for storing data, such as files or backups, often linked to cloud storage services like Amazon S3.

You can paginate through the results using `page` and `page_size` parameters.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X GET \
  'http://your-hub-address:8000/api/buckets/?page=1&page_size=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "access_key_id": "**********",
      "secret_access_key": "**********",
      "access_key": "",
      "credentials_json": {},
      "external_targets": [],
      "name": "buckettest",
      "label": "BT 1",
      "storage_type": "AmazonS3",
      "container": "",
      "storage_account": "",
      "credentials_file": null,
      "region": "London",
      "endpoint": "s3.amazonaws.com",
      "port": 443,
      "scheme": "https",
      "migration_target_folder": "",
      "ssl_verify": true,
      "soft_deleting": false,
      "advanced_settings": {}
    }
  ]
}
```

Response Fields

Field	Type	Description
`id`	integer	Unique identifier for the bucket.
`access_key_id`	string (masked)	The access key ID used to authenticate with the storage service (masked for security).
`secret_access_key`	string (masked)	The secret access key for authentication (masked for security).
`access_key`	string	Azure-specific access key information, if applicable. Used in place of `access_key_id` as required by Azure (unlike S3).
`credentials_json`	object	JSON object with detailed credential info.
`external_targets`	array	List of external targets linked to this bucket.
`name`	string	Name of the bucket.
`label`	string	User-friendly label or alias for the bucket.
`storage_type`	string	Type of storage service (e.g., `"AmazonS3"`).
`container`	string	Container name or identifier (if applicable).
`storage_account`	string	Storage account identifier (if applicable).
`credentials_file`	string / null	Path to credentials file, if used.
`region`	string	Geographical region of the storage bucket (e.g., `"London"`).
`endpoint`	string	URL endpoint for the storage service (e.g., `"s3.amazonaws.com"`).
`port`	integer	Network port used to connect (typically `443` for HTTPS).
`scheme`	string	Protocol scheme, usually `"https"`.
`migration_target_folder`	string	Folder path for migration targets, if applicable.
`ssl_verify`	boolean	Whether SSL certificate verification is enabled.
`advanced_settings`	object	Additional advanced settings specific to the bucket.

POST /api/buckets/

Description

This API endpoint creates a new storage bucket configuration. Buckets represent storage containers in cloud services like Amazon S3, and this endpoint lets you add new ones by providing the necessary connection and authentication details.

Request Parameters

Name	Type	Required	Description
name	string	Yes	Name of the bucket. Must contain only lowercase letters, numbers, periods (.) and dashes (-).
label	string	No	User-friendly label or alias for the bucket.
storage_type	string	Yes	Type of storage service (e.g., “AmazonS3”).
container	string	No	Container name or identifier, if applicable.
access_key_id	string	Yes	Access key ID used for authenticating with the storage service.
secret_access_key	string	Yes	Secret key paired with the access key ID for authentication.
access_key	string	No	Additional access key info, if any.
storage_account	string	No	Storage account identifier, if applicable.
credentials_json	object	No	JSON object containing detailed credential information.
credentials_file	string	No	Path to a credentials file, if used.
region	string	No	Geographical region of the bucket (e.g., “us-west-1”).
endpoint	string	No	URL endpoint for the storage service (e.g., “s3.amazonaws.com”).
port	integer	No	Network port for connecting (default usually 443 for HTTPS).
scheme	string	No	Protocol scheme, usually “https”.
migration_target_folder	string	No	Folder path for migration targets, if applicable.
ssl_verify	boolean	No	Whether SSL certificate verification is enabled (default: true).
advanced_settings	object	No	Additional advanced settings specific to the bucket.

Example Request

```
curl -X POST \
  'http://your-hub-address:8000/api/buckets/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "bucket1",
    "label": "test 2",
    "storage_type": "AmazonS3",
    "container": "string",
    "access_key_id": "string",
```

```
    "secret_access_key": "string",
    "access_key": "string",
    "storage_account": "string",
    "credentials_json": {},
    "credentials_file": "string",
    "region": "string",
    "endpoint": "string",
    "port": 2147483647,
    "scheme": "https",
    "migration_target_folder": "string",
    "ssl_verify": true,
    "advanced_settings": {}
  }'
```

Successful Response

HTTP Status Code: `201 CREATED`

Content-Type: `application/json`

Example Response


```
{
  "id": 2,
  "access_key_id": "********************",
  "secret_access_key": "********************",
  "access_key": "********************",
  "credentials_json": {},
  "external_targets": [],
  "name": "bucket1",
  "label": "test 2",
  "storage_type": "AmazonS3",
  "container": "string",
  "storage_account": "string",
  "credentials_file": "string",
  "region": "string",
  "endpoint": "string",
  "port": 2147483647,
  "scheme": "https",
  "migration_target_folder": "string",
  "ssl_verify": true,
  "soft_deleting": false,
  "advanced_settings": {}
}
```


Response Fields

Please refer to the request parameters.

GET /buckets/{id}/

Description

To retrieve details of a specific bucket by its unique ID.

Request Parameters

Name	Type	Required	Description
id	string	Yes	Unique identifier of the bucket.

Example Request

```
curl -X GET \
  'http://your-hub-address:8000/api/buckets/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "id": 2,
  "access_key_id": "**********",
  "secret_access_key": "**********",
  "access_key": "**********",
  "credentials_json": {},
  "external_targets": [],
  "name": "bucket1",
  "label": "test 2",
  "storage_type": "AmazonS3",
  "container": "string",
  "storage_account": "string",
  "credentials_file": "string",
  "region": "string",
  "endpoint": "string",
  "port": 2147483647,
  "scheme": "https",
  "migration_target_folder": "string",
  "ssl_verify": true,
  "soft_deleting": false,
  "advanced_settings": {}
}
```


Response Fields

The fields returned in the response are explained at **POST /api/buckets/**. Please refer to those descriptions for more information about each field.

PATCH /buckets/{id}/

Description

Update details of an existing bucket by its unique `ID`.

Request Parameters

Name	Description
id (path)	The unique identifier of the bucket to update.
data (body)	JSON object containing fields to update for the bucket.

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/buckets/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "bucket2",
    "label": "string",
    "storage_type": "AmazonS3",
    "container": "string",
    "access_key_id": "string",
    "secret_access_key": "string",
    "access_key": "string",
    "storage_account": "string",
    "credentials_json": {},
    "credentials_file": "string",
    "region": "string",
    "endpoint": "string",
    "port": 2147483647,
    "scheme": "https",
    "migration_target_folder": "string",
    "ssl_verify": true,
    "advanced_settings": {}
  }'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "id": 2,
  "access_key_id": "********************",
  "secret_access_key": "********************",
  "access_key": "********************",
  "credentials_json": {},
  "external_targets": [],
  "name": "bucket2",
  "label": "string",
  "storage_type": "AmazonS3",
  "container": "string",
  "storage_account": "string",
```

```
  "credentials_file": "string",
  "region": "string",
  "endpoint": "string",
  "port": 2147483647,
  "scheme": "https",
  "migration_target_folder": "string",
  "ssl_verify": true,
  "soft_deleting": false,
  "advanced_settings": {}
}
```

Response Fields

The fields returned in the response are explained at **POST /api/buckets/**. Please refer to those descriptions for more information about each field.

DELETE /buckets/{id}/

Description

Deletes a specified **bucket** by `ID`. This operation also propagates the deletion settings to external targets across all sites where the bucket is available.

It is used for complete removal of the bucket from the NGENEA Hub as well as from any connected external storage systems, based on the system's configured deletion policies.

Request Parameters

Name	Type	Required	Description
id	string	Yes	The unique identifier of the bucket.

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/buckets/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <api-key>'
```

Successful Response

HTTP Status Code: `204 No Content`

Content-Type: `application/json`

No content is returned in the response body upon successful deletion.

> **Note:** This operation is **irreversible**, once deleted, the bucket and its settings are removed from all locations.
>
> Ensure that the bucket is not in use before calling this endpoint.

Deletion is subject to permission and may fail if the bucket is protected or referenced by other resources.

Storage Pools

A storage pool is a logical collection of physical storage resources, such as hard drives or SSDs, grouped together to simplify management and optimize the usage of available storage.

In systems like Pixstor (GPFS), storage pools allow administrators to treat multiple devices as a single unit, enabling better scalability, load balancing, and efficient data management. These pools are essential for organizing and managing file systems across distributed environments.

In Ngenea Hub, storage pool metadata, including capacity and health status, is regularly refreshed to ensure accurate tracking and monitoring. By using storage pools, administrators can easily manage space, track storage utilization, and ensure system performance remains optimal.

GET /storagepools/

Description

The `GET /storagepools/` API retrieves a list of storage pools within the Ngenea Hub system, showing details like the pool's capacity, usage, and the site it belongs to.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/storagepools/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "name": "sas1",
      "filesystem": "mmfs1",
      "site": 1,
      "capacity": 89120571392,
      "used_space": 3070230528,
      "free_space": 86050340864,
      "is_default": true
    },
    {
      "name": "sas1",
      "filesystem": "mmfs1",
      "site": 2,
      "capacity": 89120571392,
      "used_space": 2944401408,
      "free_space": 86176169984,
      "is_default": true
    }
  ]
}
```


Response Fields

Field	Description	Example
count	The total number of storage pools available.	2
next	The URL for the next page of results, if available. `null` if no more pages.	null
previous	The URL for the previous page of results, if available. `null` if first page.	null
results	An array containing details about each storage pool.	

Each entry in `results` contains the following fields:

Field	Description	Example
name	The name of the storage pool.	“sas1”
filesystem	The file system associated with the pool.	“mmfs1”
site	The site ID where the storage pool is located.	1
capacity	The total capacity of the storage pool in bytes.	89120571392
used_space	The amount of space currently used in the pool (in bytes).	3070230528
free_space	The remaining space available in the pool (in bytes).	86050340864
is_default	A boolean indicating if this is the default storage pool.	true

GET /storagepools/{id}/

Description

This endpoint is used to retrieve detailed information about a specific storage pool identified by the 'ID' parameter. This is a more focused API call compared to the previous one, as it allows you to get data for a particular storage pool rather than a list of all storage pools.

Request Parameters

Field	Description	Example
id	The unique identifier (ID) for the specific storage pool to retrieve.	2

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/storagepools/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "name": "sas1",
  "filesystem": "mmfs1",
  "site": 2,
  "capacity": 89120571392,
  "used_space": 2961178624,
  "free_space": 86159392768,
  "is_default": true
}
```

Response Fields

Please refer to the response fields under **GET/storagepools/**.

Note: In the example response:

- The storage pool is named **"sas1"** and uses the **"mmfs1"** filesystem.
- It is located at **site 2**.
- The total **capacity** of the pool is approximately **89 GB**, with around **3 GB** used and **86 GB** remaining.
- The pool is marked as the **default** pool (true).

API Endpoints - Configuration & System Settings

Configurations

GET /configurations/

Description

This endpoint retrieves the current runtime configuration settings of the **Ngenea Hub**.

These are dynamic configurations, they can be read (and potentially modified through other endpoints , explained below) without requiring a restart of the Hub. This contrasts with the static configuration values that are set during the initial Hub setup, which typically require a restart to take effect.

These configuration values control various operational aspects of the **Ngenea Hub**, such as search behavior, job lifecycle, timeouts, analytics, and user management settings.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/configurations/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
    "jobs_ttl": 90,
    "search_backend": "analytics",
    "search_max_results": 200,
    "search_result_ttl": 7,
    "snapshot_create_delete_retry_timeout": 1000,
    "stat_timeout": 10,
    "health_timeout": 1,
    "stat_refresh_period": 30,
    "task_invalidation_timeout": 3600,
    "snapdiff_stream_timeout": 10,
    "force_local_managed_users": false,
    "custom_statistics_tasks": [],
    "analytics_timeout": 10,
    "maximum_external_results": 100,
    "node_health_last_heartbeat_interval": 120,
    "initial_uid": 100000,
```

```
    "initial_gid": 100000,
    "default_space_location": "data",
    "schedules_enabled": true,
    "salt_task_timeout": 60.0,
    "keycloak_jwks_timeout": 3600,
    "filebrowser_proxies_backend": "pixstor_search"
}
```

Response Fields

Field	Type	Description
`jobs_ttl`	integer	How long job details should be stored after job completion, in days.
`search_backend`	string	Backend to use when performing searches. Currently supported backends: `analytics`, `pixstor_search`.
`search_max_results`	integer	Maximum number of search results to fetch from the search backend, per site. Some backends may have a hard limit of 10,000 results.
`search_result_ttl`	integer	How long search results should be stored, in days.
`snapshot_create_delete_retry_timeout`	integer	Time in seconds after which workers will give up retrying snapshot create and delete operations. Set to 0 to disable retries.
`stat_timeout`	integer	How long to wait for results from the `/api/file/` endpoint, in seconds.
`health_timeout`	integer	How long to wait for an active worker response for Space browser operations before the worker is considered unavailable
`stat_refresh_period`	integer	How long a file's details will be retained as a cache within the file browser, in seconds.
`task_invalidation_timeout`	integer	Idle timeout (in minutes) when waiting for delete and move tasks to complete during snapdiff workflows.
`snapdiff_stream_timeout`	integer	Timeout in seconds for snapdiff stream operations.
`force_local_managed_users`	boolean	Allows local NAS users to be managed on AD-joined sites.
`custom_statistics_tasks`	array	A list of custom tasks that add their reported file count to the job total, regardless of when they're executed in a workflow.

Field	Type	Description
`analytics_timeout`	integer	How long to wait for results from search operations when using the analytics backend, in seconds.
`maximum_external_results`	integer	Maximum number of items to retrieve per page from an external target scan.
`node_health_last_heartbeat_interval`	integer	Maximum allowed time (in seconds) between the last recorded heartbeat and now for a node to be considered online.
`initial_uid`	integer	Minimum UID value for NAS users that can be configured.
`initial_gid`	integer	Minimum GID value for NAS groups that can be configured.
`default_space_location`	string	The default location to create Spaces on all filesystems.
`schedules_enabled`	boolean	Whether automated tasks (Policies and Scheduled Workflows) are enabled.
`salt_task_timeout`	integer	How long to wait for settings tasks to return results, in seconds.
`keycloak_jwks_timeout`	integer	How long to cache the JSON Web Key Set (JWKS) used to verify Keycloak JSON Web Tokens (JWTs)
`filebrowser_proxies_backend`	string	Which service provides proxies to the Hub UI. Valid options are: `null`,`pixstor_search`.

PATCH /configurations/

Description

Updates one or more runtime configuration settings on the NGENEA Hub.

These settings are applied without requiring a restart of the Hub. They influence the behavior of search, task execution, user management, and other key services.

This endpoint allows partial updates, only the fields provided in the request body are modified.

Request Parameters

Name	Type	Required	Location	Description
`data`	object	Yes	Body	A JSON object with one or more configuration fields to update.

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/configurations/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <REDACTED>' \
  -H 'Content-Type: application/json' \
  -d '{
    "jobs_ttl": 90,
    "search_backend": "analytics",
    "search_max_results": 200,
    "search_result_ttl": 7,
    "snapshot_create_delete_retry_timeout": 1000,
    "stat_timeout": 10,
    "stat_refresh_period": 30,
    "task_invalidation_timeout": 3600,
    "snapdiff_stream_timeout": 10,
    "force_local_managed_users": false,
    "custom_statistics_tasks": [
      "dynamo.custom.get_all_pngs"
    ],
    "analytics_timeout": 100,
    "maximum_external_results": 100,
    "node_health_last_heartbeat_interval": 120,
    "initial_uid": 100000,
    "initial_gid": 100000,
    "default_space_location": "data",
    "schedules_enabled": true,
    "salt_task_timeout": 60.0,
    "keycloak_jwks_timeout": 3600,
    "filebrowser_proxies_backend": "pixstor_search"
}'
```

Request Body Fields

Field	Value	Description
`jobs_ttl`	90	Time-to-live records (in day
`search_backend`	“analytics”	The backend e for search oper
`search_max_results`	200	Maximum nu results retur search query.
`search_result_ttl`	7	Time-to-live (ir cached search
`snapshot_create_delete_retry_timeout`	1000	Timeout in m before retrying create/delete o
`stat_timeout`	10	Timeout in se file stat operati

Field	Value	Description
`health_timeout`	1	How long to w active worker for Space operations b worker is unavailable
`stat_refresh_period`	30	Refresh inte seconds) for statistics.
`task_invalidation_timeout`	3600	Time in second background becomes invali
`snapdiff_stream_timeout`	10	Timeout in se snapdiff operations (eff timeout).
`force_local_managed_users`	false	Enforces management handled loca than externally
`custom_statistics_tasks`	[“dynamo.custom.get_all_pngs”]	A list of custom advanced processing.
`analytics_timeout`	10	Timeout in se analytics (effectively no
`maximum_external_results`	100	Max number returned by search (effectively no
`node_health_last_heartbeat_interval`	120	Interval in se checking t heartbeat o (effectively no
`initial_uid`	100000	Starting UID f generated user
`initial_gid`	100000	Starting GID f generated grou
`default_space_location`	“data”	The default l create Space filesystems.
`schedules_enabled`	true	Whether sched are enabled on
`salt_task_timeout`	60.0	Timeout in se tasks hand SaltStack orche means no time

Field	Value	Description
`keycloak_jwks_timeout`	3600	How long to JSON Web Key used to verify JSON Web Toke
`filebrowser_proxies_backend`	“pixstor_search”	Which service proxies to the Valid optio `null,pixstor_`

Note: All the data provided above is for example purposes only.

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
    "jobs_ttl": 90,
    "search_backend": "analytics",
    "search_max_results": 200,
    "search_result_ttl": 7,
    "snapshot_create_delete_retry_timeout": 1000,
    "stat_timeout": 10,
    "health_timeout": 1,
    "stat_refresh_period": 30,
    "task_invalidation_timeout": 3600,
    "snapdiff_stream_timeout": 10,
    "force_local_managed_users": false,
    "custom_statistics_tasks": [],
    "analytics_timeout": 10,
    "maximum_external_results": 100,
    "node_health_last_heartbeat_interval": 120,
    "initial_uid": 100000,
    "initial_gid": 100000,
    "default_space_location": "data",
    "schedules_enabled": true,
    "salt_task_timeout": 60.0,
    "keycloak_jwks_timeout": 3600,
    "filebrowser_proxies_backend": "pixstor_search"
}
```


Response Fields

Please refer to the response fields under **GET /configurations/**.

Features - Feature Flag

Feature flags are configuration settings used to enable or disable specific functionality within an application without requiring code changes or redeployments. They are especially useful for testing new or experimental features in a controlled manner.

In Ngenea Hub, feature flags allow administrators to preview features that may still be under development or evaluation.

By default, all preview features are disabled to ensure system stability. However, they can be selectively enabled for testing or early access. It's important to understand that enabling these features comes with the risk of encountering incomplete or unstable behavior.

To interact with feature flags, you can use the ngclient command-line tool or make requests via the Ngenea Hub REST API.

- To turn features on or off using the command line, see the section: Via ngclient command-line tool
- To see the results using Swagger, see below.

GET /features/

Description

This API endpoint is used to **list all available feature flags** in the system. You can see which features are currently available and whether they are enabled or disabled.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/features/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "count": 0,
  "next": null,
  "previous": null,
  "results": []
}
```

Response Fields

Field	Type	Description
`count`	integer	Total number of available features.
`next`	string	URL to the next page of results, or `null` if this is the last page.
`previous`	string	URL to the previous page of results, or `null` if this is the first page.
`results`	array	List of feature objects (name, description, and enabled status). Empty if no features are available.

There are **no feature flags** currently available.

GET /features/{name}/

Description

This endpoint is used to retrieve details of a specific feature flag by its name. For example, to check the status of a feature named `searchui`.

Request Parameters

Name	Type	In	Description
`name`	string	path	(Required) The name of the feature to look up.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/features/searchui/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <api_key>'
```

Response

HTTP Status Code: `404 Not Found`

Example Response

```
{
  "detail": "No Feature matches the given query."
}
```

For example, we used `searchui` in the request. When the feature flag is available, this will return its details such as name, description, and whether it is enabled.

PATCH /features/{name}/

Description

This endpoint is used to **enable or disable a specific feature flag** by name. You send a request with `{"enabled": true}` to turn the feature **on**, or `{"enabled": false}` to turn it **off**.

As there are **no feature flags currently available**, we are using `searchui` as an **example only** in this request.

Request Parameters

Name	Type	In	Description
`name`	string	path	(Required) The name of the feature flag to update.
`data`	object	body	(Required) JSON object with the `enabled` field set to `true` or `false`.

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/features/searchui/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key **********' \
  -H 'Content-Type: application/json' \
  -d '{
  "enabled": true
}'
```

Successful Response

HTTP Status Code: `404 Not Found`

Example Response

```
{
  "detail": "No Feature matches the given query."
}
```

Nodes

GET /nodes/

Description

This API endpoint retrieves a list of **worker nodes** that are registered with the Ngenea Server. Each node represents a system running the ngenea-worker component. This endpoint supports pagination, and each node includes details such as its name, site, status (online/offline), and last heartbeat timestamp.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/nodes/?page=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "url": "http://<server-ip>:<port>/api/nodes/1/",
      "id": 1,
      "name": "node-siteA-1",
      "site": "siteA",
      "online": true,
      "has_default_queue": true,
      "last_heartbeat": "2025-06-30T13:04:44.758754Z"
    },
    {
      "url": "http://<server-ip>:<port>/api/nodes/2/",
      "id": 2,
      "name": "node-siteB-1",
      "site": "siteB",
      "online": true,
      "has_default_queue": true,
      "last_heartbeat": "2025-06-30T13:04:44.845662Z"
    }
```

```
  ]
}
```

Response Fields

Field	Type	Description
`count`	integer	Total number of nodes in the system.
`next`	string or null	URL to the next page of results, if any.
`previous`	string or null	URL to the previous page of results, if any.
`results`	array	List of node objects. Each object includes the following fields:
→ `url`	string	API URL for the specific node.
→ `id`	integer	Unique ID of the node.
→ `name`	string	Name of the node.
→ `site`	string	Site or cluster the node belongs to.
→ `online`	boolean	Indicates if the node is currently online.
→ `has_default_queue`	boolean	Indicates if the node has a default task queue assigned.
→ `last_heartbeat`	string (timestamp)	Timestamp of the last time the node checked in (ISO 8601 format).

GET /nodes/{id}/

Description

This API endpoint retrieves detailed information for a specific worker node identified by its unique ‘ID’. This is useful for monitoring or managing individual nodes.

Request Parameters

Name	Type	Location	Required	Description
`id`	string	path	Yes	The unique identifier of the node you want to retrieve.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/nodes/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "url": "http://<server-ip>:<port>/api/nodes/2/",
  "id": 2,
  "name": "node-siteB-1",
  "site": "siteB",
  "online": true,
  "has_default_queue": true,
  "last_heartbeat": "2025-06-30T13:05:00.863706Z"
}
```


Response Fields

Please refer to response fields under **GET /nodes/**.

DELETE /nodes/{id}/

Description

Deletes a specific worker node from the Ngenea Server by its `ID`. Once deleted, the node will no longer be registered **until it sends another heartbeat**.

This can be useful if a node has been intentionally taken offline and you no longer want the hub to report on it, otherwise, the site health status may show as *not OK*.

Request Parameters

Name	Type	Location	Required	Description
`id`	string	path	Yes	The unique identifier of the node you want to retrieve.

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/nodes/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `204 No Content`

Indicates the node was successfully deleted. There is no response body when deletion is successful.

Settings Tasks

What is a task ?

In Ngenea hub, a task is a defined unit of work that performs a specific operation on a set of files or paths.

It is typically implemented as a Python function or method that receives a standardized set of arguments, such as the job ID (`jobid`), a list of paths to process (`paths`), and optional keyword arguments.

The task processes the inputs and returns a structured payload that describes the result, including which files were processed, skipped, failed, or aborted.

Tasks follow a strict input and output format to ensure compatibility within the Ngenea ecosystem.

Why is a Task Needed?

Tasks are essential because they encapsulate discrete pieces of logic that can be reused and composed into larger workflows. Each task is responsible for carrying out a single step in a larger data processing pipeline, such as migrating data, deleting files, analyzing metadata, or verifying integrity.

How is a Task Related to a Job?

A job in Ngenea is a higher-level structure that coordinates and tracks the execution of a workflow, a series of tasks. When a job is initiated, it orchestrates the execution of its constituent tasks in a defined sequence. Each task within the job receives the job's ID, so all task results can be traced back to the originating job. The job manages the state, progress, and outcome of the entire workflow, while tasks handle the execution of individual steps.

In essence, tasks are the functional building blocks of jobs. A job cannot execute without tasks, and tasks are only meaningful when they are executed within the context of a job. Together, they form the backbone of Ngenea's data processing capabilities, allowing for flexible, trackable, and customizable workflows.

GET /settingstasks/

Description

This API endpoint retrieves a list of tasks associated with a specific job. It supports filtering by task state and task type, as well as pagination. It is typically used to monitor the status of tasks within a job.

Request Parameters

Name	Type	Required	Description
`job_id`	integer	Yes	The ID of the job whose tasks should be retrieved. Must be ≥ 1.
`page`	integer	No	Page number in the paginated result set. Defaults to 1.
`page_size`	integer	No	Number of results per page. Range: 20–100. Use 0 to disable pagination.
`tasktype`	string (max 50)	No	Filters tasks by their type (e.g., `dynamo.tasks.set_config_and_apply`).

Name	Type	Required	Description
`state`	array[string]	No	Filters tasks by their state. Accepted values include `SUCCESS`, `ERROR`, `PENDING`, etc.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/settingstasks/?
page=1&state=SUCCESS&job_id=84' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Request Body Fields

Field	Type	Description
`count`	integer	Total number of tasks matching the query.
`next`	string or null	URL to the next page of results, if any.
`previous`	string or null	URL to the previous page of results, if any.
`results`	array of objects	List of task objects. Each represents one task.
`stats`	object	Summary statistics about the returned tasks, broken down by state and tasktype.

Task Object (Inside `results`)

Field	Type	Description
`url`	string	URL of the individual task resource.
`id`	integer	Unique ID of the task.
`task_id`	string	UUID representing the task instance.
`tasktype`	string	Type of the task (function name).
`state`	string	Current state of the task (e.g., SUCCESS).
`job`	integer	Job ID the task belongs to.
`site`	string	Site where the task ran.
`started`	string	Start time in UTC (ISO 8601 datetime format).

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "url": "http://your-hub-address:8000/api/settingstasks/78/",
```

```
      "id": 78,
      "task_id": "b8bc9471-ea1f-47ec-bea7-75230452e7c5",
      "tasktype": "dynamo.tasks.set_config_and_apply",
      "state": "SUCCESS",
      "job": 84,
      "site": "siteA",
      "started": "2025-06-25T12:00:02.108785Z"
    }
  ],
  "stats": {
    "state": {
      "SUCCESS": 1
    },
    "tasktype": {
      "dynamo.tasks.set_config_and_apply": 1
    }
  }
}
```


Response Fields

Please refer to the resquest body fields.

GET /settingstasks/{id}/

Description

This API endpoint retrieves detailed information about a specific task by its 'ID'.

Request Parameters

Name	Type	Location	Required	Description
`id`	string	path	Yes	The unique identifier of the no

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/settingstasks/78/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "url": "http://your-hub-address:8000/api/settingstasks/78/",
  "id": 78,
  "task_id": "b8bc9471-ea1f-47ec-bea7-75230452e7c5",
```

```
  "tasktype": "dynamo.tasks.set_config_and_apply",
  "state": "SUCCESS",
  "started": "2025-06-25T12:00:02.108785Z",
  "completed": "2025-06-25T12:01:17.021510Z",
  "runtime": 74.912725,
  "job": 84,
  "site": "siteA",
  "request": {
    "values": {
      "ngeneahsm_targets:mmfs1-test2:Port": 443,
      "ngeneahsm_targets:mmfs1-test2:Bucket": "test2",
      "ngeneahsm_targets:mmfs1-test2:Region": "London",
      "ngeneahsm_targets:mmfs1-test2:Scheme": "https",
      "ngeneahsm_targets:mmfs1-test2:enabled": true,
      "ngeneahsm_targets:mmfs1-test2:Endpoint":
"s3.amazonaws.com",
      "ngeneahsm_targets:Test_Space-test2:Port": 443,
      "ngeneahsm_targets:mmfs1-test2:SSLVerify": true,
      "ngeneahsm_targets:Test_Space-test2:Bucket": "test2",
      "ngeneahsm_targets:Test_Space-test2:Region": "London",
      "ngeneahsm_targets:Test_Space-test2:Scheme": "https",
      "ngeneahsm_targets:mmfs1-test2:StorageType": "AmazonS3",
      "ngeneahsm_targets:Test_Space-test2:enabled": true,
      "ngeneahsm_targets:Test_Space-test2:Endpoint": "s3.amazonaws
.com",
      "ngeneahsm_targets:Test_Space-test2:SSLVerify": true,
      "ngeneahsm_targets:mmfs1-test2:DeleteOnRecall": false
      // ... more values ...
    },
    "timeout": 0,
    "encrypted": [
      "ngeneahsm_targets:mmfs1-test2:AccessKeyId",
      "ngeneahsm_targets:mmfs1-test2:SecretAccessKey",
      "ngeneahsm_targets:Test_Space-test2:AccessKeyId",
      "ngeneahsm_targets:Test_Target-buckettest:AccessKeyId",
      "ngeneahsm_targets:Test_Space-test2:SecretAccessKey",
      "ngeneahsm_targets:Test_Target-buckettest:SecretAccessKey"
    ],
    "multi_update": false
  },
  "results": {
    "value": {
      "ngeneahsm_targets:mmfs1-test2:Port": {
        "value": 443,
        "is_set": true,
        "schema_key": "ngeneahsm_targets:mmfs1-test2:Port"
      },
      "ngeneahsm_targets:mmfs1-test2:Bucket": {
        "value": "test2",
        "is_set": true,
        "schema_key": "ngeneahsm_targets:mmfs1-test2:Bucket"
      },
      "ngeneahsm_targets:mmfs1-test2:Region": {
        "value": "London",
```

```
        "is_set": true,
        "schema_key": "ngeneahsm_targets:mmfs1-test2:Region"
      },
      "ngeneahsm_targets:mmfs1-test2:Scheme": {
        "value": "https",
        "is_set": true,
        "schema_key": "ngeneahsm_targets:mmfs1-test2:Scheme"
      },
      "ngeneahsm_targets:mmfs1-test2:enabled": {
        "value": true,
        "is_set": true,
        "schema_key": "ngeneahsm_targets:mmfs1-test2:enabled"
      },
      "ngeneahsm_targets:mmfs1-test2:Endpoint": {
        "value": "s3.amazonaws.com",
        "is_set": true,
        "schema_key": "ngeneahsm_targets:mmfs1-test2:Endpoint"
      }
      // ... more results ...
    },
    "status": "succeeded",
    "apply_output": {
      "siteA-1.pixstor": "succeeded"
    }
  },
  "friendly_name": "Example Friendly Name"
}
```


Note: This API response represents the details of a completed task that configures and applies multiple Amazon S3 storage settings to a specific site in the system. The task executed successfully, confirming that the configuration parameters (including security credentials, bucket names, endpoints, and access settings) were applied without errors to the target site `siteA`. The response provides a detailed snapshot of both the requested settings and the resulting state, useful for auditing and verification purposes.

Response Fields

Field	Type	Description
`url`	string	The API URL for this specific task resource.
`id`	integer	Unique identifier of the task instance (database ID).
`task_id`	string	UUID representing the specific task instance.
`state`	string	Current status of the task, e.g., `"SUCCESS"`, `"ERROR"`.
`started`	string	Timestamp when the task started (ISO 8601 UTC datetime).
`completed`	string	Timestamp when the task completed (ISO 8601 UTC datetime).
`runtime`	float	Duration of the task in seconds (completion time - start time).

Field	Type	Description
`results`	object	Detailed output of the task after completion, including values, status, and apply output summary.
`friendly_name`	string	A user-friendly descriptive name for the task (optional).

Tags

A **Tag** is a case-sensitive, user-defined keyword used to categorize and organize assets. Tags help group related assets together, making it easier to manage and retrieve them within your environment.

Tags are a core part of the **Search** functionality, powered by **PixStor Search**. Once applied, Tags allow users to filter and find assets more efficiently using the Search interface. Tags are indexed asynchronously, so there may be a short delay before newly applied Tags are searchable.

While there is no dedicated endpoint to create or assign Tags directly, Tags can be retrieved using the GET /tags endpoint. To apply Tags to assets, use the PATCH /search/metadata endpoint to update asset metadata with the desired Tags.

For more details, refer to the Search documentation and **Search API reference** to learn how Tags integrate with your search workflows.

GET /tags/

Description

This API endpoint allows clients to retrieve a list of tags. You can optionally filter the results by passing a substring to match tag names.

Request Parameters

Name	Type	Description	Required	Constraints
filter	string	Substring that returned tags must contain (query)	No	Minimum length: 1

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/tags/?filter=ex' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
[
  {
    "name": "example"
  },
  {
    "name": "three"
  },
  {
    "name": "two"
  },
  {
    "name": "one"
  }
]
```


Response Fields

Field	Type	Description
name	string	Name of the tag

Servers

Servers are integral components of a `site`, and they work together to handle different data operations, including file storage, retrieval, and data synchronization. They also facilitate network interactions and ensure that data is accessible to users or other systems within the network.

Servers are closely linked to `sites`, as they are the primary units within a site that enable the actual storage and management of data, with configurations being centrally managed through Ngenea Hub for optimal performance and monitoring.

No servers will be listed unless the gateway role is enabled on the site's Pixstor system.

To enable the gateway role, run: `pixstor-enablefeature gateway`

After enabling, it may take some time for CTDB servers to appear in the Ngenea Hub. To force detection and synchronization, you can run the following Salt commands:

```
salt '' saltutil.sync_grains
salt '' mine.update
ngclient sites refresh
```

These steps ensure the server data is refreshed and made available in Hub for further configuration of network interfaces.

GET /servers/

Description

The `GET /servers/` endpoint retrieves a list of servers associated with configured sites. Each server entry includes its name, the associated site ID, and a list of network interfaces available on that server. This data is useful for viewing and managing the network configuration of servers within a site.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/servers/?page=1&page_size=20'
\
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "name": "siteA-1.pixstor",
      "site": 1,
      "list_of_interfaces": [
        "ens5",
        "ens6",
        "dummy0",
        "br-7cec54d64949",
        "br-88ea4b8e41a3",
        "br-02ebdc5f687f",
        "br-d15e54973410",
        "br-07214fcfa2b6",
        "br-bed322cf088d"
      ]
    }
  ]
}
```


Response Fields

Field	Type	Description
`count`	integer	Total number of server records available
`next`	string / null	URL to next page of results, or null if no next page
`previous`	string / null	URL to previous page of results, or null if on first page
`results`	array of objects	List of server objects

Server Object (Inside `results`)

Field	Type	Description
`name`	string	Fully qualified name of the server
`site`	integer	ID of the site the server belongs to
`list_of_interfaces`	array of strings	Network interfaces available on the server

GET /servers/{id}/

Description

Retrieves detailed information about a specific server by its unique ID, including the server's name, associated site, and list of network interfaces.

Request Parameters

Name	Type	Location	Required	Description
`id`	string	path	Yes	The unique identifier of the node you want to retrieve.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/servers/1/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "name": "siteA-1.pixstor",
  "site": 1,
  "list_of_interfaces": [
    "ens5",
    "ens6",
    "dummy0",
    "br-7cec54d64949",
    "br-88ea4b8e41a3",
    "br-02ebdc5f687f",
    "br-d15e54973410",
    "br-07214fcfa2b6",
    "br-bed322cf088d"
  ]
}
```


Response Fields

Please refer to Response Fields under **GET /servers/**.

API Endpoints - Workflows, Automation & Jobs

Workflows

Workflows are thoroughly covered under Custom Workflows, but here's a quick overview of the essential fields required to `POST` a workflow using Swagger:

Concept	Description
Workflow	A workflow is a set of rules and steps designed to automatically process files based on their current state or condition. It functions like a recipe, guiding the system on which files to examine, what conditions to check, and which actions to perform.

Concept	Description
Define a Workflow	A workflow is defined by specifying a few key elements: a `Name`, which is a unique identifier with no spaces (e.g., `send_files`); a `Label`, which is a user-friendly display name that can include spaces; visual `Icons` for UI representation; `Filter Rules` to determine which files should be acted upon; and `Fields`, which are inputs or parameters required when the workflow is executed. You can also optionally include discovery tasks that help locate files dynamically.
Filter Rules	Filter rules are conditions within a workflow that determine which files should be included for processing. These rules can match file states (such as new or modified) and file types (like files, directories, symlinks, or all). Filters can include or exclude files based on path patterns (for example, only processing `.txt` files). Special filter states include `default`, which catches unmatched files, and `all`, which applies the rule to all files regardless of state or type.
Actions	Actions are the specific tasks performed on files that match the filter rules. Each action has a task name, such as “migrate file” or “delete file”, and can optionally include parameters like the target site or processing queue. These define what happens to each selected file.
Runtime Fields	Runtime fields are additional pieces of information requested from the user when a workflow is executed. They allow the workflow to be dynamic and customizable at runtime instead of relying solely on hardcoded values. Examples include entering a target site, choosing from dropdowns, or checking boxes. Supported field types include free text, numbers, checkboxes, and dropdowns.
Running Workflows	Workflows can be executed in multiple ways. Users can select files or folders through a user interface and apply the desired workflow. Alternatively, workflows can be triggered through an API call by providing a JSON input that includes file paths, site, queue, workflow name, and any required fields. Additionally, discovery tasks can be used to automatically search for files inside selected folders, enabling recursive or change-based file detection.
Discovery Tasks	Discovery tasks are processes used to scan input paths and find files within them. For example, the `recursive` task identifies all files inside folders, while `snapdiff` detects files that have changed since the last scan, including created, updated, moved, or deleted files. These tasks help automate and refine the file selection process.
Includes & Excludes	Include and exclude filters help refine which files a workflow will consider. “Include” patterns limit processing to only those files that match certain criteria, such as `*.txt` files. “Exclude” patterns prevent files from being processed, like ignoring `*.tmp` files. These filters can be applied globally, per individual filter rule, or for a specific workflow run.

GET /workflows

Description

This endpoint retrieves a list of workflows configured in the system. It supports pagination through query parameters and returns detailed information about each workflow, including their names, labels, discovery methods, filter rules, fields, and operational status.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X GET "http://your-hub-address:8000/api/workflows/?page=1" \
     -H "accept: application/json" \
     -H "Authorization: Api-Key <your-api-key>"
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "count": 16,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "name": "migrate",
      "label": "Dehydrate",
      "icon_classes": [
        "hurricane-01"
      ],
      "discovery": "recursive",
      "enabled": true,
      "visible": true,
      "schedule_only": false,
      "external_only": false,
      "fields": [
        {
          "name": "lock_level",
          "type": "choices",
          "label": "Locking mode",
          "choices": [
            {
```

```
            "label": "Per block (Implicit)",
            "value": "implicit"
          }
        ],
        "default": "implicit"
      }
    ],
    "filter_rules": [
      {
        "type": "all",
        "state": "all",
        "action": [
          {
            "name": "dynamo.tasks.migrate",
            "lock_level": "*lock_level"
          }
        ],
        "description": "Migrates a file off from a given path"
      }
    ],
    "discovery_options": {
      "allow_missing_paths": false
    },
    "preferred_queue": null
  },
  {
    "id": 2,
    "name": "premigrate",
    "label": "Pre-Stage",
    "icon_classes": [
      "check-done-01"
    ],
    "discovery": "recursive",
    "enabled": true,
    "visible": true,
    "schedule_only": false,
    "external_only": false,
    "fields": [
      {
        "name": "lock_level",
        "type": "choices",
        "label": "Locking mode",
        "choices": [
          {
            "label": "Per block (Implicit)",
            "value": "implicit"
          }
        ],
        "default": "implicit"
      }
    ],
    "filter_rules": [
      {
        "type": "all",
```

```
          "state": "all",
          "action": [
            {
              "name": "dynamo.tasks.migrate",
              "lock_level": "*lock_level",
              "premigrate": true
            }
          ],
          "description": "Premigrates a file on a given site"
        }
      ///////// - more workflows
      ],
      "discovery_options": {
        "allow_missing_paths": false
      },
      "preferred_queue": null
    }
      "discovery_options": {
        "allow_missing_paths": false
      },
      "preferred_queue": null
  ]
    }
```

Response Fields

Field	Description
id	Unique workflow identifier (e.g., `1`).
name	Short, unique workflow name (e.g., `"migrate"`).
label	User-friendly name shown in the UI (e.g., `"Dehydrate"`).
icon_classes	List of CSS icon classes for UI display (e.g., `["hurricane-01"]`).
discovery	Discovery method used to find files (e.g., `"recursive"`).
enabled	Boolean indicating if the workflow is active (`true` or `false`).
visible	Boolean indicating if the workflow is visible in the UI (`true` or `false`).
fields	List of input fields required when running the workflow (e.g., `lock_level` choice field).
filter_rules	Rules defining file types, states, and actions performed (e.g., migrate task on all files).
discovery_options	Additional settings for discovery (e.g., `allow_missing_paths`).
preferred_queue	Optional preferred queue name for running tasks (`null` if not set).

POST /workflows

Description

This API endpoint creates a new workflow in the system. It allows you to define a workflow by specifying various parameters such as the name, label, discovery type,

schedule, and other custom fields related to the workflow. This is useful for automating tasks across sites.

Request Body Fields

Name	Description	Type	Required	Default Value
name	The internal name for the workflow.	string	Yes	N/A
label	The display label for the workflow.	string	Yes	N/A
icon_classes	List of icon names associated with the workflow.	array	No	`[]`
discovery	The discovery method to be used for the workflow (e.g., `"snapdiff"`).	string	Yes	N/A
enabled	Whether the workflow is enabled.	boolean	Yes	`true`
visible	Whether the workflow is visible in the system.	boolean	Yes	`true`
schedule_only	Whether the workflow can only be scheduled.	boolean	Yes	`false`
external_only	Whether the workflow is only applicable to external targets.	boolean	Yes	`false`
fields	A list of custom fields to include in the workflow (e.g., target site, etc.).	array	No	`[]`
filter_rules	Filter rules that define the actions for the workflow.	array	No	`[]`
discovery_options	Additional options for discovery (currently empty by default).	object	No	`{}`
preferred_queue	The preferred queue to which the workflow should be assigned.	string	No	N/A

Example Request

```
{
  "name": "transfer_files",
  "label": "Transfer files between sites",
  "icon_classes": [],
  "discovery": "snapdiff",
  "enabled": true,
  "visible": true,
  "schedule_only": false,
  "external_only": false,
```

```
  "fields": [
    {
      "name": "target_site",
      "type": "enum[site]",
      "label": "Destination Site",
      "default": "london"
    }
  ],
  "filter_rules": [
    {
      "type": "all",
      "state": "all",
      "action": [
        {
          "name": "dynamo.tasks.migrate",
          "site": "*target_site"
        }
      ]
    }
  ],
  "discovery_options": {},
  "preferred_queue": "highpriority"
}
```

Successful Response

HTTP Status Code: `201 Created`

Content-Type: `application/json`

Example Response

```
{
    "id": 19,
    "name": "transfer_files",
    "label": "Transfer files between sites",
    "icon_classes": [],
    "discovery": "snapdiff",
    "enabled": true,
    "visible": true,
    "schedule_only": false,
    "external_only": false,
    "fields": [
        {
            "name": "target_site",
            "type": "enum[site]",
            "label": "Destination Site",
            "default": "london"
        }
    ],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
```

```
            "action": [
                {
                    "name": "dynamo.tasks.migrate",
                    "site": "*target_site"
                }
            ]
        }
    ],
    "discovery_options": {},
    "preferred_queue": "highpriority"
}
```

Response Fields

Please refer to the response fields under `GET`**/workflows**.

GET /workflows/{id}/

Description

This API endpoint retrieves detailed information about a specific workflow using its unique ID. It provides details such as the workflow's name, label, fields, filter rules, and more. This is useful for viewing or reviewing the configurations of existing workflows.

Request Parameters

Name	Description	Type	Required	Default Value
id	The unique ID of the workflow to be viewed.	string	Yes	N/A

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/workflows/7/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
    "id": 7,
    "name": "reverse_stub",
    "label": "Reverse Stub",
    "icon_classes": [],
    "discovery": null,
```

```
    "enabled": true,
    "visible": false,
    "schedule_only": false,
    "external_only": false,
    "fields": [
        {
            "name": "hydrate",
            "type": "bool",
            "label": "Hydrate files on site"
        },
        {
            "name": "overwrite",
            "type": "bool",
            "label": "Overwrite files on site"
        }
    ],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.tasks.reverse_stub",
                    "hydrate": "*hydrate",
                    "overwrite": "*overwrite"
                }
            ]
        }
    ],
    "discovery_options": {},
    "preferred_queue": null
}
```


Response Fields

Please refer to the response fields under `GET`**/workflows**.

PATCH /workflows/{id}/

Description

This endpoint allows partial updates to an existing workflow by its ID. It is used to modify editable fields of the workflow such as label, visibility, discovery method, schedule settings, and other workflow-specific configuration details.

Request Parameters

Parameter	Location	Type	Required	Description
id	path	string	Yes	The unique ID of the workflow to update.
data	body	object	Yes	JSON object containing updatable fields.

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/workflows/1/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>' \
  -H 'Content-Type: application/json' \
  -d '{
  "label": "Dehydrate_updated",
  "icon_classes": [],
  "discovery": "recursive",
  "enabled": true,
  "visible": true,
  "schedule_only": false
}'
```

Request Body Fields

Only the following fields are allowed for updates:

Field	Type	Description
label	string	User-friendly label for the workflow.
icon_classes	array	List of CSS icon classes associated with the workflow.
discovery	string	Discovery mode (e.g., `"recursive"`).
enabled	boolean	Whether the workflow is enabled.
visible	boolean	Controls if the workflow is visible in the UI.
schedule_only	boolean	Whether this workflow is only used for scheduling.
fields	array	List of field definitions (e.g., user inputs).
filter_rules	array	Rules that define workflow behavior and actions.
discovery_options	object	Options related to discovery behavior.
preferred_queue	string/ null	Optional field to set the preferred task queue.

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
    "id": 1,
    "name": "migrate",
    "label": "Dehydrate_updated",
    "icon_classes": [],
    "discovery": "recursive",
    "enabled": true,
    "visible": true,
    "schedule_only": false,
```

```
    "fields": [
        {
            "name": "lock_level",
            "type": "choices",
            "label": "Locking mode",
            "choices": [
                {
                    "label": "Per block (Implicit)",
                    "value": "implicit"
                }
            ],
            "default": "implicit"
        }
    ],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.tasks.migrate",
                    "lock_level": "*lock_level"
                }
            ],
            "description": "Migrates a file off from a given path"
        }
    ],
    "discovery_options": {
        "allow_missing_paths": false
    },
    "preferred_queue": null
}
```


Response Fields

Please refer to the response fields under `GET`**/workflows**.

DELETE /workflows/{id}/

Description

This API endpoint deletes a workflow identified by its unique ID. The workflow will be permanently removed from the system, and the operation is irreversible. Use this endpoint to clean up workflows that are no longer needed.

Request Parameters

Name	Description	Type	Required	Default Value
id	The unique ID of the workflow to be deleted.	string	Yes	N/A

Successful Response

HTTP Status Code: `204 No Content`

Content-Type: `application/json`

Response Body: No content is returned in the response body.

There are no fields returned in the body when the deletion is successful.

Automated Schedules

GET /api/automatedschedules/

Description

This API endpoint retrieves a list of automated schedules that are configured in the system. Each schedule defines when and where a specific policy or workflow job should automatically run.

Users can view schedules and policies under Automated Schedules.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X GET \
  'http://your-hub-address:8000/api/automatedschedules/?
page=1&page_size=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
```

```
            "name": "Test1",
            "schedule_type": "workflow",
            "enabled": true,
            "site_name": "siteB",
            "spaces": [
                {
                    "id": 1,
                    "name": "mmfs1"
                }
            ],
            "discovery_name": "Test1",
            "discovery_id": 1,
            "discovery_managed_paths": ["/mmfs1"],
            "discovery_workflow": "Dehydrate",
            "minute": "0",
            "hour": "0",
            "day_of_week": "*",
            "day_of_month": "*",
            "month_of_year": "*"
        }
    ]
}
```


Response Fields

Top-Level Pagination Fields

Field	Type	Description
count	integer	Total number of schedules available.
next	string	URL for the next page of results, or null.
previous	string	URL for the previous page of results, or null.
results	array	List of schedule objects for the current page.

Inside results[ ]

Field	Description
`name`	Name of the schedule
`schedule_type`	Type of schedule (e.g., `workflow`)
`enabled`	Indicates if the schedule is active (`true`) or not (`false`)
`site_name`	Name of the site where the schedule is applied
`spaces`	List of space objects with `id` and `name`
`discovery_name`	Name of the discovery job associated with the schedule
`discovery_id`	ID of the associated discovery job
`discovery_managed_paths`	List of paths being managed by the discovery job
`discovery_workflow`	Name of the workflow triggered by the schedule
`minute`	Minute value in cron format
`hour`	Hour value in cron format
`day_of_week`	Day of the week in cron format (`*` means every day)
`day_of_month`	Day of the month in cron format (`*` means every day)

Field	Description
`month_of_year`	Month of the year in cron format (`*` means every month)

Tasks

Custom tasks in Ngenea Worker plugins allow users to define their own task logic to be used in custom workflows, alongside predefined tasks. These tasks must follow a specific structure, starting with accepting standard keyword arguments such as `jobid`, which identifies the job the task belongs to, and paths, which is a list of file or folder `paths` to be processed. Additional parameters can be included as needed, and tasks should also support arbitrary keyword arguments via `**kwargs`.

Tasks are required to return a structured payload containing at least the job ID, the list of resulting paths, and a `status` dictionary that includes metadata about the task execution, such as the task name, the original input paths, the number of input files, a summary of file statuses (e.g., processed, skipped, failed), detailed results for each file, and a timestamp of when the task started. To simplify this, developers can use helper classes like `FileActionStatus` to build the payload, or `StreamTaskResults` for streaming scenarios.

Read More ..

GET/ tasks/

Description

This endpoint retrieves information about tasks in the system. You can filter the results based on various parameters, such as task state, job ID, task type, or task ID. *If no state is specified*, the API returns *tasks in all states* by default. This endpoint is useful for monitoring and managing tasks based on their execution status, such as `SUCCESS`, `FAILURE`, `STARTED` and more.

Request Parameters

Name	Type	Description
page	integer	Page number within the paginated results. Defaults to the first page if not specified.
page_size	integer	Number of tasks per page (20–100). Use 0 to disable pagination and retrieve all results.
tasktype	string	Filter by task type. Max length: 50 characters.
task_id	string	Filter by specific task ID. Max length: 100 characters.
state	array[string]	Filter by one or more task states (SUCCESS, FAILURE, STARTED, etc.).
job_id	integer	Filter by the job ID associated with the task. Minimum value: 1.
internal	string	Set to “false” to exclude internal system tasks.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/tasks/?
state=SUCCESS&internal=false' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

This example fetches all tasks with the `SUCCESS` state and excludes internal system tasks.

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
    "count": 242,
    "next": "http://your-hub-address:8000/api/tasks/?
internal=false&page=2&state=SUCCESS",
    "previous": null,
    "results": [
        {
            "url": "http://your-hub-address:8000/api/tasks/2164/",
            "id": 2164,
            "task_id": "f856aada-8d80-412d-af1d-38163937c2a7",
            "tasktype": "dynamo.tasks.recursive_navigate",
            "state": "SUCCESS",
            "job": 191,
            "site": "siteB",
            "started": "2025-08-21T00:00:01.702996Z"
        }
    ],
    "stats": {
        "state": {},
        "tasktype": {}
    }
}
```


Note: Only a single result is shown above for example purposes.

Response Fields

Field	Type	Description
count	integer	Total number of matching tasks.
next	string or null	URL to the next page of results, if any.
previous	string or null	URL to the previous page of results, if any.
results	array[object]	List of task objects. Each task includes:

Task Object Fields (inside `results`):

Field	Type	Description
→ url	string	API endpoint for the specific task.
→ id	integer	Internal task ID.
→ task_id	string	UUID of the task (Celery task ID).
→ tasktype	string	Type of the task (e.g., `dynamo.tasks.recursive_navigate`).
→ state	string	Current state of the task (`SUCCESS`, `FAILURE`, etc.).
→ job	integer	Associated job ID.
→ site	string	Site where the task ran.
→ started	datetime	When the task started (ISO 8601 format).
stats	object	Aggregated stats (empty in the example).

GET /tasks/{id}/

Description

This endpoint retrieves detailed information about a specific task by its unique ID. It returns task metadata such as the task type, state, start and completion timestamps, associated job, site, runtime, and a comprehensive result breakdown. This endpoint is useful for analyzing individual task execution, including its input/output paths, file and directory counts, and any errors or skipped operations.

Request Parameters

Name	Type	Location	Required	Description
`id`	string	path	Yes	The unique identifier of the node you want to retrieve.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/tasks/2164/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
    "url": "http://your-hub-address:8000/api/tasks/2164/",
    "id": 2164,
    "task_id": "f856aada-8d80-412d-af1d-38163937c2a7",
    "tasktype": "dynamo.tasks.recursive_navigate",
    "state": "SUCCESS",
    "started": "2025-08-21T00:00:01.702996Z",
    "completed": "2025-08-21T00:00:02.487565Z",
    "runtime": 0.784569,
```

```
    "job": 191,
    "site": "siteB",
    "paths": "http://your-hub-address:8000/api/tasks/2164/files/",
    "result": {
        "log": [],
        "skip": false,
        "jobid": 191,
        "paths": ["examples will be shown"],
        "queue": "siteB",
        "status": {
            "task": "dynamo.tasks.recursive_navigate",
            "details": {
                "dirs": ["examples will be shown"],
                "files": ["examples will be shown"],
                "aborted": [],
                "failures": [],
                "symlinks": [],
                "empty_dirs": ["examples will be shown"]
            },
            "started": "2025-08-21T00:00:01.231687+00:00",
            "summary": {
                "dirs": 84,
                "files": 26,
                "aborted": 0,
                "failures": 0,
                "symlinks": 0,
                "empty_dirs": 260
            },
            "input_paths": ["examples will be shown"],
            "input_total": 12,
            "has_dynamic_children": true
        }
    },
    "friendly_name": null
}
```


Response Fields

Field	Type	Description
url	string	API URL for the task object.
id	integer	Internal task ID.
task_id	string	UUID representing the Celery task.
tasktype	string	Type of task executed.
state	string	Status of the task (e.g., SUCCESS, FAILURE, etc.).
started	string	ISO timestamp for when the task started.
completed	string	ISO timestamp for when the task completed.
runtime	float	Execution duration in seconds.
job	integer	Associated job ID.
site	string	Site or environment where the task was executed.
paths	string	Link to fetch associated file paths.

Field	Type	Description
result	object	Detailed output of the task including logs, paths, summary statistics, and directory/file breakdowns.
friendly_name	string or null	Optional user-defined name for the task.

GET /tasks/{id}/files/

Description

Retrieves the list of files and directories associated with a specific task. Supports optional query parameters to *filter results by status*, *type*, and *site*. If no filters are provided, all available entries are returned (subject to pagination).

Request Parameters

Name	Type	Location	Required	Description
`id`	string	path	Yes	The unique identifier of the node you want to retrieve.

Query Parameters

Name	Type	Required	Description
page	integer	No	Page number for pagination (default: 1).
page_size	integer	No	Number of results per page. Range: 20–1000. Use 0 to disable pagination and return all.
state	string	No	Filter entries by their processing state (e.g., processed, skipped).
type	string	No	Filter by file type (if applicable).
site	string	No	Filter by site (if multiple sites are in use).

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/tasks/2164/files/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
```

```
        {
            "path": "/mmfs1/data/Project1/goodbye",
            "state": "processed",
            "type": null,
            "size": null
        },
        {
            "path": "/mmfs1/data/Project1/foo",
            "state": "processed",
            "type": null,
            "size": null
        },
        {
            "path": "/mmfs1/data/Project1",
            "state": "skipped",
            "type": null,
            "size": null
        }
    ]
}
```


Response Fields

Field	Type	Description
count	integer	Total number of matching entries.
next	string or null	URL to next page (if paginated).
previous	string or null	URL to previous page (if paginated).
results	array	List of file/directory entries.

Each item in `results` contains:

Field	Type	Description
path	string	Absolute path of the file or directory.
state	string	Processing status (e.g., processed, skipped).
type	string or null	Type of the entry (e.g., file, directory).
size	integer or null	Size in bytes (if applicable).

Queues

Ngenea hub uses `job queues` to organize and manage task execution efficiently. Each queue acts as a processing line where jobs wait for available worker threads to run them.

Why Use Custom Queues ?

- Prioritize important tasks (e.g., high-priority jobs).
- Isolate specific types of jobs (e.g., Transparent Recall).
- Control how many jobs run simultaneously using threads.

Key Features

- `Default Queue`: Always available. Runs all task types, including interactive and settings.
- `Custom Queues`: User-defined. Only support worker, discovery, and custom functions.
- `Per-Function Thread Limits`: Fine-tune how many threads each function uses in a queue.
- `Disable Functions`: Prevent specific task types from running in a queue.
- `Automatic Discovery`: Queues are automatically started/stopped based on config changes.

Configuration File: Queues are defined in `/etc/ngenea/ngenea-worker.conf`.

Memory Notes: Each thread uses ~40MB of RAM idle, and up to 125MB depending on functions configured. More threads = more memory use.

Removing Queues: Removing a queue from config stops it. It must be manually de-registered using: `ngeneahubctl manage remove_queue <site> <queue>`. To re-use a name, restart the worker and use `ngeneahubctl manage remove_queue <site> <queue> --offline-only`.

Read More..

GET /queues/

Description

Retrieves a list of active job queues in the system, grouped by site. Each queue includes details such as the site it belongs to, nodes handling the queue, its name, and the timestamp of its last heartbeat (i.e., when it was last active).

This is useful for monitoring which queues are currently running across different sites in your cluster.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/queues/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "site": {
                "id": 1,
                "name": "siteA",
                "label": "siteA"
            },
            "nodes": [
                {
                    "id": 1,
                    "name": "siteA-1"
                }
            ],
            "name": "default",
            "label": "default",
            "last_heartbeat": "2025-09-02T14:23:27.048619Z"
        },
        {
            "id": 2,
            "site": {
                "id": 2,
                "name": "siteB",
                "label": "siteB"
            },
            "nodes": [
                {
                    "id": 2,
                    "name": "sitetest"
                },
                {
                    "id": 3,
                    "name": "siteB-1"
                }
            ],
            "name": "default",
            "label": "default",
            "last_heartbeat": "2025-09-02T14:23:28.171405Z"
        }
    ]
}
```

Response Fields

Field	Type	Description
count	integer	Total number of queues.
next	string | null	URL to the next page of results, if paginated.
previous	string | null	URL to the previous page of results, if paginated.
results	array	List of queue objects.

Inside `results` array:

Field	Type	Description
results[].id	integer	ID of the queue.
results[].site	object	Site where the queue is running.

Inside `results[].site`:

Field	Type	Description
site.id	integer	ID of the site.
site.name	string	Name of the site.
site.label	string	Display label of site.
results[].nodes	array	Nodes running this queue.

Inside `results[].nodes`:

Field	Type	Description
nodes[].id	integer	ID of the node.
nodes[].name	string	Name of the node.
results[].name	string	Name of the queue (e.g., default).
results[].label	string	Display label of the queue.
last_heartbeat	string (ISO 8601)	Timestamp of the last queue activity (heartbeat) in UTC.

GET /queues/{id}/

Description

Fetch detailed information about a specific job queue using its unique ID. This includes: the site it belongs to, the nodes it’s running on, queue name and label, the last time it sent a heartbeat (used to check if the queue is alive).

Useful for queue monitoring, debugging, or administrative checks on a particular queue.

Request Parameters

Name	Type	Location	Required	Description
`id`	string	path	Yes	The unique identifier of the node you want to retrieve.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/queues/2/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
    "id": 2,
    "site": {
        "id": 2,
        "name": "siteB",
        "label": "siteB"
    },
    "nodes": [
        {
            "id": 2,
            "name": "sitetest"
        },
        {
            "id": 3,
            "name": "siteB-1"
        }
    ],
    "name": "default",
    "label": "default",
    "last_heartbeat": "2025-09-02T14:39:37.163821Z"
}
```


Response Fields

Field	Type	Description
id	integer	Unique identifier of the queue.
site	object	Details of the site this queue belongs to.

Inside `site` object:

Field	Type	Description
site.id	integer	Unique ID of the site.
site.name	string	Internal name of the site.
site.label	string	Human-readable label of the site.
nodes	array	List of nodes where this queue is active.

Inside `nodes[]` array:

Field	Type	Description
nodes[ ].id	integer	ID of the node.
nodes[ ].name	string	Name of the node.
name	string	Internal name of the queue (e.g., default).
label	string	Human-readable label of the queue.
last_heartbeat	string (ISO 8601)	Last time this queue was active (UTC time).

API Endpoints - Monitoring and Analytics

Alerts

GET /alerts/

Description

Retrieves a list of active, non-muted alerts from the Ngenea Hub system.

This endpoint is primarily used by the Hub UI to populate both **Global** and **Site-specific** alert dashboards. The API supports filtering by multiple parameters such as `name`, `site`, `node`, `severity`, and more.

Note:

- All parameters in the Swagger UI for this endpoint are **optional**.
- The response will be **filtered based on the parameters you provide** (e.g., filtering by `name`, `site`, or `severity`).
- If no parameters are specified, the API will return **all active and non-muted alerts by default**.

Request Parameters

Name	Type	Location	Description
`name`	string	query	Filter alerts by name (e.g., `SiteHealth`)
`site`	string	query	Filter alerts by site name
`node`	string	query	Filter alerts by node name
`severity`	string	query	Filter alerts by severity (`INFO`, `WARNING`, `CRITICAL`)
`summary`	string	query	Filter alerts by summary text
`timestamp`	string	query	Filter alerts by timestamp (ISO 8601 format)
`suppressed`	string	query	Filter alerts by suppression status

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/alerts/?name=SiteHealth' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
[
  {
    "id": 99,
    "site": "testexample",
    "name": "SiteHealth",
    "node": null,
    "queue": null,
    "created": "2025-07-03T14:33:00.031598Z",
    "timestamp": "2025-08-07T12:16:00.018308Z",
    "severity": "INFO",
    "summary": "No workers are joined",
    "suppressed": false
  }
]
```


Response Fields

Field	Type	Description
`id`	integer	Unique ID of the alert
`site`	string	Name of the site where the alert occurred
`name`	string	Alert name/type (e.g., `SiteHealth`)
`node`	string	Node name if applicable
`queue`	string	Queue name if applicable
`created`	string	ISO timestamp when alert was created
`timestamp`	string	ISO timestamp of the most recent alert event
`severity`	string	Alert severity (`INFO`, `WARNING`, `CRITICAL`)
`summary`	string	Short summary of the alert
`suppressed`	boolean	Whether the alert is currently suppressed

GET /alerts/{id}/

Description

Fetch detailed information about a **specific alert** using its unique ID. This endpoint returns the full alert object, including metadata such as severity, timestamps, and associated site or node.

Used by the Hub system to display detailed alert information in dashboards or drill-down views.

Request Parameters

Name	Type	Location	Description
`id`	integer	path	A unique integer identifying the alert. (**Required**)

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/alerts/99/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "id": 99,
  "site": "testexample",
  "name": "SiteHealth",
  "node": null,
  "queue": null,
  "created": "2025-07-03T14:33:00.031598Z",
  "timestamp": "2025-08-07T12:28:00.014935Z",
  "severity": "INFO",
  "summary": "No workers are joined",
  "suppressed": false
}
```

Response Fields

Please refer to the response feilds under `GET /alerts/`.

Health

GET /health/

The `GET /health` endpoint provides a real-time overview of the health status of the Ngenea Hub infrastructure components.

This endpoint is used to monitor:

- The health of the **Hub database** (PostgreSQL)
- The health of the **Hub task queue**
- The health of **Redis**

The system reports health using the following states:

- `working`: The service is online and Ngenea Hub can make use of it

- `unavailable`: The service could not be contacted or reported an error. The response will detail the specific error.

By querying this endpoint, administrators can quickly assess which parts of the infrastructure are healthy and identify potential issues. This is useful for operational monitoring, alerting, and overall system visibility. It can also be used to decide when to failover a Hub instance in a configuration with multiple Hub instances.

Request Parameters

This endpoint **does not require any parameters**. Simply calling the endpoint returns the health status of the Hub components.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/health/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
    "CeleryPingHealthCheck": "working",
    "DatabaseBackend": "working",
    "RabbitMQHealthCheck": "working",
    "RedisHealthCheck": "unavailable: Unable to connect to Redis:
Connection Error"
}
```

Field	Type	Description
`CeleryPingHealthCheck`	string	Status of the task queue (Celery)
`DatabaseBackend`	string	Status of the Hub database (PostgreSQL)
`RabbitMQHealthCheck`	string	Status of RabbitMQ if used as the task queue broker
`RedisHealthCheck`	string	Status of Redis

GET /health/sites/

Description

The `GET /health/sites` endpoint provides a real-time overview of the health status of the Ngenea Hub and all connected sites.

This endpoint is used to monitor:

- The health of the **Hub service** itself
- All **registered sites**
- The **nodes** (servers/machines) within each site
- Any associated queues

The system reports health using the following states:

- `ok`: All nodes in the site are functioning normally
- `warning`: Some nodes are offline or unresponsive
- `critical`: An entire site is offline (no working nodes)

By querying this endpoint, administrators can quickly assess which parts of the infrastructure are healthy and identify potential issues at the site or node level. This is useful for operational monitoring, alerting, and overall system visibility.

Request Parameters

This endpoint **does not require any parameters**. Simply calling the endpoint returns the health status of the Hub and all registered Sites.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/health/sites/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "overall_health": "ok",
  "hub_status": {
    "health": "ok"
  },
  "site_status": [
    {
      "site": "Site 1",
      "health": "ok",
      "nodes": [
        {
          "name": "Site 1",
          "health": "ok",
          "online": true,
          "has_default_queue": true,
          "last_heartbeat": "2025-08-07T13:34:23.276099+00:00",
          "worker_version": "2.6.0",
```

```
          "ngenea_version": "1.31.0",
          "analytics_version": "2.7.1",
          "search_version": "1.2.0"
        }
      ],
      "queues": [
        {
          "name": "default",
          "health": "ok",
          "online": true,
          "last_heartbeat": "2025-08-07T13:34:23.276099+00:00"
        }
      ]
    },
    {
      "site": "Site 2",
      "health": "ok",
      "nodes": [
        {
          "name": "Site 1",
          "health": "ok",
          "online": true,
          "has_default_queue": true,
          "last_heartbeat": "2025-08-07T13:34:24.781445+00:00",
          "worker_version": "2.6.0",
          "ngenea_version": "1.31.0",
          "analytics_version": "2.7.1",
          "search_version": "1.2.0"
        },
        {
          "name": "sitetest",
          "health": "ok",
          "online": true,
          "has_default_queue": true,
          "last_heartbeat": "2025-07-01T10:54:54.269000+00:00",
          "version_details": "Could not retrieve version
information for example@sitetest"
        }
      ],
      "queues": [
        {
          "name": "default",
          "health": "ok",
          "online": true,
          "last_heartbeat": "2025-08-07T13:34:34.791954+00:00"
        }
      ]
    },
    {
      "site": "testexample",
      "health": "ok",
      "nodes": [],
      "queues": []
    }
```

```
  ]
}
```

Response Fields

Field	Type	Description
`overall_health`	string	Overall system health status (`ok`, `warning`, `error`, etc.)
`hub_status.health`	string	Health status of the central Hub component
`site_status`	array	List of health status entries per site
`site_status[].site`	string	Name of the site
`site_status[].health`	string	Health of the site (`ok`, etc.)
`site_status[].nodes`	array	List of nodes for that site
`site_status[].nodes[].name`	string	Node name
`site_status[].nodes[].health`	string	Health of the node
`site_status[].nodes[].online`	boolean	Whether the node is online
`site_status[].nodes[].has_default_queue`	boolean	Whether the node has the default queue
`site_status[].nodes[].last_heartbeat`	string	Timestamp of last node heartbeat
`site_status[].nodes[].worker_version`	string	Worker component version (if available)
`site_status[].nodes[].ngenea_version`	string	Ngenea service version (if available)
`site_status[].nodes[].analytics_version`	string	Analytics service version
`site_status[].nodes[].search_version`	string	Search service version
`site_status[].nodes[].version_details`	string	Additional version info or errors (optional)
`site_status[].queues`	array	List of queue objects for the site
`site_status[].queues[].name`	string	Name of the queue
`site_status[].queues[].health`	string	Health of the queue
`site_status[].queues[].online`	boolean	Whether the queue is online
`site_status[].queues[].last_heartbeat`	string	Timestamp of last queue heartbeat

Search

Ngenea Hub's `Search` feature allows users to quickly locate files and folders across all connected sites, making it easier to manage distributed storage environments. Built on top of **PixStor**, the search functionality supports both **PixStor Search** and **PixStor Analytics** backends.

Before using the search APIs, it's important to ensure the appropriate backend is configured and that the system points to the correct instance (based on your PixStor version).

There are two ways to interact with the search capability:

- **Via REST API** : Ideal for developers or automated tools needing programmatic access
- **Via UI** : A user-friendly global search page within the Ngenea Hub web interface

Depending on the setup, users can switch between search backends (e.g., from Analytics to PixStor Search) using configuration API calls. The actual data source (Elasticsearch) must also be correctly pointed to, especially for systems running **Pix Stor 6+**.

With the proper configuration, Ngenea Search enables efficient, site-wide or global file discovery with minimal effort.

> **Note:** When the search backend is configured as analytics, metadata updates are not supported.
>
> To switch to the PixStor Search backend, please refer to the Search documentation.

POST /search/

Description

Initiates a **file or folder search** across one or more sites managed by the Ngenea Hub. This endpoint allows users to search for files within a specified path using optional filters and settings.

The request body includes search criteria such as:

- The target site(s)
- Path to search from
- Filters (e.g., metadata)
- Whether to search recursively
- Whether to merge results from multiple sites

Once triggered, the search operation is submitted, and a unique search ID is returned.

Request Parameters

Name	Type	Location	Description
`data`	object	body	JSON object containing search criteria (**Required**)

Example Request

```
curl -X 'POST' \
  'http://your-hub-address:8000/api/search/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '{
  "sites": [
    "siteA"
  ],
  "path": "/path/to/file",
  "filters": {},
  "merge": false,
  "metadata_fields": [],
  "recursive": true
}'
```

Request Body Fields

Field	Type	Required	Description
`sites`	array	Yes	List of site names to search in
`path`	string	Yes	Root path where the search should begin
`filters`	object	No	Optional key-value filters (e.g., by metadata attributes)
`merge`	boolean	No	Whether to merge results from multiple sites (default: `false`)
`metadata_fields`	array	No	List of specific metadata fields to include in the response
`recursive`	boolean	No	Whether to search recursively through subdirectories (default: `true`)

Successful Response

HTTP Status Code: `201 Created`

Content-Type: `application/json`

Returns a unique ID for the search task and a URL to retrieve results.

Example Response

```
{
  "id": 21,
  "url": "http://your-hub-address:8000/api/search/21/"
}
```

Response Fields

Field	Type	Description
`id`	integer	Unique identifier for the search job
`url`	string	URL to access the status or results of the search

PATCH /search/metadata/

Description

This endpoint is used to **set or update metadata tags on files or folders** in the Ngenea Hub search index. These metadata updates help improve search relevance and filtering by associating custom tags or attributes with files.

> **Note:** This endpoint is primarily used for setting tags on files within the search system.
>
> For detailed information on how tags work and how to use them effectively, please refer to the **Tags** section in the Search documentation.

Request Parameters

Name	Type	Location	Description
`data`	array	body	Array of objects containing file path, site, and metadata updates (**Required**)

Example Request

```
curl -X 'PATCH' \
  'http://your-hub-address:8000/api/search/metadata/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key' \
  -H 'Content-Type: application/json' \
  -d '[
{
    "pathname": "/path/to/file",
    "site": "site A",
    "metadata": {
        "search.tags": ["example", "two", "three"]
    }
}
]'
```

Request Body Fields

Field	Type	Required	Description
`pathname`	string	Yes	The file or folder path where metadata is updated
`site`	string	Yes	Site name where the file/folder resides
`metadata`	object	Yes	Key-value pairs representing metadata updates (e.g., tags)

Successful Response

HTTP Status Code: `201 Created`

Content-Type: `application/json`

Indicates the metadata updates have been accepted and will be applied asynchronously.

Example Response

```
{
  "message": "updates accepted, they will be applied
asynchronously"
}
```

Response Fields

Field	Type	Description
`message`	string	Confirmation that updates have been accepted for asynchronous processing

GET /search/metadata_fields/

Description

This endpoint retrieves a list of available metadata fields that can be used for `searching` or `filtering` files and folders across one or more sites in Ngenea Hub.

These metadata fields describe file attributes (e.g., size, owner, mimetype), file system details (e.g., GPFS fields), and custom fields like `search.tags`.

Request Parameters

Name	Type	Required	Description
`page`	integer	No	Page number of the result set. Defaults to 1 if not specified.
`page_size`	integer	No	Number of results to return per page. Must be between 20 and 100. Use 0 to return all results without pagination. Defaults to 20 if not specified or below the minimum.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/search/metadata_fields/?
page=1&page_size=1' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response

```
{
  "siteA": [
    {
      "name": "core.accesstime",
      "type": "datetime"
    },
    {
      "name": "core.blocksize",
      "type": "long"
    },

    {
      "name": "search.tags",
      "type": "set"
    }
  ],
  "siteB": [
    {
      "name": "core.accesstime",
      "type": "datetime"
    },

    {
      "name": "search.tags",
      "type": "set"
    }
  ]
}
```


Response Fields

Field	Type	Description
`site_name`	string	Name of the site (e.g., `siteA`, `siteB`)
`name`	string	Name of the metadata field
`type`	string	Data type (e.g., `string`, `integer`, `set`)

GET /search/{id}/

Description

This endpoint retrieves the **results of a previously initiated search operation** using the search ID returned from the `POST /search/` request.

It supports pagination, sorting, and optional grouping of results across multiple sites. This endpoint is useful for retrieving large result sets in a controlled, page-by-page manner or for merging matching file paths by name across different sites.

Request Parameters

Name	Type	In	Description
`id`	string	path	(**Required**) Unique ID of the search operation.

Name	Type	In	Description
`page`	integer	query	Page number within the paginated result set. Defaults to the first page if not provided.
`page_size`	integer	query	Number of results per page.
`sort`	string	query	One or more metadata fields to sort the results by.
`group_by_name`	boolean	query	If true, groups results by matching file names across multiple sites. Useful for comparing file presence and metadata consistency.

Example Request

```
curl -X 'GET' \
  'http://your-hub-address:8000/api/search/21/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "count": 1,
  "next": null,
  "previous": null,
  "id": 34,
  "parameters": {
    "path": "/",
    "sites": ["siteA"],
    "filters": {
      "core.pathname": "*helloworld*"
    }
  },
  "more_results": false,
  "items": [
    {
      "href": "http://your-hub-address:8000/api/file/?
path=%2Fmmfs1%2Fdata%2Fsample_data%2Fhelloworld.txt&site=siteA",
      "site": "siteA",
      "path": "/mmfs1/data/sample_data",
      "name": "helloworld.txt",
      "metadata": {
        "core.size": 116,
        "core.type": "file",
        "core.inode": 30984,
        "core.user.name": "root",
        "core.group.name": "root",
        "core.pathname": "/mmfs1/data/sample_data/helloworld.txt",
```

```
        "core.modificationtime": "2017-05-26T14:26:28",
        "core.changetime": "2025-05-14T11:57:08",
        "gpfs.poolname": "sas1",
        "gpfs.filesetname": "root"
      },
      "proxies": {}
    }
  ],
  "complete": false,
  "errors": {}
}
```


Response Fields

Field	Type	Description
`count`	integer	Total number of matching items returned.
`next` / `previous`	string/ null	Pagination links for next/previous result pages, if available.
`id`	integer	Unique request or search ID, useful for tracking and debugging.
`parameters.path`	string	Root path used in the search query.
`parameters.sites`	array	List of site names where the search was performed.
`parameters.filters`	object	Filters used in the search query (e.g., pathname match patterns).
`more_results`	boolean	Indicates whether more results are available beyond the current response.
`items`	array	Array of matching file entries from the search.
`items[].site`	string	Site where the file was found.
`items[].path`	string	Directory path where the file is located.
`items[].name`	string	File name.
`items[].href`	string	API link to fetch file metadata or access the file.
`items[].metadata`	object	Key-value pairs containing detailed metadata about the file.
`complete`	boolean	Indicates whether the search has completed across all queried sites.
`errors`	object	Any site-specific errors encountered during the search (empty if none).

DELETE /search/{id}/

Description

This endpoint is used to **delete a previously created search operation** and its associated results from the system. It is typically used to clean up resources once the search is complete and no longer needed.

Request Parameters

Name	Type	In	Description
`id`	string	path	(**Required**) The unique search ID to be deleted.

Example Request

```
curl -X 'DELETE' \
  'http://your-hub-address:8000/api/search/21/' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key your-api-key'
```

Successful Response

HTTP Status Code: `204 No Content`

The search and its results were successfully deleted. No content is returned in the response body.

Response Fields

No response body is returned for this request.

This action is irreversible. Once a search is deleted, its results and metadata are permanently removed and cannot be retrieved again. Be sure you no longer need the data before performing this operation.

Analytics

GET /analytics

Description

The `GET /analytics/` endpoint retrieves analytics data related to file storage usage for a specified site and directory path. It provides information such as the total number of files, total bytes used, and per-directory or per-storage pool statistics. The API supports optional parameters like pagination, child directory inclusion, CSV downloads, and customizable timeout or cache settings.

This API is used when there is a need to understand how storage is being utilized within a site. It is particularly useful for analyzing a specific directory or for retrieving detailed data across child directories.

Administrators can call this endpoint periodically or on demand to review current storage metrics.

It is useful for storage monitoring, capacity planning, and troubleshooting. By analyzing file count and storage consumption, teams can optimize storage usage, detect anomalies or growth patterns, and generate reports for internal audits or compliance purposes. The option to export data as CSV makes it convenient for external reporting or integration with other tools.

Request Parameters

Name	Type	Required	Description
`site`	string	Yes	The name of the site to retrieve analytics for. Minimum length: 1.
`path`	string	No	The target directory path to analyze. If not provided, analytics will be retrieved for all space roots.
`page`	integer	No	Page number within the paginated result set. Defaults to 1 if not provided.
`page_size`	integer	No	Number of results per page. Must be between 20 and 100. Use `0` to disable pagination and retrieve all results. Defaults to 20 if not specified or too low.
`children`	boolean	No	Whether to include child directories in the response. Default is `false`.
`cache_ttl`	integer	No	Optional cache time-to-live for the analytics data (in seconds).
`custom_timeout`	integer	No	Custom timeout (in seconds) for fetching analytics data.
`download`	string	No	If set to `csv`, returns the result as a downloadable CSV file.

Example Request

```
curl -X GET \
  'http://your-hub-address:8000/api/analytics/?
page=1&site=siteB&children=true' \
  -H 'accept: application/json' \
  -H 'Authorization: Api-Key <your-api-key>'
```

Successful Response

HTTP Status Code: `200 OK`

Content-Type: `application/json`

Example Response


```
{
  "children": {
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
      {
        "site": "siteB",
        "pools": [
          {
            "name": "sas1",
            "total_used_bytes": 475529216,
            "total_files_count": 532
          },
```

```
          {
            "name": "cloud",
            "total_used_bytes": 0,
            "total_files_count": 0
          }
        ],
        "path": "/mmfs1",
        "total_used_bytes": 475529216,
        "total_files_count": 532
      }
    ]
  },
  "site": "siteB",
  "pools": [],
  "path": "",
  "total_used_bytes": null,
  "total_files_count": null
}
```


Response Fields

Field	Type	Description
`children`	object	Contains results for child directories (if `children=true` was set).
`children.count`	integer	Total number of items returned.
`children.results`	array	List of objects representing directories under the target path.
`site`	string	Name of the site queried.
`pools`	array	List of storage pools (only filled when the path is not specified or children are not included).
`path`	string	The requested directory path.
`total_used_bytes`	integer	Total disk usage in bytes for the specified path.
`total_files_count`	integer	Total number of files under the specified path.

Plugin Development

Ngenea Hub supports extending workflow functionality through custom plugins — Python packages containing Celery tasks that execute on Ngenea Worker nodes alongside built-in tasks.

This guide covers everything needed to develop, deploy, and integrate custom plugins with Ngenea Hub.

Quick Start

This page gets you from zero to a working plugin in minutes. For full details on each topic, follow the cross-references to the relevant sections.

Prerequisites

- SSH access to a Ngenea Worker host
- Write access to `/var/lib/ngenea-worker/plugins`
- Permission to restart the ngenea-worker service
- `enable_plugins=true` set in `/etc/ngenea/ngenea-worker.conf`

1. Scaffold

```
ngenea-worker plugins create my_plugin
```

Creates:

```
/var/lib/ngenea-worker/plugins/my_plugin/
├── setup.py
├── constraints.txt
└── my_plugin/
    ├── __init__.py
    └── my_plugin.py       <-- your code goes here
```

2. Write Your Task

Edit `my_plugin/my_plugin.py`:

```
import logging
from typing import List, Dict, Any

from arcapix.dynamo.server.celeryapp import app
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

logger = logging.getLogger(__name__)


@app.task(bind=True, name="dynamo.custom.my_plugin")
def my_plugin(
    self, *args,
    paths: List[Dict[str, Any]] = None,
    jobid: int = None,
    **kwargs,
):
    # 1. Create a status tracker
    status = FileActionStatus(
        self.name, paths, jobid,
        get_current_task_queue_name()
    )

    # 2. Process each path
    for item in paths:
        try:
            logger.info("Processing %s", item["path"])
```

```
            # --- YOUR LOGIC HERE ---
            status.add("processed", item["path"])
        except Exception as e:
            status.add("failures", item["path"], message=str(e))

    # 3. Return results to Hub
    return status.dump()
```

The Rules

Rule	Why
Decorator must have `bind=True`	Gives access to `self` for task metadata.
Name must be `dynamo.custom.<name>`	Ngenea Hub uses this exact string to route the task.
Must accept `paths`, `jobid`, `**kwargs`	This is the data contract with the DAG engine. See The Task Contract.
Must return `status.dump()`	Ngenea Hub needs this structure to update the Job Progress UI.
Every path must be categorised	Use `processed`, `skipped`, `failures`, or `aborted`.

3. Configure the Entry Point

Edit `setup.py`:

```
from setuptools import setup, find_packages

setup(
    name="my_plugin",
    version="1.0.0",
    packages=find_packages(),
    install_requires=[
        "celery==5.6.3"
    ],
    entry_points={
        "worker_plugin": [
            "my_plugin = my_plugin.my_plugin:my_plugin",
        ],
    },
)
```

Entry point format: `alias = package.module:function`

4. Dependencies

Edit `setup.py`:

```
from setuptools import setup, find_packages

setup(
    name="my_plugin",
```

```
    version="1.0.0",
    packages=find_packages(),
    install_requires=[
        "celery==5.6.3",
        # Any changes to the celery dependency will cause
instability, to avoid this, keep this pinned
        # Then, add your external dependencies below.
        "numpy==2.4.0",
        "tensorflow==2.21.0"
        # Example case of a tensorflow task that may leverage a
LLM
    ],
    entry_points={
        "worker_plugin": [
            "my_plugin = my_plugin.my_plugin:my_plugin",
        ],
    },
)
```

Warning: All plugins must have *celery* version *5.6.3* within their dependencies, any plugins made prior to 2.11 will need to be updated to match this version.

Any plugin defined dependencies that overlap with the existing modules within Ngenea Worker will take precedent over the standard Ngenea Worker dependencies. This may cause instability within Ngenea Worker, if this instability persists remove the offending plugin and alter the dependencies.

5. Install and Restart

```
ngenea-worker plugins install my_plugin
systemctl restart ngenea-worker

# Verify it's loaded
ngenea-worker plugins list
```

6. Create a Workflow

Custom workflows are created via the REST API at `POST /api/workflows/` (see API Endpoints - Workflows, Automation & Jobs). Minimal JSON:

```
{
    "name": "my_workflow",
    "label": "My Custom Workflow",
    "icon_classes": ["fa fa-plug fa-stack-2x text-primary"],
    "discovery": null,
    "enabled": true,
    "visible": true,
    "fields": [],
    "filter_rules": [
        {
            "type": "all",
```

```
            "state": "all",
            "action": [
                {"name": "dynamo.custom.my_plugin"}
            ]
        }
    ]
}
```

The workflow appears in the Ngenea Hub UI. Select files, choose your workflow, and run it. Or execute via the API with `POST /api/file/workflow`.

Passing Parameters

Define **fields** in the workflow and reference them with `*field_name` syntax:

```
{
    "fields": [
        {"name": "dest", "label": "Destination", "type": "string"}
    ],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.custom.my_plugin",
                    "destination": "*dest"
                }
            ]
        }
    ]
}
```

The value arrives in your task's `**kwargs`:

```
def my_plugin(self, *args, paths=None, jobid=None, **kwargs):
    destination = kwargs.get("destination")
```

You can also hardcode values directly in the action object:

```
{"name": "dynamo.custom.my_plugin", "retries": 3, "mode": "fast"}
```

For all field types and advanced workflow configuration, see Workflows.

Chaining Tasks

List multiple tasks in the `action` array. Output paths from one task feed into the next:

```
"action": [
    {"name": "dynamo.tasks.migrate"},
    {"name": "dynamo.custom.my_plugin"},
```

```
    {"name": "dynamo.custom.email_staff",
     "staff_members": ["admin@org.com"]}
]
```

Flow: `migrate` passes succeeded paths to `my_plugin`, which passes its succeeded paths to `email_staff`.

File States Reference

State	Meaning	Passed to next task?
`processed`	Successfully handled.	Yes
`skipped`	Already in desired state.	Yes
`failures`	Error occurred.	No
`aborted`	Intentionally excluded.	No

Updating Your Plugin

1. Edit the source code.
2. **Bump the version** in `setup.py` (the Worker skips unchanged versions).
3. Reinstall and restart:

```
ngenea-worker plugins install my_plugin
systemctl restart ngenea-worker
```

Troubleshooting Cheat Sheet

Problem	Fix
Task stuck in **Pending**	Ensure your function returns `status.dump()`. Check Worker logs for tracebacks.
Plugin not in `ngenea-worker plugins list`	Check entry point format in `setup.py`. Re-run install and restart. Check the plugin is importable. This can be done by running a python interpreter in the worker environment by calling *ngenea-worker -d* and then trying to import your plugin *import my.worker.plugin*
Workflow says task not found	Verify the task `name` in your decorator matches the workflow JSON exactly. Restart the Worker.
Dependency conflict on install	Do not modify the `celery` version in `setup.py`. Check `constraints.txt`.

View Worker logs:

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

For the full troubleshooting guide, see Troubleshooting.

What's Next?

This quick start covers the essentials. The rest of this guide covers:

- Architecture Overview — architecture and key concepts
- The Task Contract — full task contract and return payload details
- Workflows — filter rules, branching, fields, and the `*field_name` syntax
- Advanced Topics — dynamic tasks, sub-DAGs, streaming results, queue management
- Examples — complete worked examples with workflow definitions

Architecture Overview

What Is a Plugin?

A plugin is a standard Python package containing one or more Celery tasks. Plugins are:

- **Discovered** at runtime via Python setuptools entry points
- **Installed** into the Ngenea Worker Python virtual environment
- **Executed** on Worker nodes as steps within Ngenea Hub workflows

Use Cases

Plugins can perform any operation expressible as a Python function operating on a list of file paths:

- **Post-processing hooks** — send email notifications, trigger webhooks, update databases
- **Custom file operations** — move, copy, rename, or transform files
- **External system integration** — call cloud APIs, submit HPC jobs, sync metadata to catalogues
- **Script execution** — run shell scripts or external programs on Worker hosts
- **Validation gates** — check file integrity, enforce naming conventions before downstream tasks proceed

Danger: Plugin tasks execute with the **full permissions of the Worker service** (typically root) and have unrestricted access to the entire filesystem. A bug in your plugin code (e.g., an incorrect path variable, an overly broad glob pattern) can delete or overwrite **any file on the system**, including other users' data, system configuration, or the Worker's own state. Thoroughly test all file operations in a non-production environment first, validate all paths, and never use recursive delete without strict path validation.

Where Plugins Fit

The execution flow for a plugin task is:

1. A user (or schedule/policy) triggers a **Workflow** via the Ngenea Hub UI or REST API.
2. Ngenea Hub converts the workflow definition into a **DAG (Directed Acyclic Graph)** of tasks — a Job containing Task nodes.
3. Ngenea Hub submits each task to the appropriate Celery queue based on site and queue configuration.
4. A Ngenea Worker picks up the task, discovers the plugin function via its entry point, and executes it.
5. The plugin receives a list of **paths**, processes them, and returns a structured **result payload**.
6. Ngenea Hub's DAG engine receives the result, updates job progress in the UI, and submits downstream tasks.

Key Concepts

Concept	Description
Workflow	A reusable template defining a sequence or graph of tasks, path filters, and runtime fields. Created via the Workflows API.
Job	A single execution instance of a workflow, containing input paths and runtime field values.
Task	A single task node within a Job's execution graph. Maps to a Celery task invocation.
Filter Rule	A rule within a workflow that matches paths by type and state, then routes them to an ordered chain of task actions.
Field	A runtime parameter defined in a workflow, whose value is provided at execution time and passed to tasks as a keyword argument.
Plugin	A Python package installed on the Worker, discovered via the `worker_plugin` entry point group.

Getting Started

This section walks through creating, installing, and running a plugin end to end.

Prerequisites

- SSH access to a Ngenea Worker host
- Write permissions to `/var/lib/ngenea-worker/plugins`
- Permission to restart the ngenea-worker service
- `enable_plugins=true` set in `/etc/ngenea/ngenea-worker.conf`
- Access to the Ngenea Hub REST API for creating workflows

Scaffold the Plugin

On a Worker host, run:

```
ngenea-worker plugins create hello_world
```

This creates the following structure:

```
/var/lib/ngenea-worker/plugins/hello_world/
├── setup.py                    # Package metadata and entry
points
├── constraints.txt             # Critical package version
constraints (read-only reference)
└── hello_world/
    ├── __init__.py
    └── hello_world.py          # Your task implementation goes
here
```

Implement the Task

Edit `/var/lib/ngenea-worker/plugins/hello_world/hello_world/hello_world.py`:

```
import logging
from typing import List, Dict, Any

from arcapix.dynamo.server.celeryapp import app
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

logger = logging.getLogger(__name__)


@app.task(bind=True, name="dynamo.custom.hello_world")
def hello_world(self, *args, paths: List[Dict[str, Any]] = None, j
obid: int = None, **kwargs):
    """A minimal plugin that logs each path and marks it as
processed."""
    status = FileActionStatus(self.name, paths, jobid, get_curren
t_task_queue_name())

    for item in paths:
        logger.info("Hello from plugin! Processing: %s", item["pa
th"])
        status.add("processed", item["path"])

    return status.dump()
```

Configure the Entry Point

Edit `/var/lib/ngenea-worker/plugins/hello_world/setup.py`:

```
from setuptools import setup, find_packages

setup(
    name="hello_world",
    version="1.0.0",
    packages=find_packages(),
    install_requires=[

# The template may include celery here — do not modify its
version.
        # Add your external dependencies below.
    ],
    entry_points={
        "worker_plugin": [
            "hello_world = hello_world.hello_world:hello_world",
        ],
    },
)
```

The entry point format is: `alias = package.module:function`

- **alias**: A human-readable name used by the Worker to register the task.
- **package.module**: The dotted Python import path to the module containing the task function.
- **function**: The name of the decorated task function.

Install and Activate

```
# Install the plugin into the Worker's Python environment
ngenea-worker plugins install hello_world

# Restart the Worker to load the new task
# IMPORTANT: Check for active jobs first — restart will wait for
upto 900s for tasks to complete
systemctl restart ngenea-worker

# Verify it appears in the plugin list
ngenea-worker plugins list
```

Create a Workflow

Custom workflow definitions are created via the REST API at `POST /api/workflows/` (see Workflows API). They are not currently editable via the Ngenea Hub UI.

```
{
    "name": "hello_test",
    "label": "Hello World Test",
    "icon_classes": [
        "fa fa-plug fa-stack-2x text-primary"
    ],
    "discovery": null,
    "enabled": true,
```

```
    "visible": true,
    "fields": [],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.custom.hello_world"
                }
            ]
        }
    ]
}
```

Run It

Once created, the workflow appears in the Ngenea Hub UI. Select files in the file browser, choose the “Hello World Test” workflow, and submit.

Alternatively, execute via the API with `POST /api/file/workflow`:

```
{
    "paths": ["/mmfs1/data/sample_data/file1.dat"],
    "site": "london",
    "workflow": "hello_test",
    "fields": {}
}
```

The job progress view will show your paths being marked as “processed”.

The Task Contract

Every plugin task must satisfy a strict contract so that Ngenea Hub’s DAG engine can schedule it, pass data, receive results, and update the UI.

Task Registration

Your function must be registered as a Celery task:

```
@app.task(bind=True, name="dynamo.custom.<your_task_name>")
```

Parameter	Requirement
`bind=True`	**Required.** Gives access to `self` for task metadata (e.g., `self.name`, `self.request.id`).
`name`	**Required.** Must follow the pattern `dynamo.custom.<your_task_name>`. This exact string is used to reference the task in workflow definitions.

Function Signature

```
def my_task(self, *args, paths=None, jobid=None, **kwargs):
```

Parameter	Type	Description
`self`	Task instance	Provided by `bind=True`. Use `self.name` for the task name, `self.request.id` for the Celery task ID.
`*args`	positional	Accept but generally ignore. Present for forward compatibility.
`paths`	`List[Dict]`	The list of path objects to process. Each dict has at minimum a `"path"` key with an absolute filesystem path. May also include `"size"`, `"state"`, `"message"`, and other keys.
`jobid`	`int`	The numeric ID of the parent Job in Ngenea Hub's database.
`**kwargs`	keyword args	**Required.** Captures workflow fields and other arguments passed by Ngenea Hub.

Warning: Omitting `**kwargs` from the function signature will cause the task to raise a `TypeError` if Ngenea Hub passes any unexpected keyword arguments (e.g., new system parameters in a future version, or workflow fields). This will fail the **entire task**, not just individual paths, and halt the workflow chain. Always include `**kwargs` even if you do not currently use it.

To pass specific keyword arguments to a task, include these in the workflow definition as hardcoded values or runtime fields (see Workflows).

Path Object Structure

Each element in the `paths` list is a dictionary. The guaranteed key is `"path"`. Other keys are optional:

```
{
    "path": "/mmfs1/data/project/file1.dat",   # Always present — absolute path
    "size": 264321,                            # File size in bytes (optional)
    "state": "created",
# File state from discovery (optional)
    "message": "...",                          # Status message from upstream (optional)
    "site": "london"                           # Origin site (optional)
}
```

Return Payload

Your task **must** return a dictionary with a specific structure. The recommended approach is to use the `FileActionStatus` helper class. The raw structure is documented here for reference.

Using FileActionStatus (Recommended)

```
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

status = FileActionStatus(
    taskname=self.name,
    input_paths=paths,
    jobid=jobid,
    queue_name=get_current_task_queue_name()
)
```

For each path your task processes, call `status.add()` with the appropriate state:

```
# File processed successfully
status.add("processed", "/mmfs1/data/project/file1.dat")

# File skipped (already in desired state)
status.add("skipped", "/mmfs1/data/project/file2.dat")

# File failed with an error message
status.add("failures", "/mmfs1/data/project/file3.dat", message="
Permission denied")

# File intentionally excluded from further processing
status.add("aborted", "/mmfs1/data/project/file4.dat", message="M
issing required metadata")
```

You can also add log messages visible in the Ngenea Hub job detail view:

```
status.add_log("Connected to external API successfully")
```

Always end your task function with:

```
return status.dump()
```

Supported File States

State	Meaning	Forwarded to next task?
`processed`	File was successfully processed.	Yes
`skipped`	File was already in the desired state; no action taken.	Yes

State	Meaning	Forwarded to next task?
`failures`	An error occurred processing this file.	No
`aborted`	File was intentionally excluded and should not be processed by subsequent tasks.	No
`inprogress`	File is still being processed asynchronously. Rarely used.	No

Danger: Any input path not categorised into a state (`processed`, `skipped`, `failures`, or `aborted`) will **silently disappear** from the job. It will not be forwarded to downstream tasks, not appear in reports, and not be retried. In workflows involving file migration or deletion, this means files can be **permanently orphaned** — never moved, never cleaned up, and never flagged as an error. Always iterate over every path in the input list and assign each one a state.

Complete Pattern

```
from arcapix.dynamo.server.celeryapp import app
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

@app.task(bind=True, name="dynamo.custom.my_task")
def my_task(self, *args, paths=None, jobid=None, **kwargs):
    status = FileActionStatus(self.name, paths, jobid, get_curren
t_task_queue_name())

    for item in paths:
        try:
            do_something(item["path"])
            status.add("processed", item["path"])
        except FileNotFoundError:
            status.add("skipped", item["path"], message="File no
longer exists")
        except Exception as e:
            status.add("failures", item["path"], message=str(e))

    return status.dump()
```

What Happens If the Contract Is Violated

Problem	Symptom in Ngenea Hub
Task doesn't return `status.dump()` or equivalent dict	Task stuck in **Pending** state; traceback in Worker logs.
Missing `"jobid"` in return payload	Job progress does not update.
`"summary"` counts don't match `"details"` lists	Inaccurate statistics in the Job Progress UI.

Problem	Symptom in Ngenea Hub
Unhandled exception in task function	Task marked as **Failed**; downstream tasks may not run. See warning below.
Paths not accounted for	“Missing” files in the job report.

Danger: An unhandled exception aborts the **entire task**, not just the current file. All remaining input paths — and all downstream tasks in the chain — will not be processed. In a multi-step workflow (e.g., migrate then delete source), this can leave the system in an **inconsistent state** where some files have been migrated but cleanup was never performed. Always catch exceptions per-path inside your processing loop and categorize failed paths with `status.add("failures", ...)`.

Workflows

Once your plugin is installed and the Worker restarted, you reference it in **workflow definitions** on Ngenea Hub. Workflow definitions are created via the REST API at `POST /api/workflows/` (see Workflows API). They are not currently editable via the Ngenea Hub UI.

Workflow Structure

A workflow definition is a JSON object with these top-level fields:

```
{
    "name": "my_workflow",
    "label": "My Workflow",
    "icon_classes": ["fa fa-cog fa-stack-2x text-primary"],
    "discovery": null,
    "discovery_options": null,
    "enabled": true,
    "visible": true,
    "fields": [],
    "filter_rules": []
}
```

Field	Type	Description
`name`	string	Unique internal identifier (max 100 chars). Used in API calls and policies.
`label`	string	Human-readable name displayed in the UI (max 100 chars).
`icon_classes`	list[string]	Font Awesome CSS classes for the workflow icon in the UI.
`discovery`	string or null	Discovery method: `"snapdiff"`, `"recursive"`, or `null` (manual file selection).
`discovery_options`	object or null	Options for the discovery method (e.g., `{"skip_missing": false}` for recursive discovery). Only applies to the workflow’s default discovery.

Field	Type	Description
`enabled`	bool	Whether the workflow can be triggered.
`visible`	bool	Whether the workflow appears in the UI file browser.
`fields`	list[object]	Runtime parameter definitions. See Workflow Fields below.
`filter_rules`	list[object]	Rules that match paths and define task chains. See Filter Rules below.

Filter Rules

Filter rules are the core of a workflow definition. Each rule matches a subset of input paths and routes them through an ordered chain of task actions.

```
{
    "type": "all",
    "state": "all",
    "action": [
        {"name": "dynamo.tasks.migrate"},
        {"name": "dynamo.custom.my_plugin", "my_param": "value"}
    ]
}
```

Required Keys

Key	Values	Descripti
`type`	`"all"`, `"file"`, `"directory"`, `"symlink"`	Filters p filesystem type. `"all` everything
`state`	`"all"`, `"created"`, `"deleted"`, `"updated"`, `"modified"`, `"moved"`, `"default"`	Filters pat state (re discovery- workflows) matches `"default"` all for matched rules.
`action`	list of action objects	Ordered tasks to e the match

Optional Keys

Key	Type	Description
`include`	list of glob strings	Glob patterns to limit actions to matching paths only.

Key	Type	Description
`exclude`	list of glob strings	Glob patterns to exclude matching paths from the action.
`ignore_site_includes`	bool	If `true`, ignore global site-level include patterns for this rule.
`ignore_site_excludes`	bool	If `true`, ignore global site-level exclude patterns for this rule.

Action Objects

Each action object must have a `"name"` key with the task's registered name. It may also include `"site"` and `"queue"` keys to control where the task executes. Any other key-value pairs are passed to the task function as keyword arguments.

```
{
    "name": "dynamo.custom.my_plugin",
    "site": "london",
    "queue": "highpriority",
    "destination_path": "/archive/project",
    "notify": true
}
```

Key	Required	Description
`name`	Yes	The registered task name (e.g., `dynamo.custom.my_plugin`) .
`site`	No	Target site for task execution. Overrides the job's default site.
`queue`	No	Target queue for task execution. Overrides the job's default queue.

Important: `site` and `queue` are consumed by the DAG engine and are **not** passed to your task function. All other key-value pairs arrive in the task's `**kwargs`.

Warning: If `"site"` or `"queue"` is set incorrectly, the task will execute on a Worker at the wrong location. That Worker may not have access to the target filesystem, causing the task to fail — or worse, it may have access to a **different filesystem mounted at the same path**, causing file operations to affect the wrong data. Always verify that the target site has access to the paths your workflow will process.

Task Chaining

Actions within a rule's `"action"` list are executed **in order**, forming a chain. The output `"paths"` from one task become the input `paths` for the next:

```
"action": [
    {"name": "dynamo.tasks.migrate"},
    {"name": "dynamo.custom.validate_checksums"},
    {"name": "dynamo.custom.email_staff", "staff_members": ["admin
```

```
@org.com"]}
]
```

Execution flow:

1. `migrate` processes the input paths, returns processed/skipped paths.
2. `validate_checksums` receives only the paths that `migrate` marked as processed or skipped.
3. `email_staff` receives only the paths that `validate_checksums` passed through.

Path Filtering with Include/Exclude

Rules can use glob patterns to limit which paths reach the action chain:

```
{
    "type": "file",
    "state": "created",
    "include": ["/mmfs1/data/project_a/*"],
    "exclude": ["*.tmp", "*.log"],
    "action": [
        {"name": "dynamo.tasks.migrate"},
        {"name": "dynamo.custom.my_plugin"}
    ]
}
```

A path must match at least one `include` pattern (if defined) **and** not match any `exclude` pattern. If no `include` is defined, all paths are considered included unless excluded.

Sites can also define global include/exclude patterns. These are appended to per-rule patterns by default. Set `"ignore_site_includes": true` or `"ignore_site_excludes": true` on a rule to override this behaviour.

Multiple Rules (Branching)

Warning: In discovery workflows, paths can have various states (`created`, `deleted`, `updated`, `modified`, `moved`). If your filter rules only cover some states, paths with unmatched states are **silently dropped** — they will not be processed, reported, or flagged. Add a `"state": "default"` rule as a catch-all to ensure no paths are silently lost.

You can define multiple filter rules to create branching workflows. A common pattern with discovery workflows:

```
{
    "filter_rules": [
        {
            "type": "all",
            "state": "created",
            "action": [
                {"name": "dynamo.tasks.migrate"},
```

```
                {"name": "dynamo.custom.email_staff", "subject": "
New files migrated"}
            ]
        },
        {
            "type": "all",
            "state": "deleted",
            "action": [
                {"name": "dynamo.tasks.delete_paths_from_gpfs"}
            ]
        }
    ]
}
```

Referencing Your Plugin

The task name you use in workflow definitions must **exactly match** the `name` parameter in your `@app.task` decorator:

```
# In your plugin code:
@app.task(bind=True, name="dynamo.custom.file_mover")
```

```
{"name": "dynamo.custom.file_mover", "destination_path": "/
archive"}
```

Workflow Fields

Fields let you define runtime parameters that users provide when triggering a workflow. Field values are passed to your task function as keyword arguments.

Defining Fields

Fields are defined in the workflow's `"fields"` array:

Key	Type	Description
`name`	string	**Required.** Internal name — used to reference the field in action definitio and received as a kwarg in your task.
`label`	string	**Required.** Human-readable label shown in the UI.
`type`	string	**Required.** One of: `"string"`, `"int"`, `"bool"`, `"choices"`, `"list` `"enum[site]"`, `"enum[queue]"`.
`default`	varies	Optional. Default value if the user doesn't provide one.
`choices`	list of objects	Required if `type` is `"choices"`. Each entry must have `"label"` (display tex and `"value"` (the value passed to the task).

Field Type Reference

Type	Description
`string`	Free-text input. Value arrives as a Python `str`.

Type	Description
`int`	Numeric input. Value arrives as a Python `int`.
`bool`	Checkbox. Value arrives as a Python `bool`.
`choices`	Dropdown menu. Requires a `choices` list of `{"label": "...", "value": "..."}` objects. The selected `value` is passed to the task.
`list`	Multi-value input. Value arrives as a Python `list`.
`enum[site]`	Dropdown populated with configured sites. Value is the site name string.
`enum[queue]`	Dropdown populated with configured queues. Value is the queue name string.

Warning: String fields used as filesystem paths are passed to your task **without validation or sanitization**. A user could enter paths like `/`, `../../../etc`, or paths with special characters. Since plugins typically run as root, this can result in file operations targeting **any location on the system**. Always validate user-supplied paths in your task code: check that they are absolute, within an expected base directory, and do not contain traversal sequences.

Referencing Fields in Actions

To pass a field's runtime value to a task, use the `*field_name` syntax in the action definition:

```
{
    "fields": [
        {"name": "dest", "label": "Destination Path", "type": "st
ring"},
        {"name": "notify", "label": "Send Notification", "type": "
bool", "default": false}
    ],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.custom.file_mover",
                    "destination_path": "*dest"
                },
                {
                    "name": "dynamo.custom.email_staff",
                    "send_notification": "*notify",
                    "staff_members": ["admin@org.com"]
                }
            ]
        }
    ]
}
```

The `*dest` syntax tells Ngenea Hub to substitute the runtime value of the `dest` field. Hardcoded values (like `"staff_members"`) are passed directly.

Choices field example:

```
{
    "fields": [
        {
            "name": "compression",
            "label": "Compression Mode",
            "type": "choices",
            "choices": [
                {"label": "Fast", "value": "fast"},
                {"label": "Balanced", "value": "balanced"},
                {"label": "Maximum", "value": "max"}
            ],
            "default": "balanced"
        }
    ],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.custom.my_plugin",
                    "compression_mode": "*compression"
                }
            ]
        }
    ]
}
```

Submitting a Workflow via API

Use the following JSON body `POST /api/file/workflow` to execute a workflow programmatically.:

```
{
    "paths": ["/mmfs1/data/sample_data/file1.dat"],
    "site": "london",
    "workflow": "my_workflow",
    "fields": {
        "dest": "/archive/project",
        "notify": true
    }
}
```

This endpoint requires authentication, see API Endpoints - Workflows, Automation & Jobs for full details.

Advanced Topics

This section covers advanced plugin patterns including streaming results, dynamic task spawning, sub-DAG construction, and queue management.

Streaming Task Results

This applies to delete and move handlers in a snapdiff workflow.

Rather than returning results at the end of execution, delete and move handlers in snapdiff workflows must **stream** results back to Ngenea Hub. When Ngenea Hub expects streaming, it passes `stream_results=True` as a keyword argument to the task.

Important: If your task will be used as a move or delete handler in a snapdiff workflow, it must accept `stream_results` either as an explicit parameter or via `**kwargs`.

Danger: Snapdiff workflows can discover **millions of paths**. If your task is used as a move or delete handler without implementing `StreamTaskResults`, all results will accumulate in memory, eventually causing the Worker process to be killed by the OOM killer. This will abort the job mid-execution and may leave the filesystem in an **inconsistent state** (partially moved or deleted files). Always check for `stream_results=True` and use `StreamTaskResults` accordingly.

Using StreamTaskResults

A `StreamTaskResults` class is provided to facilitate streaming:

```
from arcapix.dynamo.server.streamstatus import StreamTaskResults
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

@app.task(bind=True, name="dynamo.custom.my_snapdiff_handler")
def my_snapdiff_handler(self, *args, paths=None, jobid=None, stre
am_results=False, **kwargs):
    queue_name = get_current_task_queue_name()

    if stream_results:
        status = StreamTaskResults(self.name, paths, jobid, queue_
name, self.request.id)
    else:
        status = FileActionStatus(self.name, paths, jobid, queue_
name)

    for item in paths:
        status.add("processed", item["path"])

    return status.dump()
```

The `StreamTaskResults` class behaves the same as `FileActionStatus`, but sends results to Ngenea Hub in batches in the background. Calling `status.dump()` flushes any outstanding results.

The additional parameter compared to `FileActionStatus` is `taskid` — the Celery task ID, retrieved via `self.request.id`.

Streaming Path Input

For snapdiff workflows, paths may be streamed *to* your task rather than passed all at once. Use `expand_paths` to consume them:

```
from arcapix.dynamo.server.utils.inputs import expand_paths
```

Danger: **Never** materialize the `expand_paths` generator into a list (e.g., `list(expand_paths(...))` or storing all values in a collection). Snapdiff workflows may stream millions of paths. Collecting them all into memory will exhaust Worker RAM, crash the process, and leave the job in an incomplete state with potential data inconsistency. Process paths one at a time or in small batches using the generator directly.

Dynamic Children

Some tasks need to spawn child tasks at runtime — for example, to batch paths for load balancing or to submit different tasks based on processing results.

Using DynamicSubmitter

The `DynamicSubmitter` class handles dynamic child task spawning and automatically manages the `will_spawn_children` flag, which Ngenea Hub uses to track job completion:

```
from arcapix.dynamo.server.dags.status import DynamicSubmitter
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

@app.task(bind=True, name="dynamo.custom.dynamic_plugin")
def dynamic_plugin(self, *args, paths=None, jobid=None, **kwargs):
    status = FileActionStatus(self.name, paths, jobid, get_curren
t_task_queue_name())

    submitter = DynamicSubmitter(
        status=status,
        settings={},
        task_id=self.request.id,
        chunk_size=40,
        chunk_gb=1,
        auto_chunk_size=False,
    )
```

```
    for path in paths:
        submitter.add("files", path["path"])

    return submitter.dump()
```

Why `will_spawn_children` Matters

The `will_spawn_children` flag tells Ngenea Hub whether your task has spawned dynamic children. Without proper tracking:

- A job may report as "ONGOING" after all tasks have completed.
- A job may report as completed while child tasks are still in flight.

Danger: Dynamic child spawning has no built-in limit on the number of tasks created. Setting `chunk_size` to a very small value (e.g., 1) on a large job, or implementing recursive child spawning without a termination condition, can flood the Celery broker with millions of tasks. This will **stall all Workers across all sites** — not just the current job — and may require manual broker intervention to recover. Always set reasonable chunk sizes and verify that your spawning logic has a well-defined termination condition.

Warning: If `will_spawn_children` incorrectly returns `False` when children have been spawned, Ngenea Hub will mark the job as **complete while child tasks are still executing**. Downstream automation (policies, notifications, cleanup) may trigger prematurely, acting on incomplete data. Always use `DynamicSubmitter` rather than implementing child spawning manually.

The `DynamicSubmitter` handles this automatically. You can override the `will_spawn_children()` method to implement custom logic:

```
class MyCustomSubmitter(DynamicSubmitter):
    def will_spawn_children(self) -> bool:
        return self._spawned_children
```

Sub-DAGs

Sub-DAGs allow you to dynamically define and submit workflows at runtime. A sub-DAG can consist of a single task or a chain of multiple tasks. Once submitted, it is automatically integrated into the parent job for reporting and job control.

Single-Task Sub-DAG

Use `task_template` to define a single-task sub-DAG:

```
from arcapix.dynamo.server.dags.util import task_template, submit_
subdag

def submit_stub_task(self, paths, source_site, dest_site,
**kwargs):
```

```
    stub_template = task_template(
        "dynamo.tasks.reverse_stub",
        kwargs={
            "hydrate": False,
            "overwrite": True,
        },
        site=dest_site,
    )

    submit_subdag(
        paths,
        self.request.id,
        "dynamocore.tasks.dag.run_subdag",
        subgraph_template=stub_template,
    )
```

Multi-Task Sub-DAG (Chains)

For complex sub-DAGs with chained tasks, use the `Graph` and `add_task` helpers from `arcapix.dynamo.server.dag.helper`:

```
from arcapix.dynamo.server.dag.helper import Graph, add_task
from arcapix.dynamo.server.dags.util import submit_subdag

def submit_chain(self, paths, jobid, source_site, dest_site, endp
oint, **kwargs):
    graph = Graph()

    xattr = graph.add_task(
        "dynamo.tasks.remove_location_xattrs_for_moved",
        kwargs={"jobid": jobid},
        site=source_site,
    )

    recall = graph.add_task(
        "dynamo.tasks.reverse_stub",
        parents=[xattr.id],
        kwargs={
            "jobid": jobid,
            "site": dest_site,
            "endpoint": endpoint,
            "hydrate": True,
            "overwrite": True,
        },
        site=dest_site,
    )

    submit_subdag(
        [{"path": p} for p in paths],
        self.request.id,
        "dynamocore.tasks.dag.run_subdag",
        subgraph_template=graph.to_dict(),
    )
```

The `parents` parameter establishes dependencies between tasks in the sub-DAG. In the example above, `recall` will only run after `xattr` completes.

> **Warning:** If the `parents` parameter is omitted or incorrect, sub-DAG tasks will execute in **parallel** rather than sequentially. For operations that depend on ordering (e.g., removing xattrs before recalling a file), parallel execution can cause **data corruption** or silent failures. Always verify the dependency chain by confirming each task's `parents` list references the correct predecessor task IDs.

Queue Utilities

Several utility functions are available for working with Celery queues in your tasks.

```
from arcapix.dynamo.server.queue import (
    get_current_task_queue_name,
    get_current_site,
    get_formatted_queue_name,
    ParsedQueueName,
)
```

Function / Class	Description
`get_current_task_queue_name()`	Returns the name of the Celery queue the current task is running on.
`get_current_site()`	Returns the name of the site the current task is running on.
`get_formatted_queue_name(site, queue, function)`	Returns the Celery queue name for a given `site`, `queue`, and `function`.
`ParsedQueueName`	A class representing a parsed Celery queue name with `site`, `queue`, and `function` attributes.

Example:

```
parsed_queue = ParsedQueueName.current_task_queue()

print(parsed_queue.site)       # london
print(parsed_queue.queue)      # highpriority
print(parsed_queue.function)   # custom

print(str(parsed_queue))       # london.highpriority#custom
```

> **Important:** The format of Celery queue names is subject to change. Always use the above functions rather than manually parsing or formatting queue names.

Plugin Lifecycle

This section covers the full lifecycle of a plugin: creating, installing, updating, listing, and uninstalling.

Creating a Plugin

Use the CLI to scaffold a new plugin:

```
ngenea-worker plugins create <plugin_name>
```

This creates the following structure under `/var/lib/ngenea-worker/plugins/<plugin_name>/`:

```
<plugin_name>/
├── setup.py                    # Package metadata and entry
points
├── constraints.txt             # Critical package version
constraints (read-only reference)
└── <plugin_name>/
    ├── __init__.py
    └── <plugin_name>.py        # Your task implementation goes
here
```

The generated module contains a default `example_task` function that you should rename and modify for your use case.

Enabling Plugins

Plugins must be enabled in the Ngenea Worker configuration:

1. Edit `/etc/ngenea/ngenea-worker.conf`.
2. Set `enable_plugins=true`.
3. Save the file.

Installing a Plugin

Install all plugins in the plugins directory:

```
ngenea-worker plugins install
```

Install a specific plugin:

```
ngenea-worker plugins install <plugin_name>
```

After installing, restart the ngenea-worker service to load the new tasks:

```
systemctl restart ngenea-worker
```

Warning: Restarting the Worker service **immediately terminates all in-flight tasks** on that host. Tasks that are mid-execution (e.g., file moves, script runs) may leave the system in an **inconsistent state** — for example, a file that has been deleted from the source but not yet written to the destination. Before restarting, check the Ngenea Hub UI for active jobs on this Worker and either wait

for them to complete or ensure the affected workflows are idempotent and can safely be re-run.

Listing Plugins

To list installed plugins and their versions:

```
ngenea-worker plugins list
```

This shows two sections:

- **Installed plugins** — plugins currently installed with their versions.
- **Available plugins** — all plugins in the `/var/lib/ngenea-worker/plugins` directory, whether installed or not.

Updating a Plugin

To update an installed plugin:

1. Modify the task code in the plugin module.
2. Best practice: Increment the version number in `setup.py` to reflect an updated plugin.
3. If necessary copy the new code to each worker node that needs the plugin.
4. Reinstall the plugin on each node:

   ```
   ngenea-worker plugins install <plugin_name>
   ```

5. Restart the service on each node:

   ```
   systemctl restart ngenea-worker
   ```

Managing Dependencies

If your plugin requires external Python packages, add them to the `install_requires` list in `setup.py`:

```
install_requires = [
    "celery<=5.4",
    "requests>=2.28",
    "my-custom-lib==1.0.0",
]
```

Danger: **Never** modify or force-install versions of `celery` or other constrained packages (listed in the templates `constraints.txt`) by bypassing the `ngenea-worker plugins install` command. Doing so can cause the Worker to fail to start, crash during task execution, or introduce **silent data corruption** in task

payloads due to serialization incompatibilities. The constraints exist to guarantee compatibility with Ngenea Hub's DAG engine.

If your plugin requests a dependency that conflicts with a constrained package, you will see an error like:

```
ERROR: Cannot install example_plugin==0.1.0 because these package
versions
have conflicting dependencies.

The conflict is caused by:
    example_plugin 0.1.0 depends on celery==5.4
    The user requested (constraint) celery==5.3.6
```

`The user requested (constraint)` refers to Ngenea Worker's internal constraints.

Your plugin directory includes a `constraints.txt` file listing the required critical package versions. This file is for reference and testing only — editing it will not change the constraints applied during installation.

Uninstalling a Plugin

Uninstall a specific plugin:

On each worker node the plugin is installed on:

```
ngenea-worker plugins uninstall <plugin_name>
```

Uninstall all plugins:

On each worker node the plugins are installed on:

```
pushd /var/lib/ngenea-worker/plugins/
ngenea-worker plugins uninstall * -y
popd
```

Danger: Running `ngenea-worker plugins uninstall *` outside of `/var/lib/ngenea-worker/plugins/` or including non-plugin package names can **u ninstall core Worker components**, causing the Worker service to fail entirely. This will halt **all workflow processing** on the affected host and require manual reinstallation to recover. Always `cd` to the plugins directory first and verify the glob expansion before confirming.

After uninstalling, restart the service:

```
systemctl restart ngenea-worker
```

Troubleshooting

Common Issues

Symptom	Likely Cause	Fix
Task stuck in **Pending**	Task does not return `status.dump()` or an equivalent dict.	Ensure the task returns a valid payload. Check Worker logs for tracebacks.
Task marked as **Failed**	Unhandled exception in the task function.	Add `try/except` around processing logic. Mark failed paths with `status.add("failures", ...)`.
Job progress not updating	Missing `"jobid"` in the return payload.	Ensure `jobid` is passed to `FileActionStatus` and included in the return dict.
Inaccurate statistics in Job UI	`"summary"` counts do not match `"details"` lists in the return payload.	Use `FileActionStatus` to avoid manual counting errors.
"Missing" files in job report	Not all input paths are accounted for in the return payload.	Ensure every path in the input list is categorised into a state (`processed`, `skipped`, `failures`, or `aborted`).
Plugin not listed after install	Version number not bumped in `setup.py`, or service not restarted.	Increment the version in `setup.py`, reinstall, and run `systemctl restart ngenea-worker`.
Dependency conflict on install	Plugin requests a version of a constrained package (e.g., `celery`) that conflicts with Ngenea Worker requirements.	Adjust the dependency version in `setup.py` to be compatible. Check `constraints.txt` for required versions.
Task not found in workflow	Task `name` in the workflow definition does not match the `name` parameter in the `@app.task` decorator.	Ensure the names match exactly (e.g., `dynamo.custom.my_task`).
Job reports "ONGOING" after completion	Dynamic children spawned without proper `will_spawn_children` tracking.	Use `DynamicSubmitter` which handles this automatically. See Dynamic Children.

Logging

Use Python's standard `logging` module in your tasks. Log output appears in the Ngenea Worker service logs:

```
import logging

logger = logging.getLogger(__name__)
```

```
@app.task(bind=True, name="dynamo.custom.my_task")
def my_task(self, *args, paths=None, jobid=None, **kwargs):
    logger.info("Starting task %s for job %s with %d paths",
self.name, jobid, len(paths))

    for item in paths:
        logger.debug("Processing: %s", item["path"])
        # ...
```

View logs with:

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

You can also add messages to the Ngenea Hub job detail view using `status.add_log()`:

```
status.add_log("Connected to external API successfully")
status.add_log("Processed 150 of 200 files")
```

These messages are visible in the UI alongside the job progress.

Inspecting Return Payloads

During development, it can be helpful to log the return payload to verify its structure:

```
result = status.dump()
logger.debug("Return payload: %s", result)
return result
```

The expected structure is documented in The Task Contract. Key things to verify:

- `"jobid"` is present and correct.
- `"status" > "summary"` counts match `"status" > "details"` list lengths.
- `"paths"` contains only the paths to forward to the next task (typically `processed` and `skipped`).
- Every input path appears in exactly one state category.

Examples

This section provides complete, working plugin examples that can be used as starting points for your own implementations.

Email Notification

This plugin sends email notifications at the end of a workflow. It accepts configurable recipients, subject, sender, and SMTP server via workflow fields or hardcoded action parameters.

Task Implementation

```
import logging
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.utils import formatdate
from smtplib import SMTP
from typing import List, Dict, Any

from arcapix.dynamo.server.celeryapp import app
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

logger = logging.getLogger(__name__)


def send_email(email_to, email_from, subject, message, server="lo
calhost"):
    """Send an email via SMTP."""
    msg = MIMEMultipart()
    msg['From'] = email_from
    msg['To'] = ', '.join(email_to)
    msg['Date'] = formatdate(localtime=True)
    msg['Subject'] = subject
    msg.attach(MIMEText(message))

    try:
        smtp = SMTP(server)
        smtp.sendmail(email_from, email_to, msg.as_string())
        smtp.close()
    except OSError as error:
        logger.error("Error sending notification email: %s",
error)
        raise


@app.task(bind=True, name="dynamo.custom.email_staff")
def email_staff(
    self,
    *args,
    paths: List[Dict[str, Any]] = None,
    jobid: int = None,
    staff_members: List[str] = None,
    message: str = None,
    subject: str = None,
    server: str = None,
    from_address: str = None,
    **kwargs,
):
    status = FileActionStatus(self.name, paths, jobid, get_curren
t_task_queue_name())

    if not message:
```

```
        message = f"Job {jobid} has completed processing {len(pat
hs)} paths"
    if not from_address:
        from_address = "ngenea-worker@example.com"
    if not subject:
        subject = "Ngenea Worker Job completed"
    if not server:
        server = "localhost"

    try:
        send_email(
            email_to=staff_members,
            email_from=from_address,
            subject=subject,
            message=message,
            server=server,
        )
        status.add_log(f"Notification sent to {', '.join(staff_me
mbers)}")
        for item in paths:
            status.add("processed", item["path"])
    except Exception as e:
        status.add_log(f"Failed to send email: {e}")
        for item in paths:
            status.add("failures", item["path"], message=str(e))

    return status.dump()
```

Entry Point (setup.py)

```
entry_points={
    "worker_plugin": [
        "email_staff = email_staff.email_staff:email_staff",
    ],
},
```

Workflow Definition

This workflow migrates files and then sends an email notification:

```
{
    "name": "migrate_notif",
    "label": "Migrate with Notification",
    "icon_classes": [
        "fa fa-cloud fa-stack-2x text-primary",
        "fa fa-refresh fa-stack-1x text-light"
    ],
    "discovery": null,
    "enabled": true,
    "visible": true,
    "fields": [],
    "filter_rules": [
```

```
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.tasks.migrate"
                },
                {
                    "name": "dynamo.custom.email_staff",
                    "staff_members": [
                        "admin@organisation.com",
                        "ops@organisation.com"
                    ],
                    "subject": "Migration complete",
                    "from_address": "notifications@organisation.c
om"
                }
            ]
        }
    ]
}
```

Script Execution

This plugin executes a shell script on the Ngenea Worker host. It supports configurable script paths, timeouts, and optional path arguments.

Danger: Scripts executed by this plugin run with the **full permissions** of the ngenea-worker service, which is typically **root**. A user-supplied `script_location` field allows **arbitrary command execution as root** on the Worker host. Never expose `script_location` as a runtime workflow field unless access to the workflow is strictly controlled. Hardcode the script path in the workflow action definition whenever possible, and ensure the script itself is not world-writable.

Warning: While the plugin uses `subprocess.run` with a list (avoiding direct shell injection), the target script receives `additional_args` and file paths as raw arguments. If the script passes these to a shell or interprets them unsafely, injection attacks are still possible. Validate and sanitize all inputs before passing them to external scripts.

Warning: If `timeout` is exposed as a runtime field, a user could set it to an arbitrarily large value, causing a hung script to block a Worker slot indefinitely. Always enforce a maximum timeout either in the task code or by hardcoding the value in the workflow definition.

Task Implementation

```
import logging
from subprocess import run as run_script, TimeoutExpired, Subproc
essError, PIPE, STDOUT
```

```
from typing import List, Dict, Any

from arcapix.dynamo.server.celeryapp import app
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

logger = logging.getLogger(__name__)
DEFAULT_SCRIPT = "/opt/cloud_script.sh"


@app.task(bind=True, name="dynamo.custom.run_cloud_script")
def run_cloud_script(
    self,
    *args,
    paths: List[Dict[str, Any]] = None,
    jobid: int = None,
    script_location: str = None,
    timeout: int = 600,
    additional_args: List[str] = None,
    use_paths: bool = False,
    **kwargs,
):
    status = FileActionStatus(self.name, paths, jobid, get_curren
t_task_queue_name())

    try:
        cmd = [script_location if script_location else DEFAULT_SC
RIPT]
        if additional_args:
            cmd = cmd + additional_args
        if use_paths:
            cmd = cmd + [item["path"] for item in paths]

        result = run_script(
            cmd,
            stdout=PIPE,
            stderr=STDOUT,
            check=True,
            timeout=timeout,
        )

        status.add_log(str(result.stdout))

        for item in paths:
            status.add("processed", item["path"])

    except TimeoutExpired:
        status.add_log("Script timed out")
        for item in paths:
            status.add("failures", item["path"], message="Script
timed out")
    except SubprocessError as e:
        status.add_log(str(e))
```

```
        for item in paths:
            status.add("failures", item["path"], message=str(e))

    return status.dump()
```

Entry Point (setup.py)

```
entry_points={
    "worker_plugin": [
        "run_cloud_script =
run_cloud_script.run_cloud_script:run_cloud_script",
    ],
},
```

Workflow Definition

Run the default cloud script after migration:

```
{
    "name": "migrate_cloud_script",
    "label": "Migrate and Run Script",
    "icon_classes": [
        "fa fa-cloud fa-stack-2x text-primary",
        "fa fa-refresh fa-stack-1x text-light"
    ],
    "discovery": null,
    "enabled": true,
    "visible": true,
    "fields": [],
    "filter_rules": [
        {
            "type": "all",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.tasks.migrate"
                },
                {
                    "name": "dynamo.custom.run_cloud_script"
                }
            ]
        }
    ]
}
```

File Mover

This plugin moves files to a configurable destination directory. It demonstrates using workflow fields to parameterise a task.

Danger: `shutil.move` **deletes the source file** after copying. This operation cannot be undone. If `destination_path` is incorrect, inaccessible, or on a volatile filesystem (e.g., `/tmp`), the source files will be **permanently lost**. Always verify the destination path before running this workflow on production data. Consider testing with a small set of non-critical files first.

Warning: `shutil.move` will silently **overwrite** any existing file at the destination with the same name. The overwritten file is permanently lost. If collisions are possible, add an existence check before moving and mark collisions as `failures` or generate unique filenames.

Warning: This example uses only the filename (`os.path.basename`) to construct the destination path, discarding the source directory structure. If multiple source files share the same filename (e.g., `/data/project_a/config.yaml` and `/data/project_b/config.yaml`), later files will **silently overwrite** earlier ones. To preserve directory structure, compute a relative path from a common root instead.

Warning: The return payload reports the **original source path** as processed, but the file has been moved to a new location. If a downstream task in the chain attempts to access the file at the original path, it will fail. To support chaining, update the path in the return payload to reflect the new destination, or use this task only as the final step in a chain.

Task Implementation

```
import logging
import os
import shutil
from typing import List, Dict, Any

from arcapix.dynamo.server.celeryapp import app
from arcapix.dynamo.server.status import FileActionStatus
from arcapix.dynamo.server.queue import get_current_task_queue_na
me

logger = logging.getLogger(__name__)


@app.task(bind=True, name="dynamo.custom.file_mover")
def file_mover(
    self,
    *args,
    paths: List[Dict[str, Any]] = None,
    jobid: int = None,
    destination_path: str = None,
    **kwargs,
):
    status = FileActionStatus(self.name, paths, jobid, get_curren
t_task_queue_name())
```

```
    if not destination_path:
        for item in paths:
            status.add("failures", item["path"], message="No
destination_path provided")
        return status.dump()

    for item in paths:
        src = item["path"]
        filename = os.path.basename(src)
        dest = os.path.join(destination_path, filename)

        try:
            os.makedirs(destination_path, exist_ok=True)
            shutil.move(src, dest)
            logger.info("Moved %s to %s", src, dest)
            status.add("processed", item["path"])
        except FileNotFoundError:
            status.add("skipped", item["path"], message="Source
file not found")
        except Exception as e:
            status.add("failures", item["path"], message=str(e))

    return status.dump()
```

Entry Point (setup.py)

```
entry_points={
    "worker_plugin": [
        "file_mover = file_mover.file_mover:file_mover",
    ],
},
```

Workflow Definition

This workflow uses a runtime field so users can specify the destination when triggering the workflow:

```
{
    "name": "move_files",
    "label": "Move Files to Destination",
    "icon_classes": [
        "fa fa-arrows fa-stack-2x text-primary"
    ],
    "discovery": null,
    "enabled": true,
    "visible": true,
    "fields": [
        {
            "name": "dest",
            "label": "Destination Directory",
            "type": "string"
```

```
        }
    ],
    "filter_rules": [
        {
            "type": "file",
            "state": "all",
            "action": [
                {
                    "name": "dynamo.custom.file_mover",
                    "destination_path": "*dest"
                }
            ]
        }
    ]
}
```

The `*dest` syntax references the runtime field value. When a user triggers this workflow, they will be prompted to enter the destination directory. See Workflow Fields for more details.

Glossary

C

Celery

An open-source tool that helps applications perform tasks in the background without slowing down the main program. It assigns tasks, like processing files or maintaining systems, to separate workers that operate independently. Celery keeps track of which tasks are waiting, in progress, or done, and it can schedule tasks for specific times or retry them if there's an issue.

Celery Beats

A scheduling tool is software that plans and runs tasks automatically at specific times. It's like setting an alarm for a task you want done regularly, such as backing up data every day or syncing files between locations every hour.

Celery Beat is one of these tools; it manages these recurring tasks in the system, ensuring they happen on schedule without needing someone to start them manually.

Celery Monitor

Celery Monitor is a tool that tracks the health and status of Celery workers, which are background processes that handle tasks. It listens for regular "heartbeats" from these workers, which are signals that confirm each worker is still active and functioning. **Celery Worker**

A Celery Worker is a process that runs in the background, ready to take on tasks sent by a central Hub server. These tasks might include organizing files, processing data, or adjusting settings. Each worker is connected to a specific list of tasks, called

a queue, which it monitors constantly. When a new task appears in its queue, the worker completes it and then reports back to the Hub server. This setup allows tasks to be distributed across multiple workers, making task handling more efficient.

D

Data Mover

A Data Mover is a tool or service that helps transfer data from a company's internal storage system to external locations, like cloud storage. It is designed to handle large amounts of data without slowing down the primary system. By using methods like data compression and parallel processing, a Data Mover makes the transfer process faster and more efficient, ensuring the system remains responsive.

Django

Django is a set of tools (web development framework) that helps developers create the backend of a website, like the Hub's user interface. It manages software requests, such as retrieving data from a database or initiating tasks, and ensures fast responses.

Django also helps different parts of the system, like databases (where onfiguration and Hub stateful data is stored) and external services (APIs that provide additional functionality), communicate with each other, allowing everything to work together smoothly.

E

External Target

An external target is any storage type that is not part of your main system. This includes things like cloud storage services (like AWS or Google Cloud) or shared network drives (like NFS). These systems help you move and store your data away from your main setup, allowing you to back up information, save it for the long term, or share it with others easily.

Another key benefit of using an external target is that by offloading data to it, you free up space on your PixStor system, enabling you to perform more tasks and improve overall efficiency.

Filesystem

A filesystem is like a digital storage organizer where all files are kept and managed. It provides a way to arrange and control how data is stored and accessed on storage devices, such as hard drives, solid-state drives, or cloud storage.

A cluster is a group of interconnected systems that work together as one. Each system in the cluster is called a node (a separate computer or device that contributes to the cluster's tasks).

In a clustered environment, the filesystem is usually shared across all (but not necessarily all) of the nodes, allowing any of them to access and change files as needed. Effective filesystem management ensures that data is organized efficiently, makes it easy to retrieve information, and helps maintain overall system performance.

G

GPFS (General Parallel File System)

Note: Definition not approved !!!

GPFS, or General Parallel File System, is a high-performance storage solution designed for clustered environments (groups of interconnected systems that work together to perform tasks).

Unlike a standard file system, which typically allows only one user or process to access a file at a time, GPFS enables multiple nodes to read from and write to the same set of files simultaneously. This parallel access significantly boosts efficiency, especially in scenarios where large datasets are involved.

GPFS is optimized for concurrent file operations, ensuring that users experience minimal delays and can manage large amounts of data effectively without performance degradation. This makes it particularly well-suited for tasks that require fast and reliable data sharing among many users or processes.

Grafana

Grafana is an open-source analytics and monitoring platform that provides dynamic visualizations of system performance metrics. It enables users to create and customize dashboards with various data representations, such as graphs, charts, and alerts.

Grafana collects real-time data on key performance indicators (KPIs) (metrics used to measure the success or performance of a system), including task completion rates, queue lengths, and system latency (the time delay between a request and its response). This allows for effective monitoring and troubleshooting of system operations.

Gateway Node (gw)

A Gateway Node is responsible for providing connectivity to users and applications. It typically runs protocol services such as SMB3, NFSv3, and NFSv4 to enable access to the cluster’s data.

Note: Protocol services are methods that define how data is transferred between devices or applications. They ensure different devices or programs can communicate with each other and share information in a standardized way. Examples include SMB3 (used for file sharing) and NFS (used for accessing files over a network).

H

Hub Server

The Hub Server serves as the central management component of the system. It coordinates tasks and workflows while facilitating communication between various services, including a database, a REST API for handling user requests, a web-based interface for automated interaction, and a task scheduler for automated operations.

M

Management Node (mn)

The main node responsible for overseeing the entire cluster. It typically runs critical functions like the Hub server, search functionality, and analytics tools. The management node coordinates tasks and manages other nodes in the cluster.

Ngenea Node (ng)

A specialized node within the cluster that is responsible for transferring data between the cluster and external storage locations, such as cloud services (e.g., AWS, Google Cloud). The Ngenea Node manages data migration and backup processes without affecting the performance of other nodes in the cluster, including gateway nodes.

Ngenea Worker

A worker is a software component that runs on designated nodes within a network. Its main role is to perform tasks assigned to it by a central server known as the Hub. These tasks often include:

- Managing Files: Primarily involves pushing data to the cloud and retrieving it, as well as organizing, moving, or deleting files as needed.
- Processing Data: Analyzing or transforming data to extract useful information.
- Changing System Settings: Modifying configurations to improve performance or functionality.

Once the worker finishes its tasks, it sends the results to Redis, from which the Hub retrieves them to coordinate overall operations and monitor the status of each worker.

Nginx

Nginx is software that serves as both a web server and a load balancer. It manages incoming user requests and directs them to the right internal service within the Hub system. For example, when a user accesses the Hub’s web interface, Nginx ensures that their request is sent to the correct part of the system to get the needed response. This helps improve performance and ensures that requests are handled efficiently.

Node

A node is an individual unit within a cluster. Each node has specific roles, such as storing data, managing the system, or allowing user access. Nodes work together in the cluster to ensure that data is processed and stored efficiently, helping the entire system function smoothly.

NVMe Node

An NVMe Node is a type of storage node that uses NVMe (Non-Volatile Memory Express), a fast data transfer technology, for quicker access to data. Instead of traditional storage connections like SAS cables, it connects using high-speed Ethernet, allowing for much faster data services.

P

PixStor Cluster

A PixStor Cluster consists of multiple interconnected nodes that function together to manage and store data efficiently. Each node within the cluster has distinct responsibilities, such as data management, storage, or facilitating access for users and applications.

Prometheus

Prometheus is a monitoring tool that collects and stores data about the performance of a system, such as CPU usage, memory consumption, and task processing times. It gathers this information over time and allows administrators to analyze it to track the health and performance of the system. Prometheus also makes this data available for visualization tools like Grafana, which helps users create dashboards to easily see how the system is performing.

R

RabbitMQ

RabbitMQ is a messaging tool (broker) that helps different parts of an application communicate. It allows the Hub server to send tasks to workers by organizing those tasks into a “queue.” Workers pick up tasks from the queue, process them, and then send a message back to RabbitMQ when they are done. This ensures tasks are handled efficiently, even if there are many tasks or workers operating at the same time.

Redis

Redis is a fast, in-memory database that stores data in a key-value format. It is commonly used for temporary data, like caching results or tracking tasks within a system. In addition to storing data, Redis facilitates communication between different parts of the system, such as the Hub server and workers, ensuring tasks are completed and results are shared quickly and efficiently.

For example, task queues are pushed through a Redis queue by default, demonstrating its role in managing task distribution and communication within the system.

S

Salt

Salt, also known as SaltStack, is a versatile tool used for managing and automating the configuration of multiple computers (or “nodes”) in a network. It ensures that all the nodes in a cluster have the same settings, software, and updates by applying changes across the entire system.

For example, if you need to install software or update configurations on all nodes, Salt allows you to do this centrally and efficiently, without having to manually update each node one by one. This helps keep all systems in sync and simplifies management tasks.

Search Backend

The part of the system that allows users to search for files and metadata (information about files) within the cluster. The search backend indexes certain files in the system so that users can quickly locate what they are looking for.

Note: “Search Backend” is a generic term that could refer to either PixStor Search or PixStor Analytics, depending on how the Hub is configured.

Site

A Site is a collection of one or more nodes (which are individual units that can process and store data) working together within a single PixStor cluster (a system that combines multiple nodes to manage large amounts of information).

Each Site has specific responsibilities, such as managing tasks, processing data, and transferring data to and from external sources. In simpler terms, a Site is a team of nodes that collaborates to handle and organize data efficiently within the PixStor system.

Snapdiff Discovery

Snapdiff Discovery is a method for finding changes in files or folders by using saved point-in-time images (snapshots) of them. Instead of checking every single file each time, it compares these snapshots to see what has changed. This makes the process more reliable and traceable, especially when there’s a lot of data to look through. In simple terms, it helps identify differences without having to search everything all over again.

Storage Node (sn)

A storage node is a part of a cluster that connects directly to storage devices like hard drives. Its main job is to manage how data is saved and accessed. In simple terms, it makes sure that information can be stored and retrieved quickly and easily within a larger group of connected devices.

W

Web UI

A web UI, or web user interface, is the part of a web application or online system that you see and interact with. It allows you to manage tasks, check the status of your files, and use various tools—all through your web browser.

Changelog

```
2.11.1: 2026-08-13
==================

Bug
---

HUB2-3931 - Fix for JWT token issue where Iris is on Hub node on
primary site, in clusters with one or more nodes on the primary
site

Documentation
-------------

HUB2-3938 - Add warning about Hub < 2.4 upgrade to docs


2.11.0: 2026-08-06
==================

Feature
-------

HUB2-2389 - Site Maintenance Mode
HUB2-3088 - Job pre and post actions
HUB2-3292 - Custom Metadata Policies
HUB2-3297 - Python 3.12 support
HUB2-3399 - Ability to manipulate site-level excludes in the GUI
HUB2-3768 - Policy Generator Improvements

Improvement
-----------

HUB2-3461 - Match daterange picker styling to the hub UI colors
HUB2-3589 - Document empty directory behaviour for recursive
workflows
HUB2-3674 - Postgres container start up now includes a try/except
```

```
retry block
HUB2-3885 - Resolve issue with differing unit sizing in Spaces
component

Bug
---

HUB2-2551 - Docs do not reflect check_sync_state behaviour in DAG
workflows causing job failures
HUB2-3240 - The iris_notify queue can now be selected for non-iris
tasks
HUB2-3453 - Fix the styling problem for filebrowser filter
checkbox
HUB2-3485 - Hubmetrics button can now be enabled when using
external_rabbit mode as well as redis
HUB2-3760 - Error updating site proxy nginx
HUB2-3763 - 500 error trying to load task results for
get_config_for_keys
HUB2-3793 - Delete from target task ignores skipped files
HUB2-3814 - Cannot wipe external db if containers are restarting
HUB2-3816 - Cannot submit workflow when discovery_options is set
to null
HUB2-3862 - Space create task error when pixstor is busy
HUB2-3873 - Space quota not updating in UI Policy scope wizard
HUB2-3903 - Recall task fails with "file no longer offline"


2.10.2: 2026-05-25
==================

Bug
---
HUB2-3812: Delete from target task no longer ignores skipped files
HUB2-3760: Appropriate errors messages are populated in hub logs
when a site refresh is required
HUB2-3814: External databases can now be wiped even if containers
are restarting
HUB2-3816: Workflows can now be submitted when discovery_options
is set to null


2.10.1: 2026-05-06
==================

Bug
---
HUB2-3753: Invalid site_id field set on Samba shares


2.10.0: 2026-05-01
==================

Feature
-------
```

```
HUB2-3293 - Database Autobackup
HUB2-3295 - Job Alerting
HUB2-3294 - High Availability
HUB2-2776 - Per-Site Shares

Bug
---
HUB2-3487 - Remove outdated documenation for Hub Messaging Queue
Configuration
HUB2-3494 - Removed repeated log messages when a removed queue is
still defined in the worker
HUB2-3542 - Admin users can no longer delete themselves from the
Users group via API only
HUB2-3585 - Ensure that only globbed folders are matched for the
UI include/exclude folders
HUB2-3706 - Multiple "all" rules cause a recursive dag workflow to
replicate its work
HUB2-3710 - Job stuck ongoing when tasks are skipped
HUB2-3735 - Intermittent hang off celery container due to locked
thread in redis-py

Improvement
-----------

HUB2-1078 - Provide ability to switch Search to Analytics backend
and visa-versa in the Global Settings UI
HUb2-2894 - Document Site Include/Exclude UI controls
HUB2-2997 - Ensure the browser back button now works for Spaces &
File browser
HUB2-3265 - Add a new Job state for all tasks complete with some
errors
HUB2-3373 - Update Docker to v25
HUB2-3383 - Externalise Docker setup to docker-compose.yml for
ngeneahubctl
HUB2-3413 - Update to Postgres v17
HUB2-3419 - Provide export including all Job history
HUB2-3490 - New command "ngeneahubctl database wipe" for external
PostgreSQL
HUB2-3513 - Certificates are correctly cleared when the hub DB is
wiped
HUB2-3520 - Snapdiff Jobs now support a FAILING status
HUB2-3527 - Ensure UI respects the new resubmittable flag
HUB2-3608 - Refreshed plugin development guide added to
Programming Guide
HUB2-3701 - Update internal database and ephemeral containers to
postgres:17.9-1.arcapix
HUB2-3725 - Add TCP Keepalive to sockets
HUB2-3726 - Add connection resilience settings for Hub worker


2.9.4: 2026-02-16
=================
```

```
Bug
---
HUB2-3535 - readonly users can delete other users via API only
HUB2-3538 - readonly users can upgrade to admin privileges via API
only
HUB2-3539 - readonly user can change group permissions via API
only
HUB2-3540 - ensure an admin user cannot remove themselves from the
Administrators group via API only
HUB2-3541 - a non-admin user can deactivate other user accounts
via API only


2.9.3: 2026-02-02
=================

Feature
-------
HUB2-2290 - Ability to force apply ngtarget advanced keys
HUB2-2834 - Manage default space location
HUB2-3463 - add dynamo.tasks.delete_from_target
HUB2-3486 - Add support for Ngenea HSM 1.33 params

Improvement
-----------
HUB2-761 - Show password with eye icon
HUB2-814 - Ensure user is aware when disabling ngenea target
HUB2-2941 - Hubmetrics button added to main menu bar
HUB2-3053 - Select pools and sites by default
HUB2-3395 - Add default workflow for manual Iris file notification
HUB2-3429 - Update space modal close on save
HUB2-3430 - Close wizard after adding space
HUB2-3431 - Keycloak logout on password change

Bug
---
HUB2-3068 - Null discovery tasks stuck ongoing after completion
HUB2-3195 - Resolve target help link issues
HUB2-3368 - CredentialsFile is not created/updated for GCS targets
HUB2-3396 - File not found during snapdiff list cleanup
HUB2-3434 - Resolve react router dom dep issue
HUB2-3437 - Recursive workflow doesnt recursive without directory
filter rules
HUB2-3452 - Fix lodash dependency issue
HUB2-3456 - Folder includes items under it when creating a
schedule in filebrowser
HUB2-3457 - Jobs reporting as STARTED after all tasks have
completed
HUB2-3483 - Remove from target doesn't detect directories
HUB2-3484 - Error when patching sites on an iris enabled space


2.9.2: 2026-01-12
=================
```

```
Feature
-------
HUB2-3395 - Ability to manually trigger Iris notification


2.9.1: 2025-10-18
=================

Bug
---
HUB2-3273 - Hub tries to fetch thumbnails and proxies when Pixstor
Search is disabled
HUB2-3274 - Files not reported as SKIPPED when skipped by
check_sync_state
HUB2-3346 - Snapdiff schedules can't find lastsnap file after
upgrade


2.9.0: 2025-10-22
=================

Feature
-------

Iris enablement for spaces
Iris Single Sign On (SSO) using Keycloak
External service support for Postgres, RabbitMQ and Redis
Ngenea HSM 1.32 support

Improvement
-----------
HUB2-2955 - Add subtitle explanations for using site includes/
excludes modals
HUB2-2956 - Cancelling is now referred to as "Dequeuing"
HUB2-2996 - The generated skip files are now included in the
snapdiff automatic cleanup
HUB2-2998 - Increase max width allowed for site and summary column
content, in alerts table
HUB2-3012 - Added an additional snapshot rotation safety check
HUB2-3055 - Removed the skip button from Spaces bucket page for
consistency
HUB2-3086 - Plugin templates are now created with up to date
dependencies
HUB2-3188 - Fixed typo about wording for "Administered Spaces" in
UI
HUB2-3152 - Schedules now accept longer schedule names

Bug
---
HUB2-2853 - Submitting policy creation with existing name no
longer creates a duplicate policy
HUB2-2927 - Jobs are no longer left indefinitely pending when
workflow and site include or excludes are used
```

HUB2-2992 - Worker unmanage task no longer uses deprecated batch_size parameter
HUB2-3052 - Ngenea worker no longer takes over 20 minutes to shutdown
HUB2-3057 - Updating a schedule in the UI no longer deletes and recreates the subscribed workflow
HUB2-3082 - API now provides the correct response when a folder does not exist in all sites
HUB2-3101 - OpenSearch now respects the provided timeout fom the hub
HUB2-3107 - Storage analytics no longer reports “ingest needed” when the Space has been ingested
HUB2-3149 - "hydrate files on site" can now be toggled off for Reverse Stub
HUB2-3153 - Ngenea targets are now populating correctly on the dropdowns for backup configurations
HUB2-3172 - Handle the case where proxies of the search result is not available with analytics backend
HUB2-3179 - StorageKey is now applies to all target sites when creating AWS targets
HUB2-3203 - A direct link to a path in the Filebrowser now correctly loads regardless of the amount of spaces in the hub
HUB2-3238 - Snapdiff jobs now correctly use their assigned queues

2.8.3: 2026-02-17
==========
Backport

HUB2-3544 - readonly users can delete other users via API only
HUB2-3545 - readonly users can upgrade to admin privileges via API only
HUB2-3546 - readonly user can change group permissions via API only
HUB2-3547 - ensure an admin user cannot remove themselves from the Administrators group via API only
HUB2-3548 - a non-admin user can deactivate other user accounts via API only

2.8.2: 2025-11-18
==========

Backport

HUB2-3179 - StorageKey is now applies to all target sites when creating AWS targets
HUB2-3346 - Snapdiff schedules can't find lastsnap file after upgrade

2.8.1: 2025-08-21
==========

```
Improvement
-----------
HUB2-3072 - Add support for new Ngenea HSM 1.32 configurations
HUB2-3080 - Archive Ngenea targets on delete

Bug
---
HUB2-3052 - Ngenea worker no longer hangs during shutdown
HUB2-3069 - Support for Ngenea targets without bucket in name
HUB2-3081 - The hub no longer provides the "directory is deleted"
warning when the folder does not exist in all sites
HUB2-3082 - The API no longer throws "path does not exist" error
when a folder does not exist in all sites
HUB2-3101 - The configured timeout is now respected for OpenSearch


2.8.0: 2025-09-05
=================

Feature
-------
* Iris Beta

Improvement
-----------
HUB2-1720 - Improve snackbar appearance
HUB2-2735 - Restart ngeneahub critical containers in the event of
a crash
HUB2-2308 - Update docs to include raising the lambda timeouts
HUB2-2846 - User Guide improvements

Bug
---
HUB2-2243 - Removing filter from failure states does not remove
filter
HUB2-2782 - Fix the issue of labels not being truncated when they
are too long
HUB2-2885 - Don't error on remove remote xattr for top level
directory
HUB2-2935 - Can't disable snapshot schedule on filesystem spaces
HUB2-2946 - Could not update global settings because of
initial_uid and initial_gid error
HUB2-2978 - Space unexpectedly deleted when adding sites via UI
after creating space via API
HUB2-3030 - Files with epoch 0 timestamp cause file browser to
fail
HUB2-3032 - API modified schedules now pickup new settings


2.7.2: 2025-10-20
=================

Backport
--------
```

HUB2-3238 - Snapdiff jobs now correctly use their assigned queues

2.7.1: 2025-08-21
=================

Backport

HUB2-3069 - Support for Ngenea targets without bucket in name
HUB2-3080 - Archive Ngenea targets on delete

2.7.0: 2025-06-24
=================

Improvement

HUB2-620 - API: Policies can now be run immediately after being created
HUB2-948 - Policies are no longer restricted to mmfs1 named file systems
HUB2-2183 - API: The default location of Spaces can be configured for all Sites
HUB2-2316 - The global settings is synced after an 'apply settings' job has been run
HUB2-2337 - The visual contrast for grouping boxes and dividers has been improved
HUB2-2338 - Spaces can now be viewed in table layout
HUB2-2339 - Job tables can now be sorted for job creator, site, queue, workflow, queue
HUB2-2340 - Workflow batch and item GB size are now available as Site options
HUB2-2342 - Timeout settings have been added to the Global Settings page
HUB2-2672 - All automated schedule times are now shown in local time, not UTC
HUB2-2677 - Ngenea Targets now have validation on their advanced key options
HUB2-2794 - File browser timeouts can now be configured in the Global Settings page
HUB2-2812 - The path when you pick a location in a space space now starts at the space mountpoint
HUB2-2829 - Policy based workflows renamed from 'Tiering' to 'Policy-based Tiering'
HUB2-2877 - Added GUNICORN_WORKERS as a synonym for WORKERS configuration key, the WORKERS configuration is now deprecated
HUB2-2878 - New GUNICORN_THREADS setting to increase API performance
HUB2-2890 - RabbitMQ container is now migrated to the configured destination when DATA_DIR is set

Bug

HUB2-1520 - Newly added sites now pick up NAS users and groups
HUB2-1699 - Hub metrics no longer fails to load due to an authentication error
HUB2-2733 - Hubmetrics dashboard no longer fails to load
HUB2-2746 - Site snapshot times no longer contain timezones
HUB2-2751 - When renaming a Site, the original name is autofilled
HUB2-2764 - Ngenea target name validation now does not allow names containing "."
HUB2-2787 - Rotate snapshot completion time is now set correctly
HUB2-2796 - Rotate tasks no longer errors when attempting to store its results
HUB2-2818 - Transparent recall no longer throws a traceback despite sucessful recall of files
HUB2-2827 - Snapdiff workflows no longer assume old queue name format for custom plugins
HUB2-2830 - Updating an NFS share in the UI no longer causes share deletion on Pixstor
HUB2-2835 - Job engine now correctly waits for the parent task in high speed workflows
HUB2-2836 - Workflow discovery option can now be patched to be null
HUB2-2845 - Creation of a discovery schedule no longer fails to create due to an existing schedule sharing the same scheduled time
HUB2-2848 - A deleted schedule no longer remains on the dashboard when a schedule is deleted
HUB2-2849 - day_of_month is no longer edited within the schedule picker
HUB2-2884 - Fix "key error" for create NAS user and update NAS group members jobs
HUB2-2915 - Added safety measures to prevent deadlock failures in snapdiff jobs
HUB2-2917 - Ensure Next button is not disabled in Network settings Page when configuring a newly added Site
HUB2-2924 - Added safety measure for looking up task_id during DAG callback
HUB2-2927 - Jobs are no longer left indefinitely pending when workflows and site include or excludes are used

2.6.1: 2025-08-21
==========

Backport

HUB2-3069 - Support for Ngenea targets without bucket in name
HUB2-3080 - Archive Ngenea targets on delete

2.6.0-2: 2025-05-27
==========

Documentation updates

```
2.6.0: 2025-04-30
==================

Feature
-------
HUB2-1651 - Simplified Ngenea Targets for Spaces
    * Targets and buckets can now be created during Space creation
    * Targets and buckets are now separately defined and can be
re-used
HUB2-1975 - Schedule Management Interface
    * There is now a dedicated page for Schedule Management
    * This controls both workflow schedules and Policies
    * There is now an additional single endpoint for serving a
read only listing of schedules and policies
HUB2-2126 - Added workflow for ngenea download only mode

Improvement
-----------
HUB2-628 - Salt related tasks timeout is now configurable
HUB2-1699 - Hub metrics no longer requires an additional login
HUB2-2303 - Policy file condition selection now prepends dot
automatically
HUB2-2304 - Space endpoint can now be filtered via site_id
HUB2-2372 - Targets are not longer required to be backup_only
HUB2-2487 - The same path is now allowed to be used by multiple
non-snapdiff schedules
HUB2-2489 - The space endpoint now returns 'is_ready'
HUB2-2329 - ngenea-worker plugins list now provides the version of
all installed plugins
HUB2-2531 - LDAP TLS Certificate support
HUB2-2646 - Global settings now has a quick link when attempting
to edit a schedule when all are disabled
HUB2-2689 - Non snapdiff job task responses have improved
performance
HUB2-2701 - Job details page now loads statistics separately
making page loading much faster

Bug
---
HUB2-1932 - Schedules from other Spaces are no longer visible
under a Space
HUB2-1990 - When searching space specific RBAC no longer returns
no results
HUB2-2087 - Spaces can now correctly be re-added to be backed up
HUB2-2229 - AD groups now support whitespaces in names
HUB2-2243 - A settings job is no longer created when changing the
site short code
HUB2-2252 - Jobs can no longer prematurely report as done
HUB2-2315 - Transparent recall through Hub no longer fails with
path containing certain characters
HUB2-2327 - Not submitting group name no longer returns a 500
error
HUB2-2404 - When immediately disabling the sync schedule on a
newly created space, it now correctly disables
```

HUB2-2415 - ngenea-worker plugins uninstall can now remove multiple plugins at once
HUB2-2418 - ngenea-worker plugins install now correctly installs only the specified plugins
HUB2-2438 - Policy exclude conditions are no longer treated the same as includes
HUB2-2432 - Removing the task filter now works as expected in the task table
HUB2-2456 - dynamo.tasks.rotate_snapshot no longer always shows as NULL
HUB2-2534 - Send to site no longer shows files both as "processed" and "pending" upon job completion
HUB2-2540 - Trailing slash in site settings no longer causes 404
HUB2-2662 - File browser in schedule wizard now shows restricted items
HUB2-2539 - Changing the pool from the default now correctly refreshes
HUB2-2656 - Fixed issue with 'sssd' not applying when applying domain settings in shares
HUB2-2658 - Deleting a target via "delete bucket" now fully clear salt pillar
HUB2-2669 - Non admin users can now update their profile
HUB2-2711 - Transparent recall error on job id expiry check
HUB2-2712 - GCS targets can now switch between credentials_json and credentials_file in a single PATCH request
HUB2-2718 - Space cards sizes are now consistent
HUB2-2719 - Ensured that shares cannot allow all_squash and no_root_sqaush
HUB2-2767 - Fixed issue with navigating Buckets using Azure based targets
HUB2-2769 - Snapdiff cache files are now consistent between multiple nodes

2.5.5: 2025-03-04
=================

Feature

HUB2-2344 - Workflows to enable partial recall and importing bytes

2.5.4: 2025-02-28
=================

Improvement

HUB2-2547 - Allow users to specify preferred queue for a workflow via API using "preferred_queue"

Bug

HUB2-2520 - Automated queue cleanup task no longer throws an

```
exception on startup
HUB2-2541 - Space creation location picker now returns correct
folders when not browsed prior in the file browser


2.5.3: 2025-02-19
=================

Bug
---
HUB2-2611 - Ensured shares defined before version 2.5.3 could be
removed from the salt pillar
HUB2-2545 - Renaming an NFS share no longer creates a new match-
all share
HUB2-2588 - Creating a Samba share no longer results in endless
looping jobs
HUB2-2600 - Existing NFS clients are no longer updated when new
NFS client is added
HUB2-2601 - Editing an existing NFS share no longer has the
potential to create new match-all shares


2.5.2: 2025-02-10
=================

Improvement
-----------
HUB2-2531 - Hub login support for LDAPs and STARTTLS


Bug
---
HUB2-2545 - Renaming NFS shares no longer creates a new wildcard
share


2.5.1-1: 2025-01-13
===================

Bug
---
HUB2-2455 - Fixed tasks unexpectedly exiting while attempting to
log job statistics


2.5.0-2: 2024-12-13
===================
Updated Ngenea HSM dependency to 1.30.1-1


2.5.0: 2024-11-13
=================

Feature
```

```
-------
HUB2-1965 - Support for setting a Spaces permission mode
HUB2-2198 - ngeneahubctl now supports database export
HUB2-2199 - ngeneahubctl now supports database import
HUB2-2280 - Provide DownloadOnly workflow in file browser

Improvement
-----------

HUB2-1333 - Started tasks no stuck after a sudden worker exit
HUB2-1519 - Improve scanning performance for policies against
single spaces
HUB2-1643 - ngclient now accepts relative paths
HUB2-2037 - Ngenea targets can be unlinked from all sites
HUB2-2068 - Added additional safety guards to ngeneahubctl wipe
HUB2-2099 - The Site management page now displays when sites are
configured on their respective nodes
HUB2-2119 - ngclient now identifes authentication issues
HUB2-2240 - Stopping/restarting a worker causes tasks to get
killed and lost
HUB2-2286 - Suppport ConfigFile, LocalFileRegex,
LocalSymlinkTargets, RemoteLocationXAttrRegex, StorageKey in
ngenea target advanced keys
HUB2-2294 - NFS share now display with multiple non-CIDR clients
in the UI
HUB2-2300 - Ensure the email entry is omitted on the users table
if not set
HUB2-2301 - Star character is no longer displayed when Search
query is submitted
HUB2-2305 - Ensure Kerberos connections are bound as authenticated
user
HUB2-2306 - The "Minimize" tooltip for left bar is no longer
incorrectly sized
HUB2-2324 - Workflows that select a secondary site now default to
the "default" queue if not provided
HUB2-2333 - Google cloud storage targets can now use
CredentialsFile as the only credentials provider
HUB2-2335 - When rabbitmq is not the target broker the container
no longer spawns
HUB2-2353 - Either credentials_json or credentials_files can be
used when defining a target with Google Cloud Storage

Bug
---
HUB2-1288 - Fix page scrollbar sizing in Job list page
HUB2-1424 - Ensure Hub cannot overwrite existing users and groups
from pixdjango
HUB2-1460 - Provide values for usage and capacity in Policy UI
when no quota is in place on a space
HUB2-1584 - Resolve issue with per-process lock aquisition in
transparent recall
HUB2-1806 - Ensure migrations to ngenea are limited to one source
pool
HUB2-1937 - The automated job to expire old jobs has been
```

```
optimised
HUB2-1955 - Resolve 500 error when deleting a schedules workflow
HUB2-1983 - NFS shares are now correctly configured with multiple
clients
HUB2-1984 - Resolve traceback observed: AttributeError: 'QuerySet'
object has no attribute 'objects' when using check_sync_state
HUB2-1986 - Ensured every client for the same NFS share have the
same fsid
HUB2-1999 - Resolve GCS bucket creation failure due to mangled
json auth
HUB2-2002 - Recursive navigate now reports failed paths
HUB2-2084 - Remove redis connection log error under normal
operation
HUB2-2174 - Job descriptions are now case sensitive
HUB2-2216 - Non-existing ngenea target now correctly raises a 404
HUB2-2220 - Ensure Space mountpoint is not presented with double
forward slash in the space wizard
HUB2-2224 - reverse_stub task no longer lists "--overwrite-local".
HUB2-2277 - Thread counts can now be explicitly for each default
queue function
HUB2-2279 - Browsing external targets now correctly parse errors
from listing cloud files
HUB2-2283 - Fix inability to change Space card colour
HUB2-2284 - Remote folder not matched of top level folder
selection no longer causes import from site jobs to fail
HUB2-2307 - Shares can now be browsed when Samba allowed_hosts is
null
HUB2-2317 - Target name too long error not correctly reported to
user in UI
HUB2-2318 - Targets now correctly supports StorageKey numerical
strings
HUB2-2322 - The 'invalid character' error message when browsing a
space no longer appears when there are no files to browse

2.4.4: 2025-02-07
=================

Feature
-------
HUB2-2479 - Improved snapdiff based workflows performance
providing snapdiff includes and excludes on the worker

2.4.3: 2024-10-16
=================

Improvement
-----------
HUB2-2245 - Snapdiff based workflow moves now support same named
queues

2.4.2: 2024-10-14
=================

Improvement
```

```
-----------
HUB2-2214 - Transparent recall now supports alternative queues

Bug
---
HUB2-2170 - Resolves LDAP plaintext authentication failure
HUB2-2193 - GPFS Policies scans now run across multiple nodes
HUB2-2209 - Scheduled jobs without a second site now correctly run
HUB2-2225 - All ngenea target keywords are now supported

2.4.1: 2024-10-07
=================

Bug
---
HUB2-2209 - Workflows without a destination site now correct run
from a schedule

2.4.0: 2024-10-04
=================

Feature
-------
Multiple Job Queues
RabbitMQ as an optional broker

Improvement
-----------
HUB2-908 - Limit ngenea target creation to specific sites via API
HUB2-1544 - Move ngenea target settings to dedicated /
external_targets endpoint
HUB2-1640 - Attempt to premigrate files from snapshot even if live
copy doesn't exist
HUB2-1776 - Summary page update for Ngenea bucket configuration
HUB2-1898 - Improve performance of Job no-op filter
HUB2-1970 - Make the job link middle- & right- clickable on "job
is created" popup
HUB2-1989 - API-Keys no longer require an email for creation
HUB2-2005 - Unify order of statuses in Job specific page
HUB2-2117 - Make node heartbeat check interval configurable

Bug
---
HUB2-599 - Workflow schedules will now wait until space creation
completes on all sites to start
HUB2-1382 - The search_keyword argument correctly filters jobs via
ID
HUB2-1641 - Snapdiff tasks no longer skip files not on the live
filesystem
HUB2-1799 - Search filter chips no longer disappear after a search
completes
HUB2-1802 - Policy related jobs are no longers always stuck in
pending
HUB2-1973 - Disable rpm build-id debug files
```

HUB2-1994 - Login page now has correct page title

Documentation

HUB2-1820 - Update existing Plugin docs for new DAG capabilities
HUB2-1866 - Existing worker task options have been updated

2.3.2: 2024-08-23
==========

Bug

HUB2-1893: Reverted; due to causing Bidirectional sync to fail intermittently

2.3.1: 2024-08-09
==========

Improvement

HUB2-1685: Added documented new engine functionality to the worker
HUB2-1016: Double clicking thumbnails within search now opens a full screen preview
HUB2-1457: Fileset snapshots retention period language is improved for international users
HUB2-1895: Snapdiff workflows now support root filesets

Bug

HUB2-1672: Fix engine deadlock issue when running a non null discovery job
HUB2-1704: Shares deleted on the UI are now always deleted on the target system
HUB2-1707: Snapshot schedule update task is submitted when editing shares
HUB2-1762: Settings jobs can now be cancelled
HUB2-1859: Resolve rare race condition when storing task results
HUB2-1865: Ganesha exports now stored with numerical value for allowing persistence on shares reload
HUB2-1893: Snapdiff rotate task now correctly runs after processing task is revoked
HUB2-1916: Resolved flickering on second page of tasks in a job
HUB2-1944: When creating a space, the scheduled job now waits for the space to be completed before starting
HUB2-1957: Resolved race condition in site settings task
HUB2-1985: Deleting a client from an NFS export now removes the share from the site
HUB2-1798: UI automatic file browser refresh no longer display some restricted folders

```
2.3.0: 2024-07-05
==================

Feature
-------

Job Reporting improvements
LDAP and Kerberos integration
Space deletion via the UI
Dynamic job batching
Database size reduction with tunable settings

Improvement
-----------
HUB2-1190: Default locking method options are now applied to all
default workflows
HUB2-1804: Custom plugin docs have been updated to support new
engine functionality
HUB2-1891: Job details page now renders regardless of statistics
being loaded
HUB2-1892: API_TIMEOUT and GATEWAY_TIMEOUT now default 10 minutes
HUB2-1356: Various database size reduction optimisations
HUB2-1644: Recursive jobs now use less calls to mmlsfileset
HUB2-1644: Recursive jobs now no longer navigate into excluded
directories
HUB2-1691: Removed search is halted message when interrupting a
search
HUB2-1700: SMB shares now supports definitions in the hosts to
allow/deny
HUB2-1719: Ngenea worker analytics requests can now have their
timeouts configured
HUB2-1281: Error and failed jobs can now be re-submitted
HUB2-1610: Job descriptions have been added to all non-settings
jobs
HUB2-1621: Users can now to filter paths across all pages when
searching for a related job
HUB2-1462: Installed internal SSL certificates can now be
displayed on the command line
HUB2-1613: Plugins and internal tasks now use jobid consistently
HUB2-1664: Transparent recall is now using the new task engine

Bug
---
HUB2-1895: Snapdiff workflows now support root fileset snapshots
HUB2-1414: Circular symbolic links no longer cause analytics
errors
HUB2-1491: Shares now correctly cannot contain white spaces
HUB2-1683: Users and groups are no longer allowed to contain white
spaces
HUB2-1693: Alerts are now shown immediately
HUB2-1821: Site options are now pre-populated in bidirectional
sync setup
HUB2-1839: Analytics caches are correctly removed on an analytics
```

error
HUB2-1861: Downloading a file list now provides the correct output
HUB2-1550: Fixed the runtime for paused jobs
HUB2-1614: Default spaces are created correctly for filesystems not mounted at the standard default
HUB2-1694: Cancelling a snapdiff with specific timings no longer causes the job to remain in an unknown state
HUB2-1701: Permissions are now correctly applied to the alerts on the the site page
HUB2-1706: Storage metric graphs now correctly display on lower resolutions
HUB2-1737: Site labels now default to their names if not set
HUB2-1464: Custom snapdiff jobs now correctly track progress correctly in job list page
HUB2-810: Space creation no longer fails due to unrelated salt state errors
HUB2-847: Site network settings now allow CIDR values as 0-32 , with 0 included
HUB2-1055: Only one job is now created when spaces are deleted
HUB2-1187: Un-necessary SSL warnings have been removed in worker logs
HUB2-1299: Ensure text in UI tooltips no longer go off the screen
HUB2-1396: Site colour picker shows multiple shades of a colour selected
HUB2-1466: None snapdiff job runtimes are now consistent between summary and detail views
HUB2-1478: Job ids can now be longer than 22 characters
HUB2-1480: Client certificate expiry is now set correctly internally
HUB2-1484: Clicking “x” for site filter chip now behaves consistently
HUB2-1492: Long space names now display correctly in the "location" section of space creation as well as summary screen
HUB2-1493: UI now correctly states the correct maximum name size for spaces
HUB2-1535: Job progress bar in the job listing page now correct reflect progress
HUB2-1540: Job progress bar in the job listing page now shows progress without manual refresh
HUB2-1551: Task details modal tooltip no longer blinks
HUB2-1566: Job "id" field is now correctly set for settings tasks
HUB2-1574: Ordering of jobs in the job listing page when sorting by state is now consistent
HUB2-1587: Internal task non_discovery_rule_filter now shows date started
HUB2-1589: ngclient authenticate now correctly uses the api_secure_verify from config file
HUB2-1603: Schedules for recreated filesets now correctly run
HUB2-1608: The rightbar no longer flickers in the job details page when scrolled down
HUB2-1609: Pending item stats have been removed snapdiff jobs as they do not use them
HUB2-1620: Non UTC datetime in custom DAG tasks now correctly refresh

HUB2-1628: Ensured that on custom plugin install that the internal packages do not change version
HUB2-1629: Populate settings job no longer get stuck in the "NEW" state with an "Unknown" owner
HUB2-1631: Delete space task now runs even if create space hasn't completed yet
HUB2-1635: Transparent recall no longer fails if existing transparent recall job does not exist
HUB2-1666: The filter bar now always displays correctly in the file browser
HUB2-1669: The current view is now retaining on jobs detail page refresh

2.2.5: 2024-06-21
==========

Improvement

HUB2-1852 - Ensured that storage analytics can have its request timeout configured

Fix

HUB2-1852 - Pass API_TIMEOUT from the hub sysconfig file to the API container

2.2.4: 2024-05-31
==========

Improvement

HUB2-1644 - Recursive navigate tasks no longer call mmlsfileset more than once

Bug

HUB2-1680 - Task failures no longer stop downstream tasks to stay in PENDING

2.2.3: 2024-05-21
==========

Improvement

HUB2-1625 - The API wide timeout can now be configured

Bug

HUB2-1626 - The job details UI now waits for job results regardless of request duration
HUB2-1667 - The job list UI now waits for results regardless of

request duration
HUB2-1682 - Job related web socket connections now waits for connection regardless of duration
HUB2-1619 - Jobs with the state "Failing" no longer disappear from jobs list page

2.2.2: 2024-05-06
=================

Bug

HUB2-1630 - Fixed issue with bidirectional sync re-submission

2.2.1: 2024-04-29
=================

Bug

HUB2-1630 - Transparent recall functionality now correctly submits with new engine

2.2.0: 2024-03-28
=================

Feature

UI Colours
Search Tags
Custom Attributes for SMB
New DAG Job Engine

Improvement

HUB2-291 - Better wrap long space names
HUB2-365 - Different colors for task table chips instead of "gray" ones
HUB2-368 - Fix scrolling arrows showing up by default on rightbar
HUB2-652 - Utility for translating schedule object to user
HUB2-770 - Automatically name policy condition group titles
HUB2-863 - Task stats should apply the filter on job details page
HUB2-895 - Provide API support for editing search tags
HUB2-990 - Provide the ability to filter site alerts by severity
HUB2-1032 - Search: Support extensions starting with dot for core.extension filter
HUB2-1035 - Enforce unique space mount point (API)
HUB2-1072 - Ensure group titles wrap better
HUB2-1084 - Swap files above tasks in job details
HUB2-1089 - Provide the ability to set the number of executed policy threads in the UI

HUB2-1103 - Add buttons for pausing and resuming a job
HUB2-1104 - API response for job Owner is different
HUB2-1105 - Adjust Job details UI labels to new DAG statistics
HUB2-1107 - Adjust websocket responses to conform with new serializers
HUB2-1112 - Ensure NAS components are not visible if all sites in AD mode
HUB2-1162 - Site filtering for policies
HUB2-1213 - Ensure apbackup chip colours are consistent and correct site colour when input is changed
HUB2-1218 - Make top bar green
HUB2-1219 - Implement yellow progress bar across wizards
HUB2-1220 - Provide the ability to set the colour of the Site Chip in the Create Site wizard
HUB2-1222 - Updated colourful left bar menu buttons
HUB2-1223 - Updated login page presentation
HUB2-1224 - Cyclic colours for space cards
HUB2-1225 - Provide the ability to set the colour of the Space Card in the Space Wizard
HUB2-1226 - General CSS changes for button/checkbox/slider etc. elements
HUB2-1227 - Add SMB Custom Options Panel to Create Space Wizard
HUB2-1228 - Add SMB Customs Options Panel to Update Space Modal
HUB2-1230 - Modify multiselect dropdown component for search of a tag
HUB2-1237 - Remove feature flags for stable features
HUB2-1256 - Provide API support for Site chip colours
HUB2-1259 - Increase intervals for site refreshes
HUB2-1261 - Provide alphabetical ordering for managed keys
HUB2-1263 - Add tag selector filter to Search filters (in main search filter list)
HUB2-1264 - Add tag editor to search side panel for asset
HUB2-1282 - Grant metadata update permission to worker service user
HUB2-1291 - Switch site chips to double colour chips
HUB2-1294 - Don't display muted alerts in the GUI
HUB2-1310 - Update Hub favicon
HUB2-1311 - Add danger notification
HUB2-1312 - Make rightbar stay at the same place on scroll for global search page
HUB2-1316 - Hide resubmit button for settings tasks
HUB2-1328 - Better render "Date started" for cancelled tasks
HUB2-1334 - Ensure backup UI components are only displayed if the site has nodes configured with the apbackup role
HUB2-1346 - Implement database expiry page in global settings page
HUB2-1358 - Make sticked sidebar on default for all elements
HUB2-1376 - Re-expose grafana dashboard
HUB2-1377 - Provide role lookup for apbackup feature
HUB2-1378 - Add filter by job creator inside jobs
HUB2-1389 - RPM of public facing html docs
HUB2-1399 - Change text inside of tags multi-select
HUB2-1401 - Remove Default Location toggle from space create wizard
HUB2-1402 - Where a site only has one pool, auto-select it in the

space creation drop-down
HUB2-1411 - Document swagger URL
HUB2-1417 - Add a base class to all non system worker tasks that check if it can run
HUB2-1418 - Adjust the cancel process to use redis and change the cancelling states
HUB2-1436 - Add the enable_plugin setting to the plugin docs
HUB2-1438 - Type filter updates on Jobs page for DAG jobs
HUB2-1442 - Job details show job status as Pausing when the job is in state Pausing
HUB2-1446 - Remove settings task types from workflow filter list in the UI
HUB2-1447 - Support for filtering on CANCELLING and PAUSING states in the UI
HUB2-1450 - Document the /hubmetrics endpoint
HUB2-1469 - Task stats should include PENDING task counts
HUB2-1496 - Provide clarification that File Stats processes more than Files
HUB2-1498 - Rename the "In-Progress" file stats label to "Pending"
HUB2-1530 - Rename Job Type to Workflow in job creator

Bug

HUB2-395 - Prevent setting space mount point to existing path
HUB2-473 - Deleting any of the site config fields and submitting does not delete the fields
HUB2-581 - Resolve an issue whereby a resubmitted job displays Job Creator unknown
HUB2-703 - Client key name max length error
HUB2-807 - Invalid date message on filebrowser table
HUB2-812 - Resolve an issue whereby full screen for task logs does not display in full screen
HUB2-892 - Resolve an issue whereby a space cannot be deleted using the python shell without deleting snapshot_schedule first
HUB2-928 - Fix timeout message in search page
HUB2-946 - Small Browser Window Stylesheet Issue
HUB2-997 - Task type filter options are not loaded for some of the system jobs
HUB2-1006 - Ensure that site creation summaries are accurate
HUB2-1061 - Provide job not found page
HUB2-1079 - Resolve an issue whereby full screen video preview does not fullscreen
HUB2-1083 - Resolve an issue whereby joined sites have no job site labels
HUB2-1086 - Resolve an issue whereby key names can be obscured
HUB2-1136 - Very long file name takes all the width in file browser table
HUB2-1144 - Resolve an issue whereby a invalid nginx config can cause frontend service interruption
HUB2-1146 - Error handling error response from search
HUB2-1184 - Cursor jumps to end of line as you type in the space name textbox
HUB2-1185 - Job gets stuck in processing if permissions on ngmigrate do not allow execution

HUB2-1189 - Remove unneeded warnings in the worker logs
HUB2-1232 - Cannot add API key via UI if profile fields aren't set
HUB2-1242 - Resolve an issue whereby all NAS interfaces cannot be removed
HUB2-1245 - Resolve an issue whereby the Username is truncated in the top right
HUB2-1260 - Resolve an issue whereby the Sites page shifts off screen when the left bar is expanded
HUB2-1262 - Resolve an issue whereby flickering is observed when loading pages
HUB2-1279 - 500 Server error when trying to verify an auth token
HUB2-1284 - Resolve an issue whereby SAMBA configuration entries are duplicated on update or deletion
HUB2-1285 - Unable to save NAS settings
HUB2-1289 - Fix duplicate job detection preventing Samba keys from being updated more than once
HUB2-1290 - Resolve an issue whereby a basic auth login prompt is presented
HUB2-1293 - Pre-set public url is not populated on create site wizard
HUB2-1301 - Setting a schedule to 1 hour causes it to run every minute
HUB2-1307 - Unable to create NAS users after deleting NAS user
HUB2-1314 - Create and update Space wizard error if no ngenea targets have been defined
HUB2-1319 - Fix validation email for create user wizard & update user modal
HUB2-1322 - Migration policy with three pools skips the second pool
HUB2-1323 - Cannot create a policy with a schedule
HUB2-1324 - Site settings - existing extra domain can not be deleted & validation needed for extra domain without name
HUB2-1326 - public_url = None crashes UI for unconfigured site
HUB2-1339 - Fix "get() returned more than one Server" issue
HUB2-1359 - Fix height of the multi-select dropdowns to have valid minimum height
HUB2-1360 - Selecting a tag for an item, assign it for all items with exact path, which is not unique
HUB2-1362 - Job status appears as Ongoing for job in state UNKNOWN_FAILURE
HUB2-1366 - Resolve an issue whereby ngenea target secrets are visible in log files
HUB2-1367 - Spaces can suddenly become unlinked where multiple "sites" are defined for a given PixStor cluster
HUB2-1379 - Filter input inside jobs is not working
HUB2-1380 - transparent_recall does not honour api_secure_verify
HUB2-1384 - Cannot change snapshot schedule on a space
HUB2-1394 - SettingsTask deduplication logic prevents legitimate jobs from running
HUB2-1400 - Redis will consume all available ram on a system over time
HUB2-1403 - Fix settings refresh
HUB2-1404 - Timestamp rules for policy weighting are incorrect
HUB2-1412 - Space settings loads a blank screen if site label is

not set
HUB2-1413 - Job submitter does not show anything if site label is missing
HUB2-1415 - Make sure the UI does not break when workflows are visible=False
HUB2-1416 - Link to space path lost on login
HUB2-1420 - Job submitter does not allow running workflow
HUB2-1423 - DAGs submit custom tasks to the wrong queue
HUB2-1425 - Tasks filter not applying to live updates
HUB2-1426 - Unable to disable site apbackup
HUB2-1434 - Tasks go into state "STARTED" even though they are still "PENDING"
HUB2-1441 - Ngenea worker service cannot start up when no filesystem exists
HUB2-1448 - Possible to cancel PAUSING jobs via the jobs overview
HUB2-1455 - Job details rightbar shows ongoing when the job is in cancelling state
HUB2-1467 - Job stats are not updated correctly when a DAG job is cancelled
HUB2-1477 - Docker images manifest file is missing from rpms
HUB2-1494 - UI is redirected to /undefined
HUB2-1495 - Only in progress File stats are updating
HUB2-1497 - Clicking the files stats number retrieves no content into the pop up JSON modal
HUB2-1499 - job task list IDs are not being sorted correctly
HUB2-1500 - API request should call refresh token once expired, rather than logging the user out
HUB2-1509 - Long job descriptions break the UI on job table
HUB2-1517 - Cancelling a recursive job can cause tasks to be stuck in PAUSED
HUB2-1518 - Cancelling a job while "recursive_navigate" task is in started, causes objects to be stuck "In-progress"
HUB2-1521 - Error in return status of ngmigrate task
HUB2-1549 - Completed time not set for cancelled job with failed tasks
HUB2-1559 - UI does not allow having empty emails on user update modal

2.1.1: 2024-02-08
=================

Bug

HUB2-1367 - Spaces can become unlinked where multiple "sites" are defined on the same PixStor cluster

2.1.0: 2023-12-22
=================

Feature

Ability to configure Ngenea Backup

Ability to manage NAS users and groups
Ability to view disk metrics
New storage info page
New Alerts UI

2.0.2: 2023-11-17
=================

Feature

New Search page, integrating PixStor Search into Hub
New Policies page, to configure and run data migration policies from the Hub UI

HUB2-27 - Prevent access to UI components according to user permissions
HUB2-660 - Display all GPFS filesystems in Site Dashboard
HUB2-691 - Ability to give space shares custom names
HUB2-692 - Ability to create multiple shares per space at different paths
HUB2-754 - API to fetch settings tasks list for a job
HUB2-791 - Global view of Jobs in the UI
HUB2-792 - Support for submitting custom workflows in the UI
HUB2-823 - Support for setting analytics and metrics URL from UI
HUB2-828 - Ability to specify target path for Space creation
HUB2-831 - Filesystems API endpoint
HUB2-1003 - System folders are hidden in the UI
HUB2-1014 - Ability to bookmark and link locations in the file browser
HUB2-1043 - Ability to view job file lists in job details page rightbar
HUB2-1071 - Added DNS search domains to Site settings
HUB2-1075 - Support for editing job Schedules in the UI

Improvement

HUB2-370 - Improved progress bars proportion in job details page
HUB2-527 - Sensitive settings such as ngenea secret keys are hidden from the UI and API
HUB2-785 - Include share path in file list for share jobs
HUB2-795 - Jobs page view persists when refreshing the page
HUB2-825 - All members of the group can be viewed via the "view all members" button
HUB2-833 - Improved UI for settings Job details
HUB2-834 - Adds user-friendly job descriptions
HUB2-835 - Snapshot time can only be set when the frequency is greater than 1 day
HUB2-861 - Ability to search for a job by ID on jobs page
HUB2-905 - Updated storage target type naming in ngenea wizard
HUB2-915 - "view task" modal is now full width in the jobs page
HUB2-927 - Users list is sorted alphabetically in the groups tab
HUB2-951 - Server address is now optional when creating S3 ngenea targets

HUB2-1025 - Tooltip for long values in the details sidebar
HUB2-1030 - Validation notice for Samba shares in the Space wizard
HUB2-1070 - Don't allow setting server names longer than NETBIOS allowed limit
HUB2-1124 - Updated labels and icons for default workflows

Bug

HUB2-595 - Create space parent directory if it doesn't already exist
HUB2-783 - Set owner for automated settings tasks to 'system'
HUB2-796 - Exclude certain interface types from site NAS settings
HUB2-805 - Filters not working on Users or Groups pages
HUB2-816 - Workflow file list selected whole space instead of individual files
HUB2-832 - Validation for space placement pools
HUB2-854 - Folder metadata not loading in the file browser
HUB2-856 - Fix settings page link for when a custom base URL is configured
HUB2-931 - None values being written to samba shares
HUB2-932 - Multiple delete jobs being triggered for the same share
HUB2-933 - Don't force changing site shortcode on create site wizard
HUB2-935 - Broken spaces page when only one site is configured
HUB2-949 - Go back button for "advanced configuration" breaks UI on ngenea target wizard summary
HUB2-953 - Add api_secure_verify support to ngclient
HUB2-1028 - Site settings weren't updated in the UI after applying a change

2.0.1: 2023-09-19
=================

Bug

HUB2-931: Don't write None values to samba shares

Documentation

HUB2-673: Updated user docs for Ngenea Hub 2.0

2.0.0: 2023-08-11
=================

Feature

Initial Release:

- Hub 2.0 UI
- Provide the ability to view pixstor sites and storage
- Provide the ability to configure pixstor sites from the Hub UI

```
- Provide the ability to create Spaces
- Provide the ability to view files globally across all Space
associated sites
- Provide the ability to perform data workflows
- Provide the ability to monitor jobs and tasks
- Provide the ability to manage Hub users and groups
- Support for Active Directory/LDAP logon
- Worker support of Hub 2.0 functions
  - Analytics data
  - Get and Set pixstor settings for a site
  - Autojoin to Hub
```

License

Ngenea Hub is licensed under the DataCore EULA:

```
DataCoreTM Software Corporation End User License Agreement
1610407359.4 Rev. May 1, 2025

THIS IS A BINDING CONTRACT (“AGREEMENT”) BETWEEN DATACORE SOFTWARE
CORPORATION
(“DATACORE”, “WE”, “US”, “OUR”) AND THE COMPANY OR OTHER LEGAL
ENTITY WISHING TO USE OR
USING OUR SOFTWARE (“YOU”, “YOUR”, “LICENSEE”). THE SECTION BELOW
HEADED “INDIVIDUAL
USERS” IS ALSO BINDING ON EACH INDIVIDUAL USER OF OUR SOFTWARE. BY
CLICKING ACCEPT OR
AGREE, OR BY ACCESSING, OPENING, DOWNLOADING, INSTALLING, COPYING
OR OTHERWISE USING
ANY OF THE SOFTWARE (INCLUDING BY AUTHORIZING, ARRANGING FOR AND/
OR PERMITTING ANY
EMPLOYEE OR OTHER MEMBER OF LICENSEE PERSONNEL TO DO ANY OF THE
SAME), YOU ACCEPT
AND AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT (“AGREEMENT
EFFECTIVE DATE”). IN
THIS AGREEMENT, THE TERM “SOFTWARE” MEANS: (A) THE SOFTWARE IN
RESPECT OF WHICH THIS
AGREEMENT HAS BEEN PROVIDED TO YOU, INCLUDING BUT NOT LIMITED TO,
ALL DATACORE BRANDED
SOFTWARE, PERIFERY BRANDED SOFTWARE,SANSYMPHONY BRANDED SOFTWARE,
PULS8 BRANDED
SOFTWARE, SWARM BRANDED SOFTWARE, FILEFLY BRANDED SOFTWARED, AI+
BRANDED SOFTWARE,
OBJECT MATRIX BRANDED SOFTWARE, PIXIT MEDIA BRANDED SOFTWARE,
ARCASTREAM BRANDED
SOFTWARE, ARCAPIX BRANDED SOFTWARE, OR SOFTWARE THAT MAY BE PRE-
INSTALLED ON
APPLIANCES (“HARDWARE”) UNDER ANY DATACORE, SANSYMPHONY, OBJECT
MATRIX, PERIFERY,
PULS8, PIXIT MEDIA, ARCASTREAM BRAND OR ARCAPIX BRAND; SOFTWARE
UNDER THE CARINGO OR
DATACORE FILEFLY OR SWARM BRANDS; AI+ BRANDED SOFTWARE; (B) ANY
```

AND ALL REMOTELY
ACCESSED CLOUD AND/OR WEB SERVICES PROVIDED OR MADE AVAILABLE TO YOU UNDER THE
DATACORE BRAND AND/OR ANY OF THE SANSYMPHONY, PERIFERY, PULS8, OBJECT MATRIX, PIXIT
MEDIA, ARCASTREAM, ARCAPIX, CARINGO, FILEFLY, SWARM AND AI+ BRANDS; AND (C) ANY AND ALL
OTHER SOFTWARE (INCLUDING ANY APPLICATION PROGRAMMING INTERFACE (API)) AND CLOUD AND
WEB SERVICES PROVIDED OR MADE AVAILABLE TO YOU BY DATACORE. YOU ACKNOWLEDGE AND
AGREE THAT: (A) IF DATACORE PROVIDES YOU WITH ANY OTHER LICENSE IN RESPECT OF SUCH
SOFTWARE IN ADDITION TO THIS AGREEMENT, UNLES SUCH LICENSE EXPRESSLY PROVIDES
OTHERWISE, IT SHALL SUPPLEMENT AND NOT SUPERSEDE OR REPLACE THIS AGREEMENT AND, IN
THE EVENT OF CONFLICT BETWEEN SUCH LICENSE AND THIS AGREEMENT, THE PROVISIONS OF THIS
AGREEMENT SHALL PREVAIL); AND (B) THE SOFTWARE AND ITS ACCOMPANYING DOCUMENTATION
ARE PROVIDED SOLELY UNDER LICENSE AND NOT SOLD TO YOU. YOU DO NOT ACQUIRE ANY
OWNERSHIP INTEREST IN THE SOFTWARE OR DOCUMENTATION UNDER THIS AGREEMENT.

DATACORE IS WILLING TO LICENSE THE SOFTWARE TO YOU ONLY IF YOU ACCEPT ALL OF THE TERMS
IN THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THESE TERMS, YOU MAY NOT ACCESS, OPEN,
DOWNLOAD, INSTALL, COPY OR OTHERWISE USE ANY SOFTWARE, AND YOU MUST PROMPTLY
DESTROY ALL DOWNLOADED SOFTWARE IN YOUR POSSESSION OR CONTROL, INCLUDING ANY BACK-
UP COPY, AND RETURN ALL OTHER SOFTWARE TO THE VENDOR FROM WHOM IT WAS ACQUIRED IN
ACCORDANCE WITH THE VENDOR’S RETURN POLICY FOR THE SOFTWARE.

THE TERMS OF THIS AGREEMENT MAY BE UPDATED FROM TIME TO TIME AT DATACORE’S SOLE
DISCRETION, WITH OR WITHOUT NOTICE TO YOU. IT IS YOUR RESPONSIBILITY TO REVIEW AND
COMPLY WITH THE CURRENT VERSION OF THIS AGREEMENT, WHICH CAN BE FOUND AT:
https://www.datacoreassets.com/resources/legal/DataCore-EULA.pdf.

CERTAIN DATACORE SOFTWARE REQUIRES ACTIVATION. IF THE SOFTWARE REQUIRES ACTIVATION,
YOU WILL BE PROMPTED TO ACTIVATE THE SOFTWARE IN ACCORDANCE WITH THE INSTRUCTIONS
PROVIDED. IF THE SOFTWARE REQUIRES ACTIVATION AND IS NOT ACTIVATED WITHIN THIRTY (30) DAYS
AFTER SOFTWARE INITIALIZATION, IT WILL CEASE FULL OPERATION UNTIL THE SOFTWARE IS

ACTIVATED. SOME EVALUATION VERSIONS OF THE SOFTWARE MAY NOT BE ELIGIBLE FOR ACTIVATION.
NOTICE TO DATACORE HARDWARE USERS: DATACORE HARDWARE MAY CONTAIN, BUT MAY NOT BE
LIMITED TO, SOFTWARE FROM VMWARE, RED HAT, PROXMOX, UBUNTU, ELASTICSEARCH, AND/OR
MICROSOFT (TOGETHER, “OTHER SOFTWARE”). NOTHING IN THIS AGREEMENT WILL BE CONSTRUED
AS GRANTING ANY RIGHTS TO YOU, BY LICENSE OR OTHERWISE, IN OR TO SUCH OTHER SOFTWARE.
YOU ARE RESPONSIBLE FOR OBTAINING THE APPROPRIATE LICENSE RIGHTS FOR ITS USE FROM
THEM DIRECTLY. NONE OF THE TERMS AND CONDITIONS OF THIS AGREEMENT EXTEND TO OR
GOVERN THE OTHER SOFWARE AND DATACORE HAS NO LIABILITY OR OBLIGATION WITH RESPECT
TO ANY SUCH OTHER SOFTWARE.

NOTICE TO EVALUATION USERS: UNTIL YOU HAVE CONVERTED TO A PAID PRODUCTION USE
LICENSE, THIS LICENSE IS SOLELY FOR THE PURPOSE OF YOUR EVALUATION OF THE SOFTWARE.
YOU ARE NOT PERMITTED TO USE THE SOFTWARE IN A PRODUCTION ENVIRONMENT OR TO PROVIDE
SERVICES. IF YOU CHOOSE NOT TO CONVERT TO A PAID PRODUCTION USE LICENSE, THE SOFTWARE
WILL CEASE OPERATING IN THIRTY (30) DAYS, OR SUCH OTHER DATE AS DATACORE MAY SPECIFY
(THE “EVALUATION PERIOD”). FOR THE PURPOSES OF THIS AGREEMENT, “EVALUATION” SHALL MEAN
ANY FREE, TRIAL, PILOT OR DEMONSTRATION USE; EVEN IF EXTENDED BEYOND THIRTY DAYS BY
WRITTEN PERMISSION OF DATACORE FOR SUCH LIMITED USE.

Individual Users. As stated above, in all other parts of this Agreement, references to “You” or “Your” are to
Licensee. In this paragraph only, references to “You” or “Your” are to the specific individual employee or other
authorized user of Licensee using our Software. You hereby warrant and represent that either: (i) Licensee has
previously entered into and is bound by this Agreement and You are authorized by Licensee to use the Software
on its behalf, or, if Licensee is not already bound by this Agreement, (ii) You are authorized to enter into, and have
entered into, this Agreement on behalf of Licensee and bind Licensee to comply with this Agreement and You are
authorized by Licensee to use the Software on its behalf. You further confirm and agree that: You will not use the
Software or act (or omit to act) in any way that would put Licensee in breach of this Agreement [and, if applicable,
that You will use the Software at all times only in accordance with and subject to the Terms of Use that may apply
to the Software.

You acknowledge and agree that the Software may contain a time-sensitive disablement feature (“Time-Bomb”) that will automatically deactivate the Software upon the expiration of any Term License (as defined below) for the Software. You agree not to tamper with, disable, or attempt to bypass the Time-Bomb feature. You acknowledge that the Time-Bomb feature is integral to the enforcement of the applicable term of the license for the Software and agree that DataCore shall not be liable for any loss or damage that may arise from the disablement of the Software as a result of the Time-Bomb feature. Upon the deactivation of the Software by the Time-Bomb feature, You must cease all use of the Software and undertake to delete or destroy all copies of the Software in Your possession or control within a (sixty) 60-day grace period.

Evaluation License Grant. If Your quote, agreement, proposal and/or order form (“Order”) is for an evaluation of the Software, subject to the terms and conditions in this Agreement, and solely during the Evaluation Period, DataCore hereby grants to You a revocable, nontransferable, non-exclusive, non-sublicensable, limited license to download, install and use an on-premises version, or access and use on a third party cloud infrastructure (as specified on the Order), one copy of the binary code version of the Software and its accompanying documentation for Your internal use only and subject to all the following: such use must be (i) solely for purposes of Your internal evaluation and testing of the operation of the Software in a nonproduction environment; (ii) during the term of the Evaluation Period only; and (iii) limited to the parameters and optional features (if any), specified on the Order. Limited parameters may include, but not be limited to, number of computing devices (including virtual machines running in Your own environment or on a third party cloud infrastructure authorized by DataCore), storage capacity, time period, and number of users. You will not use or cause or permit any use of the Software in a production environment. You will adhere to the license restrictions (and all applicable provisions) of this Agreement. You may not publish or disclose to third parties any evaluation of the Software without DataCore’s prior written consent.

Production Use License Grant. If Your Order is for commercial use of the Software on a non-evaluation basis, subject to the terms and conditions in this Agreement, DataCore hereby grants to You a non-transferable, non-exclusive, non-sublicensable, limited to the paid term (as applicable), license to download, install and use an on-premises version, or access and use on a third party cloud infrastructure authorized by DataCore (as specified on the Order), the binary code version of the Software for Your internal use, subject to the parameter limits and optional

features (if any), of Your license as may be stated on Your DataCore Order. License parameter limits may include, but not be limited to, number of computing devices (including virtual machines running in Your own environment or on a third party cloud infrastructure), storage capacity (total, used, and managed; as explained below), number of users, time period, and other limitations specified by DataCore. Storage capacity means Your entire storage capacity accessible to DataCore, less any capacity occupied by data protection algorithms, which may include more than one DataCore software application or instance. For the avoidance of doubt, storage capacity is the sum total of all storage space accessible to DataCore software applications, even if not (yet) used or managed at a given time; used capacity is storage space taken up by data, metadata and system information managed by DataCore and managed capacity is storage space under the control of a DataCore software application and available for consumption even though not in use. Storage capacity will be used and monitored to determine storage-based license fees. If You want to increase Your license parameters and/or features beyond that for which You have paid, You must purchase and pay DataCore for such increased parameters and/or features at DataCore's then applicable rates. Unless otherwise specified on Your Order, such rates will be the most recent rates published by DataCore at the relevant time. DataCore may share Your data and information with third party service providers, which may include but not be limited to, OpenAI and Amazon, in order to provide the Software. In some cases, at DataCore's sole discretion, a Letter of Credit or other financial arrangement may be required by DataCore before accepting Your Order.

DCSPP Participants. Pursuant to Your valid DataCore Cloud Service Provider Program ("DCSPP") Agreement, You may only use the Software to provide Hosted IT Services. "Hosted IT Services" means an internet-based subscription service operated by a service provider entity that consists of providing multiple end service users (as defined below) access to: (i) the storage resources of systems operated by the service provider entity (such as utility or grid computing), or (ii) various software applications that are installed and operated on the systems of the service provider entity. For purposes of this Agreement, "end service users" may be: (i) independent third-parties with which the service provider has a commercial relationship; or (ii) departments, divisions or workgroups served by a subscribing central hosting service provider. Your authorized users, (employees or others), may access the Software remotely through a wide area network or VPN, or other secure remote access method, provided that You comply with the parameter limitations of the Software and subscription plan for which You have been expressly authorized

by DataCore. All DCSPP participants (authorized Cloud Service Providers (“CSPs”)), must have accepted and be subject to a current and valid DataCore Cloud Service Provider Agreement, as well as a current and valid CSP/Aggregator Agreement that includes, but is not limited to: (i) complete terms of use for the DCSPP, (ii) Software usage requirements and limitations, and (iii) DataCore’s liability limits, in accordance with this Agreement. For the purposes of this Agreement, a DataCore Aggregator (“Aggregator”) shall mean a Software Distributor that has a current and valid contract with DataCore, which authorizes them to sell and distribute the Software for the DCSPP.
If applicable, You may make one backup copy of the Software, provided the copy must contain all of the original Software’s copyright, trademark and other proprietary notices.

Object Matrix Users. Object Matrix users shall comply with this Agreement along with any additional terms and conditions provided to You by DataCore. Object Matrix perpetual software licenses are not transferrable between devices. For the avoidance of doubt, Object Matrix perpetual software licenses are tied solely to the device(s) upon which they are initially installed. For Legacy (as defined below) Object Matrix users, specific terms and conditions of Your use of Object Matrix products and services (“OM Terms”) may be found at: https://www.datacore.com/legal/, under the Legacy Object Matrix section. “Legacy” means Object Matrix users that meet all conditions stated at the Legacy Object Matrix section. Rights granted to You apply only to the specific products to which You have subscribed as set out in Your Object Matrix Order (as the case may be) (the “Order”) relate. For Legacy Object Matrix users, in the event of a conflict arising between this Agreement and OM Terms, then the OM Terms shall prevail.
Pixit Media and ArcaStream Users. Pixit Media, ArcaStream and Arcapix users (“PMA Users”) shall comply with this Agreement along with any additional terms and conditions provided to You by DataCore. Certain license parameters and reissuance procedures may apply to Pixit Media licenses. For Legacy PMA Users (as defined below), specific terms and conditions of Your use of Pixit Media and/or ArcaStream products and services (“Legacy PMA Terms”) may be found at: https://www.datacore.com/legal/, under the Legacy Pixit Media and ArcaStream section.
“Legacy PMA Users” means Pixit Media and/or ArcaStream users that meet all conditions stated at the Legacy Pixit Media and ArcaStream section. Rights granted to You apply only to the specific products to which You have subscribed as set out in Your associated Order (as the case may be) (the “PMA Order”) relate. For Legacy PMA Users, in the event of a conflict arising between this Agreement and Legacy PMA Terms, then the Legacy PMA Terms shall prevail.

License Restrictions. Your employees may access the Software remotely through a wide area network or VPN, or other secure remote access method, provided that, if applicable, You may in no event exceed the number of permitted concurrent uses or users of the Software for which You have been expressly authorized by DataCore. In the absence of any express written authorization by DataCore to the contrary, the number of concurrent uses or users of the Software, the number of computing devices (including virtual machines) on which the Software may be used, and the number of copies of the Software You may make, shall be one. You may also make one back-up copy of the Software, provided the copy must contain all of the original Software's copyright, trademark and other proprietary notices. Any copy remains the exclusive property of DataCore or its licensors. You will not cause or permit: (i) use or copying of the Software or documentation, except as expressly provided in this Agreement; (ii) modification, adaption, assignment, distribution, rental, sub-license, lease, lending or transfer of the Software; (iii) reverse engineering, disassembly, translation, decoding or decompilation of the Software, except to the extent expressly permitted by law notwithstanding this prohibition; (iv) creation of any derivative works based on the Software; (v) removal, deletion, circumvention or alteration of any trademarks, copyright or other intellectual property rights notices provided with the Software; (vi) removal, disablement or circumvention of any security or copy protection features; (vii) use of the Software to violate or circumvent any law, regulation or rule, or for any purpose other than its intended use in accordance with the documentation; (viii) except as DataCore may otherwise agree in writing, use of the Software in connection with a service bureau or other use or configuration whereby the Software is used by, for the benefit of, or to provide a service on the computer equipment of, a third party. Notwithstanding the foregoing, if You have express written authorization from DataCore that grants You leasing, factoring, or other alternative servicing rights to the Software, such rights may be exercised subject to full compliance with such Agreement. The Software requires integration with an Amazon Web Services (AWS) account and an OpenAI account. You are responsible for obtaining and maintaining the AWS and OpenAI accounts that are integrated with the Software and complying with any terms and conditions of Your AWS and OpenAI accounts. Failure to do so will not relieve Your obligations under this Agreement, including Your obligation to make payments for the Software that are due to DataCore.

Ownership. The Software (including any copies) is licensed, not sold. DataCore and its licensors retain all right, title and interest in the Software and documentation, and in all

copies, improvements, updates, revisions, enhancements, modifications and derivative works thereof, including, without limitation, all patent, copyright, trade secret, trademark and database rights. The terms of this Agreement are intended to benefit any third party licensors, who may directly enforce applicable terms of this Agreement to protect their interest in any of the Software. DataCore and its licensors reserve all rights not expressly granted to You herein, and no other licenses, whether express, implied or otherwise, are granted to You.

Maintenance, Support and Updates. DataCore has no obligation to provide any maintenance, support, updates, or fixes for any Software that You use during an Evaluation Period. For Software licensed to You on a term basis or as a subscription service, DataCore, from time to time, may update, fix, and maintain the Software as DataCore, in its sole discretion, deems necessary, or as DataCore may otherwise expressly agree in writing. For perpetual Software licenses, except as provided under the limited warranty set forth below, and except as DataCore may otherwise expressly agree in writing, DataCore is under no obligation to maintain, support or update the Software in any way, or to provide updates or error corrections, however, such services may be separately purchased under a support contract. To receive technical support entitlements for Software, all licenses currently activated for the Software residing on or utilized by Your system must be under a current support contract with DataCore (either as included in the purchase of a Term License or as a separately purchased support contract for a perpetual license). Systems running any active licenses that are not covered by a current support contract shall be deemed ineligible for technical support. Support contracts are tied to specific licenses and are not interchangeable with other licenses. For all licenses (term, subscription or perpetual), if DataCore provides You with a bug fix, maintenance release or update to the Software, it is provided to You as is and shall be considered Software subject to the terms of this Agreement, unless You receive a separate license from DataCore for that release or update that expressly supersedes and replaces this Agreement. In addition, the Software may contain support and/or data collection functions that provide DataCore with certain system telemetry information that DataCore may use to provide support, analysis, and reporting to You, as well as to update or enhance the Software. By using the Software, You consent to DataCore’s access and use of such information for such purposes. Subject to satisfaction of any eligibility conditions stated therein, the terms of the DataCore Support Services Terms and Conditions, as available through the following link: https://www.datacoreassets.com/resources/support/DataCore-Software-Support-Services-Terms-and-Conditions.pdf, shall apply and hereby be incorporated

into this Agreement by this reference. Certain support and maintenance terms applicable to Hardware, may be found under the Legacy Object Matrix section at: https://www.datacore.com/legal/.

System Data Rights. You agree that DataCore will collect and track technical and related information about You and Your use of the Software, which may include, but not be limited to, Your internet protocol address, hardware identifying information, operating system, application software, storage capacity, peripheral hardware, and Software usage statistics (together “System Data”), to assist with the operation and function of the Software, the provision of updates, support, invoicing, marketing by DataCore or its agents, benchmarking and research and development. A current summary of information collected by DataCore (the “DataCore Collection Manifest”) is available through the following link:https://docs.datacore.com/intelligenceservice/intelligenceservice/collection-manifest.htm, and hereby incorporated into this Agreement by this reference. The terms of the DataCore Data Processing Agreement (“DPA”) as available through the following link: https://www.datacoreassets.com/resources/legal/DataCore-Data-Protection-Provisions-Addendum.pdf shall also apply and hereby be incorporated into this Agreement by this reference.

Automated Data Collection. You agree that DataCore may utilize automated data collection tools (“ADC”) for automated collection, processing and monitoring of Your Software usage data for DataCore’s reporting, billing and associated purposes. Furthermore, You will cooperate with DataCore to enable ADC, as necessary.

(Examples of ADC may include, but not be limited to, DataCore Insight Services (“DIS”), Phone Home or any other telemetry service tool utilized by DataCore). In the event You disable, turn off, circumvent or otherwise prevent ADC from functioning correctly during the term of this EULA, You acknowledge and agree that You may be billed (and will be required to pay) at a usage rate of two times (“2x”) the peak monthly amount billed in any prior month in which ADC was functioning.

Confidentiality. The Software contains confidential and proprietary information of DataCore and/or its licensors (“Confidential Information”). You agree not to use Confidential Information except as necessary to perform this Agreement and shall not disclose Confidential Information to any third party. You agree to take all reasonable and adequate measures, and no less than measures You take to secure Your own confidential and proprietary information, to protect and secure the Confidential Information

and Software from unauthorized disclosure or use.
Payment. If a non-recurring installation, setup or onboarding charge (“Installation Charge”) is specified on an Order, DataCore will invoice You for such Installation Charge upon the effective date of the Order. If a recurring charge (“Recurring Charge”) (e.g., Monthly Charge, Quarterly Charge, Annual Charge, etc.) is specified on an Order, DataCore will invoice You for the Recurring Charge in advance for each period. If applicable, DataCore will invoice You and You will pay such invoices for any additional charges for products and services described on an Order. You will pay all invoices within thirty (30) days from the date of such invoice.
All invoices must be paid in accordance with their terms without setoff or deduction, and late payments will accrue interest on the unpaid sum as of the date of the invoice at the lesser of (i) the highest legal rate of interest permitted by applicable law or (ii) one and one-half percent (1.5%) per month. DataCore may apply any payments received by DataCore to any one of Your then outstanding charges.

Taxes. All amounts payable by You to DataCore hereunder are exclusive of any sales, use and other taxes or duties (collectively, “Taxes”). You are solely responsible for payment of any Taxes, except for those taxes based on the income of DataCore. You will not withhold any Taxes from any amounts due to DataCore.
Limited Warranty. During the Evaluation Period the Software is provided and You accept the Software “AS-IS” and “WITH ALL FAULTS.” Upon commencement of the production use license, DataCore warrants for a period of ninety (90) days thereafter that the Software will substantially conform under normal use to DataCore’s specifications contained in the user guides and operating manuals provided by DataCore with or in respect of the Software. DataCore will, at its sole discretion, either promptly replace any Software that fails to comply with this warranty at its cost or refund the amount paid for the Software and attributable to the remainder of the production license period. Any claims submitted under this section must be submitted in writing to DataCore within the specified warranty period. This limited warranty is void if failure of the Software results from accident, abuse, misapplication, abnormal use or a virus. Any replacement for the Software, and any bug fix, maintenance release or update to the Software, will be warranted under this limited warranty for the remainder of the original warranty period applicable to the Software or thirty (30) days from its delivery, whichever is longer. THIS SECTION STATES YOUR SOLE AND EXCLUSIVE REMEDY, AND DATACORE’S AND ITS SUPPLIERS’ SOLE AND EXCLUSIVE LIABILITY, IN CONNECTION WITH THE SOFTWARE, INCLUDING FOR ANY BREACH OF THE WARRANTY RELATING TO THE SOFTWARE. THIS LIMITED WARRANTY SET FORTH IN THIS

SECTION GIVES YOU
SPECIFIC LEGAL RIGHTS. YOU MAY HAVE ADDITIONAL RIGHTS, WHICH VARY
FROM JURISDICTION TO
JURISDICTION.

THE FOREGOING WARRANTY IS EXCLUSIVE AND IN LIEU OF ALL OTHER
WARRANTIES AND
CONDITIONS, WHETHER EXPRESS OR IMPLIED, AND DATACORE AND ITS
SUPPLIERS EXPRESSLY
DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, INCLUDING ANY
IMPLIED WARRANTIES OF
MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT, FITNESS
FOR A PARTICULAR
PURPOSE, AND AGAINST HIDDEN DEFECTS TO THE FULLEST EXTENT
PERMITTED BY LAW. NO ADVICE
OR INFORMATION, WHETHER ORAL OR WRITTEN, OBTAINED FROM DATACORE OR
ELSEWHERE WILL
CREATE ANY WARRANTY OR CONDITION NOT EXPRESSLY STATED IN THIS
AGREEMENT. DATACORE
DOES NOT WARRANT THAT THE SOFTWARE WILL MEET YOUR REQUIREMENTS OR
THAT USE OF THE
SOFTWARE WILL BE UNINTERRUPTED, ERROR FREE, OR FREE OF VARIATIONS
FROM THE
DOCUMENTATION. DATACORE IS NOT RESPONSIBLE FOR ANY INTERFERENCE
WITH OR INABILITY TO
USE THE SOFTWARE RESULTING FROM ADDITIONAL SOFTWARE OR SERVICES
PROVIDED BY THE
CLOUD INFRASTRUCTURE SUPPLIER, IF ANY, THROUGH WHICH YOU ACCESS
THE SOFTWARE.

Evaluation License Termination. The Evaluation license granted
under this Agreement shall terminate automatically
upon the expiration of the Evaluation Period. In addition, the
Evaluation license shall terminate prior to the expiration
of the Evaluation Period upon the occurrence of any of the
following: (i) either You or DataCore at any time give the
other written notice of termination (with or without cause); (ii)
You breach any provision in this Agreement; or (iii) if
applicable, Your access to the Software is terminated by Your
cloud infrastructure provider. The Software will cease
to function or become inaccessible in whole or in part upon
termination of the evaluation license.
Production Use License Termination. The production use license
granted to You under this Agreement will terminate
immediately and automatically without notice upon the occurrence
of any of the following: (i) if You have licensed the
Software on a term basis (a “Term License”) or on a subscription
basis (“Subscription Term”), upon the expiration of
Your pre-paid Term License or Subscription Term; (ii) You breach
any provision in this Agreement; (iii) DataCore does
not receive payment in full for Your license; (iv) if You entered
into an Exchange Agreement with DataCore in which
Your perpetual license(s) are terminated in exchange for Term
License(s), Subscription Term, a different edition

of perpetual licenses, or license key exchange for equivalent license or capacity (TB); or (vi) if applicable, Your access to the Software is terminated by the cloud infrastructure provider through which You acquired the production use license. Upon termination of the Term License or Subscription Term, as applicable, You will discontinue use of the Software and remove any and all copies of the Software and any part of the Software from any and all computing devices, including any back-up copy, and destroy the Software. At DataCore's request, You will certify in writing to the foregoing.

With regard to Term Licenses and Subscription Terms, at the termination or expiration date of Your Term License or Subscription Term, as applicable, and without notice from DataCore, You will be provided a (sixty) 60-day grace period to collect Your exportable data. In the event Your Term License or Subscription Term is renewed within the (sixty) 60-day grace period (requiring the payment of all fees due), Your use of the Software and access to Your data may be resumed, provided that You have properly backed-up Your configuration information and it can be restored. If DataCore does not receive and acknowledge Your Term License or Subscription Term renewal within the (sixty) 60-day grace period, DataCore may delete or destroy Your configuration and historic information associated with the Software. After the (sixty) 60-day grace period, Your previous Term License or Subscription Term may (at DataCore's sole discretion; conditions and fees may apply) no longer be renewed and any Term License or Subscription Term added may be treated as a new Term License or Subscription Term, as applicable, and unrelated to the terminated Term License or Subscription Term. DataCore has no obligation to restore access to any previous Term License or Subscription Term configuration information, history or data. In no event shall DataCore be liable to You for loss or accuracy of Your configuration information or data. The provisions of this Agreement, except for the license grant and warranty, will survive termination.
Term. This Agreement comes into force on the Agreement Effective Date and shall continue in effect until terminated in accordance with its terms.

Early Termination. Except for early termination due to a material breach of this Agreement by DataCore, You may not terminate Your Term License, Subscription Term or Support Agreement without prior payment of all fees due for the entire period for which Your Order applies. For multi-year license or support terms, all future payments must be received by DataCore prior to DataCore's acceptance of early termination, which acceptance shall be made at DataCore's sole discretion. In the event of early termination due to a material breach of this Agreement by

DataCore, you must first pay all fees due and unpaid in respect of the entire period for which Your Order applies up to the date of such breach. In no event shall any of Your prepaid fees be returned to You by DataCore for early termination.

Effect of Termination. Termination or expiration of this Agreement shall not: (i) affect any of the rights, obligations or liabilities accruing to You or DataCore prior to the date of termination or expiration, or (ii) release You from any obligations, including the payment of all fees due, or liabilities you may have under any other agreement with any DataCore distributor, reseller or aggregator. Notwithstanding termination or expiration of this Agreement, all provisions of this Agreement which are expressly stated to survive termination or expiration of this Agreement or which by their nature are intended to survive termination or expiration shall survive and remain in effect in accordance with their terms.

Audit Rights. During the term of this Agreement and for a period of two (2) years from the date of its termination[,cancellation] or expiration: (i) You shall maintain complete, clear, and accurate records that demonstrate compliance with the terms and conditions of this Agreement and any other agreement(s) in force between You and DataCore, and (ii) upon thirty (30) days written notice, DataCore (or DataCore’s representative) will be entitled to audit Your books, records and use of the Software and any other DataCore materials provided to You, to verify compliance with the terms of this Agreement and any other agreement(s) in force between You and DataCore, the functioning of ADC, and all applicable DataCore policies. You shall promptly pay to DataCore or the relevant DataCore distributor, reseller or aggregator, as may be applicable, any underpayments owed to such party as revealed by any such audit, plus any applicable late payment fees. Any such audit will be performed at DataCore’s expense during normal business hours, in a manner so as to try not to upset Your normal business operations and procedures, provided that You shall cooperate fully with DataCore and shall promptly reimburse DataCore in full for the cost of such audit if such audit reveals: (a) an underpayment by You of more than five percent (5%) of the amounts payable by You in respect of the period audited (b) any evidence that You have violated any laws, DCSPP rules (if applicable to You) or DataCore policies, (c) You have disabled, turned off, circumvented or otherwise prevented ADC from functioning correctly, or (d) any other material breach of this Agreement.

Indemnification. You agree to indemnify and hold DataCore, its suppliers, subcontractors and other partners, and

their respective officers, agents, partners and employees, harmless from any fee, loss, awards, damages, liability, claim, or demand, including, but not limited to, reasonable attorneys' fees, made by any third-party due to or arising out of or in connection with Your use of the Software, Your violation of any proprietary rights of another by Your data or information provided to the Software.
Limited Liability. UNDER NO CIRCUMSTANCES WILL DATACORE OR ITS SUPPLIERS BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS, LOSS OF SAVINGS OR ANTICIPATED SAVINGS, LOSS OF BUSINESS, BUSINESS OPPORTUNITY, BUSINESS INTERUPTION, GOODWILL, REPUTATION OR DATA, COST OF COVER, RELIANCE DAMAGES OR ANY OTHER SIMILAR DAMAGES OR LOSS, EVEN IF DATACORE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND REGARDLESS OF WHETHER ARISING UNDER CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE. EXCEPT AS LIMITED BY APPLICABLE LAW, DATACORE'S AND ITS SUPPLIERS' TOTAL LIABILITY UNDER THIS AGREEMENT OR OTHERWISE SHALL IN NO EVENT EXCEED THE GREATER OF $500 OR THE LICENSE FEE PAID BY YOU FOR THE SOFTWARE IN THE 12 MONTHS PRECEDING THE EVENT GIVING RISE TO THE LIABILITY. THE LIABILITY LIMITATIONS SET FORTH IN THIS AGREEMENT SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED IN THIS AGREEMENT OR THE INVALIDITY OF ANY OTHER PROVISION.

U.S. Government Rights. The Software and its documentation are "commercial computer software" and "commercial computer software documentation," respectively as such terms are used in FAR 12.212 and other relevant government procurement regulations. Any use, duplication, or disclosure of the Software or its documentation by or on behalf of the U.S. government is subject to restrictions as set forth in this Agreement.
Export Law Assurances. The Software, including its documentation and related technical data, is subject to the export control laws and regulations of the United States ("Export Laws"). You agree not to export or re-export (directly or indirectly) the Software (including its documentation and related technical data) or any direct product thereof without fully complying with the Export Laws.
License Compliance Assurances. DataCore reserves the right to run periodic license compliance tests to determine and enforce the parameters of Your license(s). Without interruption to users, license compliance data may be reviewed and/or collected on demand, whether by automatic or manual means. Parameters associated

with such compliance tests may include, but not be limited to, system memory, number of CPU cores, storage capacity, users, time, and configuration.

Governing Law. This Agreement will be governed by and construed in accordance with the laws of the State of Florida U.S.A., excluding the United Nations Convention on Contracts for the International Sale of Goods, and without regard to principles of conflicts of law. Each party consents to the jurisdiction and exclusive venue of the state and federal courts of Broward County, Florida U.S.A. provided that DataCore shall at all times have the right to commence proceedings in any other court of its choice of appropriate jurisdiction to obtain an injunction, specific performance or other equitable relief for protection of intellectual property rights.
Transfers/Assignment. This Agreement will bind and inure to the benefit of each party's permitted successors and assigns. You may not transfer, novate, assign or otherwise part with (each, a "transfer" of) this Agreement or any of Your rights or obligations under it, in whole or in part, without DataCore's prior written consent. You agree to notify DataCore of any transfer in writing and provide all proposed transferee information requested by DataCore. Consent to any requested transfer shall not be guaranteed and shall be made at DataCore's sole discretion. Any attempt to assign this Agreement without such notice and consent will be null and void. You agree that DataCore may transfer this Agreement or any of its rights or obligations under it to a third party without the requirement for further consent from you in the event of the sale of all or substantially all of the business of DataCore, in the context of a solvent reorganization or any other analogous procedure or otherwise on (thirty) 30 days prior written notice to you.

Third-Party Code. The Software may contain or be provided with third-party components subject to terms and conditions of such third-party licenses ("Third-Party Software"). Third-Party Software may include Open Source Software, as defined below. Third Party Software shall be deemed to be incorporated within the Software for the purposes of this Agreement (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 Agreement in relation to such Third Party Software. You shall indemnify and hold DataCore harmless against any loss or damage which DataCore may suffer or incur as a result of Your breach of any Third Party Additional Terms howsoever arising, and DataCore may treat Your breach of any Third Party Additional Terms as a material

breach of this Agreement. Third Party Software is provided “AS-IS” and the performance of and any issues caused by or arising from any Third Party Software shall be considered an event outside of DataCore control. The Software may contain or be provided with components subject to the terms and conditions of “open source” software licenses (“Open Source Software”). Open Source Software may be identified in the user guides and operating manuals provided by DataCore with the Software, or DataCore may provide a list of the Open Source Software for a particular version of the Software upon written request. To the extent required by the license that accompanies the Open Source Software, the terms of such license will apply in lieu of the terms of this Agreement with respect to such Open Source Software, including, without limitation, any provisions governing access to source code, enhancement, modification or reverse engineering. Conflicts. In the event of a conflict or inconsistency arising between the subject matter of this Agreement and the DPA then the terms of the DPA shall prevail.

General Provisions. If any provision hereof shall be held illegal, invalid, or unenforceable, in whole or in part, such provision shall be modified to the minimum extent necessary to make it legal, valid and enforceable, and the legality, validity and enforceability of all other provisions of this Agreement shall not be affected thereby. No delay or failure by either party to exercise or enforce at any time any right or provision hereof shall be considered a waiver thereof or of such party’s right thereafter to exercise or enforce each and every right and provision of this Agreement. You shall immediately notify DataCore if You become aware of any misuse of the Software or any infringement of DataCore’s intellectual property rights in the Software and fully cooperate with DataCore in any legal action taken to enforce DataCore’s intellectual property rights. This Agreement, including any Order accepted by DataCore, is the complete and exclusive statement between You and DataCore relating to the subject matter hereof and supersedes all prior oral and written and all contemporaneous oral negotiations, commitments, and understandings of the parties, if any, including but not limited to any prior license for the Software. In the case of any conflict between the terms of this Agreement and the provisions of any Order for the Software, the terms of this Agreement shall control. The parties confirm that it is their wish that this Agreement, as well as all other documents relating hereto, have been and shall be drawn up in the English language only. The English language version of this Agreement will control in all respects, and all other versions are for convenience only and are not binding.

Contact

Ngenea Hub is provided and supported by:

- Pixit Media Contact