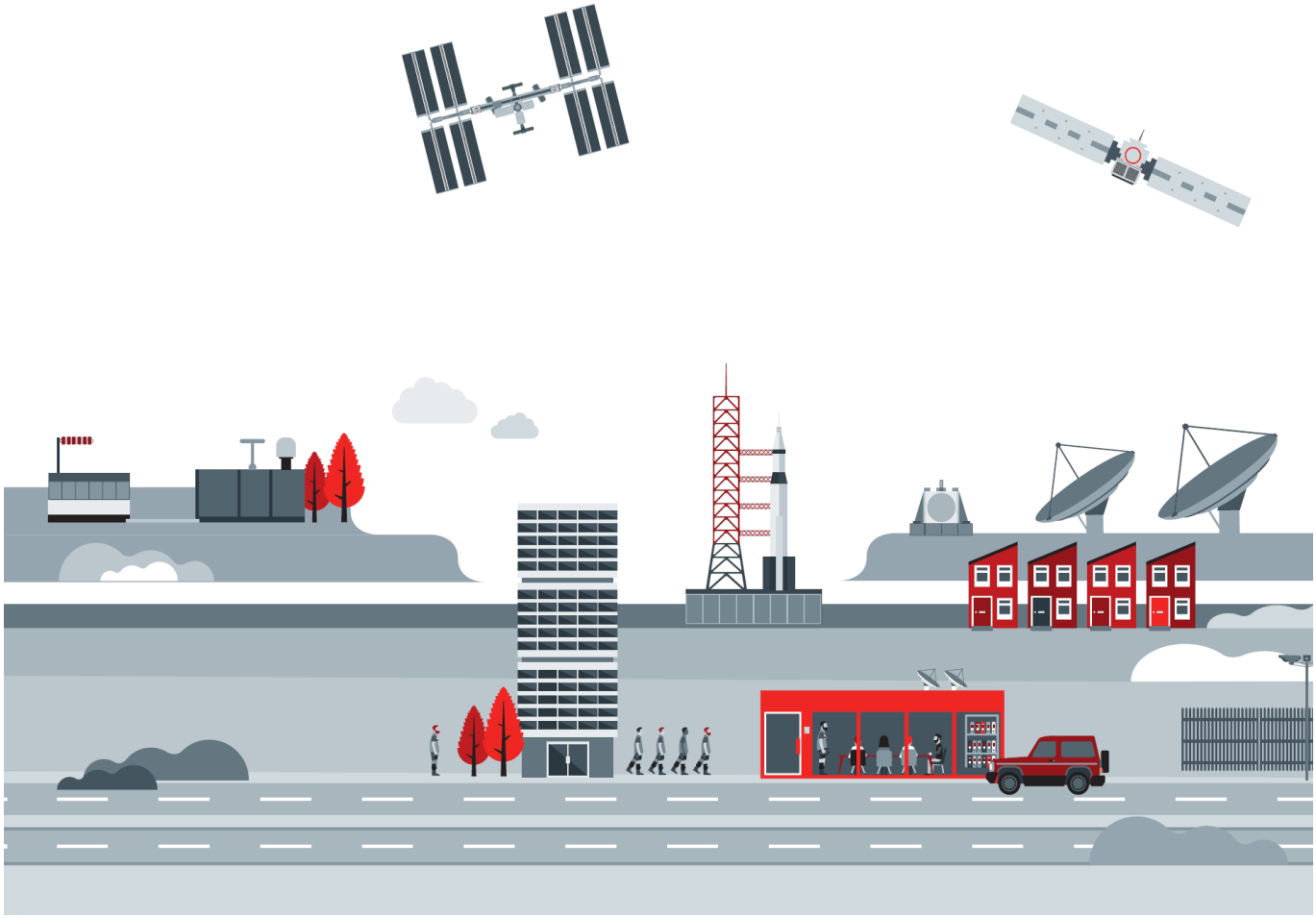
GAEL SYSTEMS
Earth Observation Data Solutions

Serco Business

serco

Collaborative Data Hub Software - Maintenance and Evolution Services - Ready for Digital Twin Earth

GSS Administration Manual



Role/Title	Name	Signature	Date
Author	GAEL Team		01/04/2026
Verified	Iryna Miatka and Giulia Carosi		
Approved	Federico Fioretti		

Change Register

Version/Rev.	Date	Change	Reason
1.0	16/12/2022		First release of the document
1.1	12/01/2023	• Add Keycloak configuration information • Add annexes for Object storage configuration	Update
1.2	02/02/2023	• Fix broken links for distribution • Update annexes to include operational scenarios • Applicable Documents updated • Reference Documents updated • Delete section about Keycloak configuration • Update installation pre-requisites for each component • Add information for Datastore options • Add a note about the filters and pattern for ingesters	Update according to feedbacks
1.3	06/04/2023	• Add docker compose part	Update according to feedbacks
1.4	12/04/2023		Update according to feedbacks
1.5.0	21/08/2023	• Add Datastore request and Ingester requin Admin API • Add database tables description • Correct query by attributes on Catalogue	
1.5.1	08/09/2023	Add information about reprocess for Ingesters	
1.5.2	11/10/2023	• Update explanations about ingester's properties • Add annexes for operational scenario	
1.5.3	13/10/2023	• Docker command	Update according to feedbacks
1.6.0	22/01/2024	• Deletion product feature	
1.6.1	29/01/2024		Update according to feedbacks
1.6.2	12/02/2024	• Configuration changes	
1.6.3	22/04/2024	• cURL commands – Catalogue & Admin-api • Tables and fonts reformat • XML and Json requests reformat • Annexes structure • Annexes reformat • Keycloak configuration – CDH-Admin-API • Keycloak configuration – CDH-Catalogue	Update according to feedbacks
1.6.4	21/05/2024		Update according to feedbacks
1.6.5	24/05/2024	• Included OData examples – Catalogue	

		• Capitailized letters of headings • Reformat	
1.6.6	20/06/2024	• Added quota management default values	Update according to feedbacks
2.0.0	11/09/2024	• STAC API • Eviction management in Admin API • Kafka logs aggregator • Read-only role for Admin API • Ingesters management in Admin API • Auto-restart of Ingesters in Admin API	
2.0.1	25/09/2024	• Add a comment for Nodes navigation • CDH-Catalogue, admin-api and stac-api configuration files adapted	§ 5.9
2.1.0	16/12/2024	• Detail toolbox execution in cahu-compose • Kafka topic creation/alteration (useful for logs aggegation in Kafka) • Add: springdoc.swagger-ui.csrf.enabled=true parameter in admin-api configuration • Add ingester.events.queue.size for admin api configuration	
2.2.1	14/01/2025	• Removal of Annexes structure • Reformat of Annexes • Rename chapter 5.6.1.2 from admin API configuration to catalogue configuration	Updates according to feedbacks
2.1.2	05/02/2025	• Insertion of S3 object-storage configuration for cdh-catalogue and cdh-ingest	Updates according to feedbacks
2.2.0	27/03/2025	Add S3 paths in STAC item Add STAC ingesters Add notifications for ingested and evicted products Update DeletionJob object Add S3 configuration in admin api and xml	DHS#8 features
2.2.1	02/06/2025	Add stac.missions.check.ql property Add stac ingesters configuration examples Add subscription creation examples Remove configuration examples	Updates after DHS#8 validation
2.2.2	31/07/2025	Add configurable S3 properties for STAC API	Updates according to feedback
3.0.0	02/09/2025	Kafka error manager in consumer Gap recovery scripts Custom default quotas Customizable JVM args	3.0.0 release
3.1.0	06/10/2025	S1 Zarr product ingestion Rework gap recovery scripts Add ingestion from PRIP	3.1.0 release
3.2.0	20/11/2025	S2 and S3 Zarr product ingestion	3.2.0 release
3.3.0	26/01/2026	Stac filter extension Change default ingesters configuration in compose module	3.3.0 release

3.4.0	01/04/2026	Optional Kafka authentication	3.4.0 release

Table of Contents

Index of Figures

1. Introduction

1.1 Scope

This document applies to the GAEL Store Service and is maintained within the service "*Collaborative Data Hub Software Maintenance and Evolution Services for Digital Twin Earth*" hereinafter called "the Collaborative service".

1.2 Purpose

The purpose of this document is to describe all essential information to make full use of the GSS software. The target audience of this document therefore are System administrators that will install the GSS software and manage it.

This manual includes a description of the functions implemented by GSS and alternate modes of operation, and step-by-step procedures for system access and use.

It describes (and dedicates a specific section to):

- How to install the GSS software
- How to administer, manage and operate the installed GSS

1.3 Document applicability

Here below we provide a cross check between the version of this document, and the relevant GSS milestone.

GSS Administration Manual Version	GSS Version
V3.4.0	3.4.0

1.4 Applicable Documents

Ref.	Title	Reference and Version
AD-1	[AD-SOW] Statement of Work: COLLABORATIVE DATA HUB SOFTWARE - MAINTENANCE AND EVOLUTION SERVICES - READY FOR DIGITAL TWIN EARTH	ESA-EOPG-EOPGC-SOW-12, v1.0
AD-2	DHS System Design Document	COPE-SERCO-TN-21-1171
AD-3	Open Data Protocol	https://www.odata.org/
AD-4	Data Distribution Interface Control Document.	ESA-EOPG-EOPGC-IF-4

1.5 Reference Documents

Ref.	Title	Reference and Version
RD-1	Collaborative Data Hub Software GSS Software Design Document	GAEL_P311 – GSS-CDH-SDD, v3.4.0
RD-2	Operational Concept Document (OCD)	COPE-SERCO-TN-21-1174, v6.0
RD-3	COPE-SERCO-TN-23-1461 - GSS COTS	COPE-SERCO-TN-23-1461 - GSS COTS Installation, v1.4
RD-4	COPE-SERCO-TN-21-1229 - Keycloak Installation and Configuration Manual	COPE-SERCO-TN-21-1229 - Keycloak Installation and Configuration Manual, v2.5
RD-5	Collaborative Data Hub Software GSS STAC Catalog Access ICD	GAEL-P311-GSS-CDH-STAC-ICD, v2.2

1.6 Acronyms and Abbreviations

Acronym	Definition
AD	Applicable Document
API	Application Programming Interface
CDSE	Copernicus Data Space Ecosystem
DHS	Data Hub Software
DHuS	Data Hub Service
EO	Earth Observation
ESA	European Space Agency
GS	Ground Segment
GSS	Gael Store Service
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICD	Interface Control Document
OData	Open Data Protocol
RD	Reference Document
SOW	Statement of Work
UUID	Universally unique identifier

2. GSS System Overview

The GSS architecture and design are fully described in the GSS Design Document [RD-1]
The following **minimum** specification is recommended/suggested for virtual machine running different GSS components.

Description	Producer/Consumer/Catalogue	Admin API
RAM	16 GB	8 GB
vPCU	8	4
SDD	200 GB	50 GB
OS	Debian 11	Debian 11

In a basic Scenario, the system needs:

- One Producer and one Consumer for Ingestion
- One Catalogue
- One Admin API
- One Notification Consumer
- One STAC catalogue

3. Other tools for GSS

We need other tools to install and manage GSS:

- Docker: 20.10.12 or later

4. Toolbox

The toolbox is a set of scripts to initialize the database and Solr. It includes database/solr maintenance and update scripts.

4.1 Installation pre-requisites

4.1.1 Infrastructure Requirements

We recommend installing:

- Docker: 20.10.12 or later
- A tool to unzip files

4.1.2 Network Requirements

Application	Default port
Solr	8983
PostgreSQL	5432

4.1.3 Software Requirements

The requirements are:

- A running SOLR 9.0.0 or later instance
- A running PostgreSQL instance (10.12 and after: https://www.postgresql.org/download/) with a database created for the software.
- Install Java 17. Make sure java 17 is installed before launching the script. Refer to [RD-3] for installation.
- JTS (Java Topology Suite). Before creating the schema, the Java Topology Suite (JTS) jar must be put in the SOLR webapp dependencies. Refer to [RD-3] for installation.
- JQ package. The package *jq* is also recommend. Refer to [RD-3] for installation.

4.2 Distribution links

This software is available in a zip distribution available at https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-toolbox/3.4.0/cdh-toolbox-3.4.0.zip

The zip archive structured is:

```
cdh-toolbox
├── db
│   └── database_updater.sh
├── etc
│   └── log4j2.xml
├── lib
├── logs
├── gap-recovery
│   └── ...
└── solr
    ├── solr_configuration.sh
    ├── solr_create_schema.sh
```

```
    ├── solr_delete_all_field.sh
    └── solr_init.sh
```

As Administrator, access via SSH to the VM where the CDH-Toolbox should be installed.
Log is as root user and download the CDH-Toolbox using the following command:

```
wget https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-toolbox/3.4.0/cdh-toolbox-3.4.0.zip
```

Unzip the file with the command:

```
unzip cdh-toolbox-3.4.0.zip
```

4.2.1 Database Updater

Enter in the "cdh-toolbox-3.4.0" folder, open the "db" folder.
The script *database_updater.sh* can be used to create and update the PostgreSQL database schema. This database is used internally by the **cdh-ingest** , **cdh-catalogue**, **cdh-admin-api** and **cdh-stac-api.**

The script must be launched from the db/ directory.
Usage:

```
./database_updater.sh --url
jdbc:postgresql://<database_server_url>:<database_port>/<database_name> --login *** --
password ***
```

4.2.2 Solr Schema Creator

Enter in the *"cdh-toolbox-3.4.0"* folder, enter in the *"solr"* folder
Three different scripts can be used to *create*, *delete* and *override* a solr schema.

- *CREATE/UPDATE*: `solr_create_schema.sh`
- *DELETE*: `solr_delete_all_field.sh`
- *OVERRIDE*: `solr_init.sh`

Before executing any of those scripts, edit the file *solr_configuration.sh.* Set the configuration file with the command:

```
set /path_to_toolbox-3.4.0/solr/solr_configuration.sh
```

Specify the properties **SOLR_URL** and **SOLR_CORE** (i.e SOLR collection). Moreover, make sure the collection indicated by **SOLR_CORE** exists in your Solr installation with a **_default** configuration.

Subsequently, create the SOLR schema running the following command:

```
./solr_init.sh
```

In the case of a schema update, use the script *solr_create_schema.sh*. It will log an error for all existing fields, it's the normal behavior.

```
./solr_create_schema.sh
```

4.2.3 Product gap recovery

When ingesting from a source like CDSE, where products are published continuously, it may happen that some products fail to be ingested, often due to network errors. If the inner GSS error management is not enough to manage to ingest these products, a shell script is provided to ingest these products. It can be run in a cron task, for example daily, to ingest the products that were in error in the last 24 hours.

This script and all associated files are in the *script/gap-recovery* folder of the toolbox. It contains a README file explaining how to use it and configure it. Basically, it will get all the products marked in error in the database in a specific date range and query the CDSE again for all these products. Found products will be sent to Kafka and a dedicated consumer will be in charge of ingesting them.

This gap recovery tool should only be used if the error manager used in the consumers, like the Kafka error manager, did not manage to automatically ingest products in error.

4.3 Database Schema

All the tables and indexes of the database are described below. Liquibase uses the tables *databasechangelog* and *databasechangeloglock* that are not described in this document.

ingested_product Represents a product that is being or has been ingested.

Column	Type	Constraint	Indexed	Description
product_name	varchar(1024)	PK	yes	unique name of a product
product_id	uuid	not null	yes	unique identifier of ingested products
status	varchar(255)	not null	no	ingestion status
description	text		no	error reason if the ingestion failed
creation_date	timestamp	not null	no	last update date of the record
total_entries	integer		no	number of sub products to be ingested
error_entries	integer		no	number of sub products that ingestion failed
checksum	varchar(1024)		no	checksum of the product

ingester_configuration Ingesters specific configurations

Column	Type	Constraint	Indexed	Description
ingester_name	varchar(256)	PK	yes	name of the ingester
last_publication_date	timestamp		no	lastPublicationDate for OData ingesters

in_progress_download Represents a current download of a product by a user.

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	unique identifier of progress download
user_id	varchar(1024)		no	unique user identifier (username in Keycloak)
start_date	timestamp		no	download start date

quota Represents a user's assigned quota.

Column	Type	Constraint	Indexed	Description
name	varchar(512)	PK	yes	quota name
user_id	varchar(1024)	PK	yes	unique user identifier (username or user ID)
value	bigint	not null	no	value of the quota
duration	bigint		no	duration in minutes of the quota

sliding_window_quota Represents a user's currently used sliding window quota.

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	Unique identifier of sliding window quota
user_quota	varchar(512)	not null	yes	quota name (quota+userId)
acquisition_date	timestamp	PK	yes	date when a value has been consumed for this quota
value	bigint		no	value of the quota

The following tables used by DB configuration of ingesters and stores

create_ql_task represents the properties for the ingestion task "Create quicklook"

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	Unique identifier of quicklook task
pattern	varchar (1024)			Pattern that the products need to match before being processed by the task
target_stores	varchar (1024)			List of datastores where to put the quicklooks
stop_on_failure	boolean			Stop or not the task in case of failure. (Default: true)
try_limit	integer			Max tries in case of error (Default: 1)
active	boolean			Is the task active or not. (Default: true)

only_use_provided_ql	boolean			Use the QL provided by source. (Default: false)
height_ql	int			QL height. (Default: 512)
width_ql	int			QL width. (Default: 512)
name_consumer	varchar (1024)	FK		Consumer which uses the task

datastore represents the datastores created

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Datatsore name
permission_read	boolean			Defines if we can read. Always set to "true"
permission_write	boolean			Defines if we can write or not from the datastore
permission_delete	boolean			Defines if we can delete or not from this datastore
type_ds	varchar (128)			Type of datastore. Possible values (HFS, SWIFT, SWIFT_GROUP, TIME_GROUP)
properties	varchar (4096)			Definition of properties of datastore separated by ";". Example: KEEEP_PERIOD_SECONDS:300;PRIORITY:1
datastore_group_name	varchar (1024)			Name of the timebased datastore group which contains this datastore. Could be null if there is no group.

error_manager represents how to manage products in error during consumer process

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	Unique identifier of the error manager
type	varchar (128)			Type of storage for products in error. Can be ("folder" or "swift" or "s3")
containers	varchar (1024)			Container/bucket name defined for Swift/S3 storage of products in error.
credentials_name	varchar (128)			Credentials name for swift storage.
s3_credentials	varchar (256)			Credentials name for S3 storage.
error_location	varchar (4096)			Path to store products in error. It is null for swift storage.
kafka_hosts	varchar (1024)			Comma separated list of kafka nodes if using kafka for errors
kafka_topic	Varchar (256)			Name of the kafka topic if using kafka for errors
kafka_user	varchar(256)			Optional username to connect to Kafka
kafka_password	varchar(256)			Optional password to connect to Kafka
name_consumer	varchar (1024)	FK		Name of the consumer which uses the error manager

extract_md_task represents the extraction of metadata task used by consumers

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	Unique identifier of the extract md
name_consumer	varchar (1024)			Name of the consumer which uses the task
pattern	varchar (1024)			Pattern that the products need to match before being processed by the task
stop_on_failure	boolean			Stop or not the task in case of failure. (Default: true)
try_limit	integer			Max tries in case of error (Default: 1)
active	boolean			Is the task active or not. (Default: true)
force_online	boolean			Defines if we force the products to be online or keep its status. (Default: false)

folder_ingester_consumer represents the source path for a consumer with a folder source

Column	Type	Constraint	Indexed	Description
name_consumer	varchar (1024)	PK	yes	Name of the consumer which uses the source
path	varchar (4096)			Path to find the products during ingestion

folder_ingester_producer represents the source path for a producer with a folder source

Column	Type	Constraint	Indexed	Description
name_producer	varchar (1024)	PK	yes	Name of the producer which uses the source
path	varchar (4096)			Path to find the products during ingestion

s3_ingester_producer producer configuration for a S3 source

Column	Type	Constraint	Indexed	Description
name_producer	varchar (1024)	PK	yes	Name of the producer which uses the source
buckets	varchar (4096)			Comma separated list of buckets to scan
credentials_name	varchar (128)	FK		name of the S3 credentials
infinite_loop	bool			Whether or not to perpetually scan the buckets
product_directories	boolean			To indicate if we ingest directoy products, like zarr
suffix	varchar (64)			If product_directories=true, indicate the suffix of products to ingest, like ".zarr"

s3_ingester_consumer consumer configuration for a S3 source

Column	Type	Constraint	Indexed	Description
name_consumer	varchar (1024)	PK	yes	Consumer name
buckets	varchar (4096)			Comma separated list of buckets where products are located
credentials_name	varchar (128)	FK		Reference to the S3 credentials to use
product_directories	boolean			To indicate if we ingest directoy products, like zarr
suffix	varchar (64)			If product_directories=true, indicate the suffix of products to ingest, like ".zarr"

stac_ingester_producer producer configuration for STAC source

Column	Type	Constraint	Indexed	Description
name_producer	varchar (1024)	PK	yes	Producer name
service_url	varchar (4096)			URL to connect to the STAC API
collection	varchar (256)			Optional STAC collection
auth_type	varchar (256)			Authentication type used. Possible values: "basic" or "oauth2"
assumed_format	varchar (256)			Format of file we assumed to find on the source. Example: "zip"
top	int			Maximum number of products to select each time on the STAC source
client_id	varchar (256)			Client id for OAuth2 authentication
client_secret	varchar(256)			Optional client secret for OAuth2 authentication
token_end_point	varchar (4096)			OAuth2 token endpoint
service_user	varchar (256)			Login username
service_password	varchar (256)			Login password
pivot_date	varchar (256)			Name of the date field used as pivot. Default is "datetime"
pivot_date_value	varchar (256)			Value of the last pivot date. Example: "2024-01-09T00:00:00.000Z"
use_date_from_db	boolean			Whether or not to use the last pivot date saved in db from previous runs. (Default: false)
fetch_quicklook	boolean			Indicates if the producer get quicklook from source or not. (Default: false)
sortby	varchar (16384)			STAC sortby expression to sort searched products.

stac_filter stac filters used by a STAC producer

Column	Type	Constraint	Indexed	Description
id	UUID	PK	yes	Auto generated PK

name	varchar (1024)			Name of the STAC filter parameter. Example: "collections"
value	varchar (1024)			Value of the STAC filter parameter. Example: "SENTINEL-2"
stac_producer_name	varchar(1024)	FK		Corresponding STAC producer in the table stac_ingester_producer

stac_ingester_consumer consumer configuration for STAC source

Column	Type	Constraint	Indexed	Description
name_consumer	varchar (1024)	PK	yes	Consumer name
service_url	varchar (4096)			URL to connect to the STAC API
auth_type	varchar (256)			Authentication type used. Possible values: "basic" or "oauth2"
retries_on_429	int			Number of tries when receiving HTTP 429 from the source. (Default: 1)
retry_wait_on429ms	bigint			Time to wait in ms before a retry when receiving HTTP 429 from the source. (Default: 1000)
client_id	varchar (256)			Client id for OAuth2 authentication
client_secret	varchar(256)			Optional client secret for OAuth2 authentication
token_end_point	varchar (4096)			OAuth2 token endpoint
service_user	varchar (256)			Login username
service_password	varchar (256)			Login password

generate_trace_task represents the generation of trace task used by consumers (not used)

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	Unique identifier of the trace task
name_consumer	varchar (1024)	FK		Name of the consumer which uses the task
pattern	varchar (1024)			Pattern that the products need to match before being processed by the task
stop_on_failure	boolean			Stop or not the task in case of failure. (Default: true)
try_limit	integer			Max tries in case of error (Default: 1)
active	boolean			Is the task active or not. (Default: true)
private_key_path	varchar (1024)			Path to private key used to generate traces
passphrase	varchar (1024)			Passphrase used to generate traces
service_context	varchar (1024)			Context of Service
service_type	varchar (1024)			Type of service
service_provider	varchar (2048)			Trace service provider

destination_folder	varchar (4096)			Destination folder for traces generated
user_name	varchar (1024)			User name to access trace server
password	varchar (1024)			Password to access traces server
server_url	varchar (1024)			URL of the server
client_id	varchar (1024)			Client id for traces server
token_endpoint	varchar (4096)			Token endpoint to identify user on traces server
trace_type	varchar (1024)			Type of traces created. Value can be "folder" or "server"

hfs_datastore represents the source path for a producer with a folder source

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Name of the datastore
path	varchar (4096)			Path to store products
depth	int			Depth for storage directories. (Default: 2)
granularity	int			Granularity for storage directory. (Default: 2)

ingest_ds_task represents the ingestion in datastore task used by consumer

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	Unique identifier of ingest ds task
pattern	varchar (1024)			Pattern that the products need to match before being processed by the task
target_stores	varchar (1024)			List of datastores where to put the quicklooks
stop_on_failure	boolean			Stop or not the task in case of failure. (Default: true)
try_limit	integer			Max tries in case of error (Default: 1)
active	boolean			Is the task active or not. (Default: true)
name_consumer	varchar (1024)	FK		Consumer which uses the task

ingest_md_task represents the ingestion in datastore task used by consumer

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	Unique identifier of metadata task
pattern	varchar (1024)			Pattern that the products need to match before being processed by the task
target_stores	varchar (1024)			List of datastores where to put the quicklooks

stop_on_failure	boolean			Stop or not the task in case of failure. (Default: true)
try_limit	integer			Max tries in case of error (Default: 1)
active	boolean			Is the task active or not. (Default: true)
name_consumer	varchar (1024)	FK		Consumer which uses the task

ingester_consumer represents the generic information for a consumer

Column	Type	Constraint	Indexed	Description
name	uuid	PK	yes	Unique identifier of the ingester
topic_pattern	varchar (1024)			Pattern used to process many topics which match
hosts	varchar (4096)			List of kafka hosts seprated by a comma (,)
user	varchar(256)			Optional username to connect to Kafka
password	varchar(256)			Optional password to connect to Kafka
topics	varchar (1024)			Queue topics where to receive messages
group_id	varchar (1024)			Group which contains the consumer
reprocess	boolean			Defines if the consumer can process or not product already ingested. (Default: false)
poll_interval_ms	bigint			Interval of time, in milliseconds, to check for products from queue. (Default: 1200000)
parallel_ingests	int			Number of parallel ingestion the consumer can process. (Default: 4)
source_delete	boolean			Defines if the process will delete or not products from the source. (Default: false)
source_delete_pattern	varchar(1024)			Optional regexp to only delete source files matching it
source_type	varchar (128)			Indicates which type of source we have. (Values: FOLDER, SWIFT, ODATA)
ingest_threads	integer			Indicates the JVM's execution order (Default: 1)
tmp_tasks	varchar (1024)			Path where to temporary store products during ingestion

ingester_producer represents the generic information for a producer

Column	Type	Constraint	Indexed	Description
name	uuid	PK	yes	Unique identifier of the ingester

hosts	varchar (4096)			List of kafka hosts seprated by a comma (,)
user	varchar(256)			Optional username to connect to Kafka
password	varchar(256)			Optional password to connect to Kafka
topic	varchar (1024)			Queue topic where to post messages
push_interval	int			Interval of time, in seconds, to put messages in the queue. (Default: 60)
filter	varchar (4096)			Filter used to select products by name.
datastource	varchar (4096)			(Default: "Unknown")
reprocess	boolean			Defines if the consumer can process or not product already ingested. (Default: false)
production_type	varchar (1024)			Defines the trace of products as production type.
process_error_retries	integer			Number of retries in case of error. (Default: 1)
process_error_active	boolean			Defines if the system will process or not product in error. (Default: false)
source_type	varchar (128)			Indicates which type of source we have. (Values: FOLDER, SWIFT, ODATA)
process_queued_product_seconds	int			If a product has been in QUEUED state for more time than this value, requeue it if the producer sees it again

metadatastore represents the metadatastores created

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Metadatastore name
permission_read	boolean			Defines if we can read. Always set to "true"
permission_write	boolean			Defines if we can write or not from the datastore
permission_delete	boolean			Defines if we can delete or not from this datastore
type	varchar (256)			Type of metadatastore. Possible values (SOLR)

strategies	varchar (4096)			Definition of strategies for the metadatastore separated by ";". Example: RequestById:300 Different values: • RequestById • RequestByFilter • CountProducts

odata_ingester_consumer represents Odata source for a consumer

Column	Type	Constraint	Indexed	Description
name_consumer	varchar (1024)	PK	yes	Consumer name
service_url	varchar (4096)			URL to connect to the OData catalogue
auth_type	varchar (256)			Authentication type used. Possible values: "basic" or "oauth2"
odata_type	varchar (128)			Type of OData catalogue. Possible values: "csc" or "dhus"
retries_on_429	int			Number of tries when receiving HTTP 429 from cataloque. (Default: 1)
retry_wait_on429ms	bigint			Time to wait before a retry when receiving HTTP 429 from catalogue. It is in milliseconds. (Default: 1000)
client_id	varchar (256)			Client id for authentication on OData Catalogue. Can be null in case of basic authentication
client_secret	varchar(256)			Optional client secret for OAuth2 authentication
token_end_point	varchar (4096)			End point for authentication on OData catalogue. Can be null in case of basic authentication
service_user	varchar (256)			Login to access OData catalogue
service_password	varchar (256)			Password to access OData catalogue

odata_ingester_producer represents Odata source for a producer

Column	Type	Constraint	Indexed	Description
name_producer	varchar (1024)	PK	yes	Producer name
service_url	varchar (4096)			URL to connect to the OData catalogue
auth_type	varchar (256)			Authentication type used. Possible values: "basic" or "oauth2"
odata_type	varchar (128)			Type of OData catalogue. Possible values: "csc" or "dhus"
assumed_format	varchar (256)			Format of file we assumed to find on the source. Example: "zip"
top	int			Maximum number of products to select each time on the OData catalogue

client_id	varchar (256)			Client id for authentication on OData Catalogue. Can be null in case of basic authentication
client_secret	varchar(256)			Optional client secret for OAuth2 authentication
token_end_point	varchar (4096)			End point for authentication on OData catalogue. Can be null in case of basic authentication
service_user	varchar (256)			Login to access OData catalogue
service_password	varchar (256)			Password to access OData catalogue
last_publication_date	varchar (256)			Last publication date to use when selecting products from OData Catalogue. Example: "2024-01-09T00:00:00.000Z"
filter	varchar (8192)			OData filter to use for the OData Catalogue
fetch_attributes	boolean			(Default: false)
use_date_from_db	boolean			(Default: false)
fetch_quicklook	boolean			Indicates if the producer get quicklook from source or not. (Default: false)
geo_post_filter	varchar (16384)			Only for "dhus" source. Polygon to apply for filtering an area of interest
csv_file	varchar(1024)			Path to a csv file containing the ids of the products to search in the odata source

salt is used to crypt and decrypt passwords in the system (Solr metadatatastores, odata sources, swift credentials …)

Column	Type	Constraint	Indexed	Description
name	varchar (256)	PK	yes	Name of the object (swift credentials, odata consumer or producer, solr metadatastore)
value	bytea			Value of password encryption
iv	bytea			Verification field for decryption

solr_metadatastore represents solr metadatastore

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Metadatastore name
hosts	varchar (4096)			Solr hosts URL. They are separated by a comma (,)
client_type	varchar (1024)			Type of Solr client. Possible values: "LBHttp" or "SolrCloud"
user_name	varchar (1024)			User name to connect to Solr
password	varchar (1024)			Password to connect to Solr
collection	varchar (1024)			Collection where the metadata are indexed

default_sort	varchar (128)			Sort to apply on the search queries. It is in the form "<solr.field.name> asc/desc". Can be null. Example: "name asc"
default_top	int			Default top to apply on Solr results
max_skip	int			Maximum number of products to skip per Solr request
storage	varchar (2048)			Defines the host where the storage of metadata is done. In this case, Solr is just used as an index. Not used in the current configuration.

swift_credentials represents swift credentials to access swift storage

Column	Type	Constraint	Indexed	Description
name	varchar (128)	PK	yes	Credential name
tenant	varchar (1024)			Tenant name
password	varchar (1024)			Password to access object storage (encrypted)
user_name	varchar (1024)			User name to access object storage
authentication_url	varchar (4096)			URL for authentication on object storage
region	varchar (1024)			Region of the object storage
domain	varchar (256)			Domain of the object storage to access. (Default: "Default")

swift_datastore represents swift storage for products

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Datastore name
container	varchar (4096)			Container used to store products
prefix_location	varchar (4096)			Description of the storage of products. Example: instrument/productType/year/month/day
credentials_name	varchar (255)	FK		Reference to the credentials to use for this datastore

swift_datastore_group represents swift storage group for products

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Datastore name
prefix_location	varchar (4096)			Description of the storage of products. Example: instrument/productType/year/month/day
credentials_name	varchar (255)	FK		Reference to the credentials to use for this datastore

first_chars	integer			In case of date container pattern, the number of chars to take to build it.
time_splitter	varchar (64)			In case of date container pattern, the time unit to use. Different values: • "DAY" • "MONTH" • "QUARTER" • "SEMESTER" • "YEAR"
regexp	varchar (4096)			In case of name container pattern, apply the regexp on the products name to put them in the containers.
pattern_mapper	varchar (4096)			Pattern to map products into containers. It is the most used currently.
filter	varchar (1024)			Filter products on the name

swift_ingester_consumer represents swift source for consumer

Column	Type	Constraint	Indexed	Description
name_consumer	varchar (1024)	PK	yes	Consumer name
containers	varchar (4096)			Container used to store products
credentials_name	varchar (128)	FK		Reference to the credentials to use for this datastore

swift_ingester_producer represents swift source for producer

Column	Type	Constraint	Indexed	Description
name_producer	varchar (1024)	PK	yes	Producer name
containers	varchar (4096)			Container used to store products
credentials_name	varchar (128)	FK		Reference to the credentials to use for this datastore
infinite_loop	boolean			Enable or disable infinite scan of containers

s3_credentials S3 credentials

Column	Type	Constraint	Indexed	Description
name	varchar (128)	PK	yes	Credentials name
access_key	varchar (1024)			Access key
secret_key	varchar (1024)			Secret key (encrypted)
endpoint	varchar (4096)			Endpoint URL
region	varchar (1024)			Region

s3_datastore S3 datastore (bucket)

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Datastore name
bucket	varchar (4096)			Bucket name
prefix_location	varchar (4096)			Prefix for storing products Example: instrument/productType/year/month/day
credentials_name	varchar (255)	FK		Reference to the S3 credentials to use for this datastore

s3_datastore_group Logical group of S3 datastores (buckets)

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Datastore name
prefix_location	varchar (4096)			Prefix for storing products Example: instrument/productType/year/month/day
credentials_name	varchar (255)	FK		Reference to the S3 credentials to use for this datastore
first_chars	integer			In case of date container pattern, the number of characters used to build it.
time_splitter	varchar (64)			In case of date container pattern, the time unit to use. Different values: • "DAY" • "MONTH" • "QUARTER" • "SEMESTER" • "YEAR"
regexp	varchar (4096)			In case of name container pattern, apply the regexp on the products name to put them in the containers.
pattern_mapper	varchar (4096)			Pattern to map products into containers. It is the most used currently.
filter	varchar (1024)			Filter products on the name

timebased_datastore_group represents time based datastore group.

Column	Type	Constraint	Indexed	Description
name	varchar (1024)	PK	yes	Datastore name
filter	varchar (1024)			Filter to apply on product's name
datastore_policy	varchar (1024)			Datastore policy used on the datastore. **Possible values:** "USER_DEFINED_PRIORITY_POLICY" or "BASIC_STORE_PRIORITY_POLICY"

timebased_components is the table representing the components of datastore group and datastores

Column	Type	Constraint	Indexed	Description
ds_group_name	varchar (255)	PK	yes	Datastore group name
ds_name	varchar (255)			Datastore name

product_eviction_by_time is the table representing the subscription for deletion events.

Column	Type	Constraint	Indexed	Description
store_name	varchar (255)	PK	yes	Datastore names
product_id	uuid			The id of products
product_name	varchar (1024)			Stores the product names
source	varchar (255)			Child of the timebased datastore group
eviction_date	timestamp			Date and time of the eviction occurrence
keep_period	integer			Period in seconds, a product will be kept in a store
processing	boolean			The processing column is set to "true" when the eviction process handles the corresponding product, so as not to process the same product twice

job_deletion_products represent the deletion job.

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	autogenerated
name	varchar (512)			A name to describe the deletion
notification_message	varchar(1024)			Optional message that will be added in notifications sent to end-users
odata_filter	varchar (4096)			Filter for products in stores
reason	varchar (128)			Reason of the deletion **Enum:** • CORRUPTED_PRODUCT • WRONG_CHECKSUM • DUPLICATED_PRODUCT • OBSOLETE_PRODUCT
status	varchar (128)			Status of the job - **Enum:** CREATED: initial deletion job status RUNNING: to execute a deletion job PAUSED: to pause a running job CANCELED: to cancel a previously created job DONE: a job execution completion Default job status on creation CREATED
creation_date	varchar (128)			Creation date of the job
updated_date	varchar (128)			Date where the job was updated
nb_products_error	int			During deletion, the number of products in error
nb_products	int			Number of products to delete / deleted

nb_threads	int			Default value: 1 Number of threads to launch for the job

subscription is the table representing the subscription for deletion events.

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	Unique identifier of the subscription
status	varchar (255)			Status of the subscription
subscription_event	varchar (255)			Event occurring on products and listened by the subscription. Default value: DELETED
filter_param	Text			OData filter to apply on products Ex: Products?$filter=contains(Name,'S')
submission_date	timestamp			Date when the subscription was called
last_query_date	timestamp			Last date when a notification was sent to the endpoint
notification_ep	varchar (255)			URL of the endpoint
notification_ep_username	varchar (255)			Username to connect to the endpoint
notification_ep_password	varchar (255)			Password to connect to the endpoint
username	varchar (1024)			User who created the subscription. Default: JonhDoe

5. Docker-compose

A Docker compose setup to run CDH is available. The configuration files must be updated to fit operational scenarios.

5.1 Pre-requesites

We recommend to install:

- Docker: 20.10.12 or later

5.2 Distribution Links

This software is available in one format, a zip archive. It is available at https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-compose/3.4.0/cdh-compose-3.4.0.zip
The archive must be downloaded and unzipped to launch containers.

```
/cdh-compose-3.4.0
├── admin/
├── backend/
├── catalogue/
```

```
├── notification/
├── ingest /
└── stac/
```

5.3 Docker compose Operational Sample

```
producer-S1:
    image: "gaeldockerhub/cdh-ingest:${TAG}"

    volumes:
     - ./gss-producer-S1.xml:/ingest/etc/gss.xml
     - ./logs:/ingest/logs
     - ./log4j2.xml:/ingest/etc/log4j2.xml

consumer-S1:
    image: "gaeldockerhub/cdh-ingest:${TAG}"
    volumes:
    - ./gss_consumer-dest-folder_S1.xml:/ingest/etc/gss.xml
    - /mnt/dhsnas/IVV_PLATFORM/GSS/DEST_FOLDER_GSS_3.4.0/S1:/ingest/folder1
    - /mnt/dhsnas/IVV_PLATFORM/GSS/DEST_FOLDER_GSS_3.4.0/error:/ingest/error
    - /mnt/dhsnas/IVV_PLATFORM/GSS/DEST_FOLDER_GSS_3.4.0/tmp:/ingest/tmp
    - ./logs:/ingest/logs
       #networks:
       #backend:
       #name: backend
       #external: true
```

5.4 Backend Infrastructure

The backend is composed of different components:

- Zookeeper (3 instances)
- Kafka (2 instances) or Kafka with authentication (2 instances)
- Solr (3 instances)
- PostgreSQL (1 instance)

Every component shares the same network named **backend** (not mandatory to use such network). Data for each component is persisted on the host machine into Docker managed volumes.

Note: It is not mandatory to install Zookeeper, Kafka, Solr and PostgreSQL components using the "Backend" infrastructure. They can be installed and configured individually following the proper operational needs.

5.4.1 Startup

To start the backend infrastructure, go to the **backend** directory and run:

```
docker compose up -d
```

You can use the following command to start backend in background:

```
docker compose up -d
```

You can check the logs with:

```
docker compose logs -f
```

If the database and solr schema must be created/updated, run the toolbox container:

```
docker compose run init
```

You can add some environement variable to the toolbox if you are, for example, using authentication for Solr. This can be edited in the service **init** in the **docker-compose.yml** file.
For example:

```
  init:
    image: "gaeldockerhub/cahu-toolbox:${TAG}"
    environment:
      JDBC_URL: jdbc:postgresql://postgres:5432/postgres
      DB_LOGIN: postgres
      DB_PASSWORD: password
      SOLR_URL: http://solr-1:8983/solr
      SOLR_CORE: cahu
      SOLR_USER: user
      SOLR_PASSWORD: password
    networks:
      - backend
    depends_on:
      - postgres
      - solr-1
      - solr-2
      - solr-3
    profiles:
      - init
```

5.4.2 Shutdown

To simply stop the backend infrastructure without deleting containers, execute the command:

```
docker compose stop
```

If you want to also delete containers (without deleting volumes):

```
docker compose down
```

If you want to also delete the volumes (**all data will be lost**):

```
docker compose down -v
```

5.4.3 Use Kafka authentication

You can start the backend with Kafka authentication enabled. It will use a configurable user/password. These credentials can be edited in the file **kafka_server_jass.conf**.

Note that the credentials will be transmitted in plain text, so if you use such configuration in a production environement, make sure your kafka nodes are accessed over HTTPS.

Use this command to start the whole backend with kafka authentication enabled:

```
docker compose --profile kafka_secure up -d
```

And use this command to stop the whole backend:

```
docker compose --profile kafka_secure down
```

5.5 Ingestion

5.5.1 Configure the Ingesters

5.5.1.1 XML Configuration

Go to the **ingest** directory. Configure the **gss-producer.xml** and **gss-consumer.xml** configuration files according to your needs. The provided files will ingest products from CDSE into a local directory (configured as an attached volume in **docker-compose.yml**) and into Solr.

RECOMMENDATION

You can find some operational configuration sample files into cdh-ingest-3.4.0.zip file. These files can be used for ingesters configuration. They fit the main operational scenarios known. The files are stored into **"operational-samples"** directory.

Please note that the legacy DHuS service was discontinued on 30 June 2025. The corresponding configuration is retained for backward compatibility and reference purposes.

5.5.1.2 Admin API Configuration

Go to **ingest** directory.

Configure **database-configuration-for-ingestion.properties** to get ingesters and datastores from database. This file allows us to define:

- The database where the configuration is stored (URL, user and password …)
- The list of datastores to use for ingestion.
- The list of metadatastores to use for ingestion.
- The list of producers to use for ingestion.
- The list of consumers to use for ingestion.
- Swift configuration to access swift containers.
- A secret key password to encrypt passwords used for Metadatastores or ingesters

NOTE: The Ingesters and Stores need to be configured previously using Admin API.

```
# DB configuration
db.url=jdbc:postgresql://<postgres.url>:<postgres.port>/<db.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=2
#db.socketTimeout=
#db.networkTimeout=


# CDH Configuration

## Datastore's names
cdh.datastores=datastore-name1;datastore-name2
## Metadatastore's names
cdh.metadatastores=metadatastore-name1;metadatastore-name2
## Producer's names
cdh.producers=producer-name1;producer-name2
## Consumer's names
cdh.consumers=consumer-name1;consumer-name2

# Generic Swift Configuration
#swiftConfiguration.segmentSizeMB=100
```

```
#swiftConfiguration.retries=5
#swiftConfiguration.retryDelayMs=50
#swiftConfiguration.maxConnections=20

# Secret for encrypt
secret.key.password=test
```

Please rename the file **docker-compose-with-database-configuration.yml** to docker-**compose.yml** in order to enable the DB configuration.

5.5.2 Launch the Ingesters

Launch 1 producer and n consumers
The following command makes sense only in the case the ingestion process has been configured via XML files.

```
docker compose up -d --scale consumer=n
```

You can use the following command to start in background:

```
docker compose up -d
```

The command to start different services separately (use in case of configuration via Admin API)

```
docker compose up <service-1> <service-2>
```

where <service-X> is the name of producers or/and consumers to start. You can find the value into **docker-compose.yml** file. Ensured that the properties file contains only the correct ingester (consumer or producer) to launch. Example: extract of **database-configuration-for-ingestion.properties** where consumers are disabled (commented):

```
## Producer's names
cdh.producers=producer-name1;producer-name2
## Consumer's names

#cdh.consumers=consumer-name1;consumer-name2
```

The command to stop different services separately

```
docker compose stop <service-1> <service-2>
```

The command to stop and remove all the ingestion processes is:

```
docker compose down -t 600
```

5.5.3 Stop the Ingesters

To stop without container remotion the ingesters, execute the command:

```
docker stop <service-1> <service-2>
```

A timeout should be added to properly stop the ingesters, re and let them finish ongoing ingestions:

```
docker compose down -t 600
```

5.6 Catalogue (OData API)

5.6.1.1 XML Configuration

Go to the **catalogue** directory. Configure the **gss-catalogue.xml** configuration file according to your needs. The provided file will match the DataStores and MetadataStores from the ingester. Authentication and quotas are disabled by default.

RECOMMENDATION
You can find some operational configuration sample files into cdh-catalogue-3.4.0.zip file. They fit to main of operational scenarios known. The files are stored into **"operational-samples"** directory.

The file **application.properties** describes some properties needed to launch the catalogue.

This file is in *cdh-compose/catalogue/application.properties*.

5.6.1.2 Database Configuration

Go to the **catalogue** directory.

Configure **application.properties** to enable database configuration.

This file is in *cdh-compose/catalogue/operational-sample/application.properties*.sample

The stores must be created with the cdh-admin-api (See section 12.5).

5.6.2 Launch the Catalogue

To launch a catalogue instance, execute the command:

```
docker compose up
```

You can use the following command to start in background:

```
docker compose up -d
```

The API will be accessible at http://localhost:8081/odata/v1.

5.6.3 Stop the Catalogue

To stop without container remotion the catalogue instance, execute the command:

```
docker stop <catalogue-service>
```

To stop and remove the catalogue instance, execute the command:

```
docker compose down
```

5.7 Admin (REST API)

Go to admin directory. Configure **application.properties** file according to your needs. It will match the database from the **backend** part. Authentication is disabled.

This file is in *cdh-compose/admin/application.properties*.

5.7.1 Launch Admin API

To launch an admin API instance, execute the command:

```
docker compose up
```

You can use the following command to start backend in background:

```
docker compose up -d
```

The API will be accessible at http://localhost:8082/gss-admin-api.

5.7.2 Stop Admin API

To stop without container remotion the admin-API, execute the command:

```
docker stop <admin-api_service>
```

To stop and remove the admin-API, execute the command:

```
docker compose down
```

5.8 Notification

Go to the **notification** directory. Configure consumer-for-notification.properties file according to your needs. It will match the database set in the **backend** part.

This file is in *cdh-compose/notification/consumer-for-notification.properties*.

5.8.1 Launch Notification

To launch a notification instance, execute the command:

```
docker compose up
```

You can use the following command to start backend in background:

```
docker compose up -d
```

5.8.2 Stop Notification

To stop without container remotion, execute the command:

```
docker stop <notification_service>
```

To stop and remove notification, execute the command:

```
docker compose down
```

5.9 STAC API

The STAC (SpatioTemporal Asset Catalog) API is a specification designed to facilitate the search and retrieval of geospatial data. STAC API enables users to query geospatial assets like satellite images or aerial photos by defining parameters such as location Bbox (via bounding boxes) and time ranges. STAC API's core endpoint is /stac/search, which returns results in the GeoJSON format, which is widely used for representing geographical data.

Go to **stac** directory. Configure **application.properties** and enable or disable Keycloak authentication.

This file is in *cdh-compose/stac/application.properties*.

IMPORTANT
If you want to explore nodes of a product, it is important to configure as true the following property in application.properties file **`stac.item.showNodes`** .

5.9.1 Launch STAC API

To launch a STAC API instance, execute the command:

```
docker compose up
```

You can use the following command to start backend in background:

```
docker compose up -d
```

The API will be accessible at http://localhost:8080/stac or http://localhost:8080/<stac-servlet>/stac

5.9.2 Stop STAC API

To stop without container remotion, execute the following command:

```
docker stop <stac_services>
```

To stop and remove STAC, execute the following command:

```
docker compose down
```

5.9.3 Custom JVM arguments

You can add/override JVM arguments for all the components. Use the environment variable JAVA_OPTS and set any parameters you want. Its primary use is to let the user control the heap size.

For example: **JAVA_OPTS=-Xms512m -Xmx4g**

6. Admin API

GSS Admin API is named CDH-Admin. It is a REST API exposing different endpoints to configure GSS components (ingests, user's quotas, datastores, metadatastores, ...). The information is stored in a database used by the different components.

6.1 Pre-requisites

6.1.1 Infrastructure Requirements

We recommend to install:

- Docker: 20.10.12 (or later)

6.1.2 Network Requirements

Application	Default port
PostgreSQL	5432
Keycloak version 18 or later	8443

6.1.3 Software Requirements

Some tools must be installed in order to use the catalogue:

- A running PostgreSQL instance, (10.12 and after: https://www.postgresql.org/download/) with a database for the software.
- A running Keycloak instance if you want to use authentication (18.0.0 and later: https://www.keycloak.org/downloads)

The database schema update/creation scripts are inside the **toolbox** module. The database should be updated before launching a new version of the Admin API.

6.2 Distribution Links

This software is available in two formats, a zip archive and a Docker image.

6.2.1 Zip Archive

Available at https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-admin-api/3.4.0/cdh-admin-api-3.4.0.zip
The archive must be downloaded and unzipped in order to launch the Admin API.

The zip archive is presented as follows:

```
cdh-admin-api/
├── etc/
│   ├── JSON-samples/
│   │   └── deletion.json
│   ├── application.properties
│   ├── docker.compose.yml
│   └── log4j2.xml
│
├── lib/
├── logs/
├── start.sh
└── stop.sh
```

The **application.properties** file of Admin API is used for the Keycloak server configuration and application configuration. This file is self described. Change the value of any properties according to your needs. An **application.properties** file must be present in the *etc/* directory to launch Admin API.

The **log4j2.xml** indicates where and how the logs of the application are written. It can be modified at will. The shipped configuration writes the logs inside a *logs/* directory in a file named *cdh-admin-api.log*. This files is rolled every day and older logs files are saved in the format *logs/cdh-admin-api-YYYY-MM-DD.log*.

With ZIP distribution, use the *start.sh* script to launch the catalogue and use *stop.sh* to gracefully stop it.

```
nohup ./start.sh &
./stop.sh
```

6.2.2 Docker Image

Available at Docker Hub `docker pull registry.hub.docker.com/gaeldockerhub/cdh-admin-api:3.4.0`

You can check that the Docker image has been downloaded with the command `docker image list | grep cdh-admin-api`. The Docker image is based on an *openjdk:17-jdk-slim* image.

The Admin API application will be launched in a created */admin-api* directory inside the Docker image. Don't forget to expose the port of the admin-api when lauching the Docker container.

To launch the Docker container, the **application.properties** files as attached volumes. Examples for those files are available in the Zip archive.

Example:

```
docker run -d --name cdh-admin-api -v /path/application.properties:/admin-api/etc/application.properties -it registry.hub.docker.com/gaeldockerhub/cdh-admin-api:3.4.0
```

The **-d** option is used to run the Docker container in a detached mode. If you want your Docker container to point to the *localhost* of the hosting machine, add the launch option `--network="host"` (https://docs.docker.com/network/host/). It means that the container shares the host machine network, the container will have the same IP and ports as the host.

For example, if you launch the container and you want to access the cdh-admin-api on localhost:8080, you must use this option. You can map the log files of the application with a local folder by adding an attached volume:

```
-v /your/path/to/logs:/admin-api/logs
```

To gracefully stop the container, add a timeout (in seconds) to the stop command:

```
docker stop -t 60 cdh-admin-api
```

6.3 Configuration

6.3.1 Admin API Configuration

Configure **application.properties** according to your needs. This file is self described and is located in *cdh-admin-api/etc/application.properties*.

6.3.2 Admin-API Keycloak Configuration

CDH-Admin-API requires Keycloak version 18 or later if you want to enable authentication. To ensure the users access to the CDH-Admin a Keycloak setting must be enabled through the Keycloak console page.

6.3.2.1 Keycloak Web-API Authentication Configuration

The Web-API authentication uses the OAuth 2.0 protocol to protect resource requests by verifying authorized access tokens. To enable this authentication, it is essential to add user/users to both the Keycloak client-access role and realm-access role.

To do so, the following steps must be taken:

- Access the Keycloak admin console.
- Select the relevant Keycloak realm.

1. **Keycloak Realm-access:**
   - Within your desired realm, navigate to the "Realm" section in the left sidebar.
   - To ensure that the user is listed in roles for realm-access, please follow the appropriate steps:
     - `Admin-console → Realm roles → <your-role> → Users in role → <desired-user>`
2. **Keycloak Client-access:**
   - Within your desired realm, navigate to the "Clients" section in the left sidebar.
   - Here, you'll see a list of clients associated with the current realm. Initially, you might see some default clients like "account" and "realm-management."
   - To ensure that the user is listed in roles for client-access, please follow the appropriate steps:
     - `Admin-console → Clients → <your-client> → roles → <your-role> → Users in role → <desired-user>`
3. **Keycloak Users:**
   - To begin, go to the "Users" section in the left sidebar of your desired realm.
   - You will find a list of users here.
   - It's important to note that there are two types of roles
     - The `realm-access role`.
     - The `client-access role`.
   - Role mapping for the realm-access role:
     - `Admin-console → Users → <desired-user> → Role Mapping → Assign role → Filter by realm roles → Filter by clients → <your-role> → Assign`
   - Role mapping for the client-access role:
     - `Admin-console → Users → <desired-user> → Role Mapping → Assign role → <your-role> → Assign`

Note:

- Please ensure to see the user/users in (Users in role) of realm-access and client-access.
- The settings in the Keycloak console may differ depending on the version utilized.
- The following website allows to see the users and roles:

  - Generate a token.
  - Decode the token.
  - Verification of the token containing the roles at the https://jwt.io
- Please refer to RD-4 for the complete Keycloak configuration.

User role mapping example in realm and client access below:

```
{
  "realm_access": {
    "roles": [
      "admin-role",              // realm-access role
      "default-roles-admin-realm",
      "offline_access",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "user-client": {
      "roles": [
        "admin-role"             // client-access role
      ]
    },
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "preferred_username": "cdh-admin"
}
```

6.3.3 Keycloak Authentication

The table below describes the Keycloak roles needed by users to perform operations on the system.

User Role	Description	DHS Actor (admin / user)
end-user	End user with just right to list products, have details on them, download them	
admin	A user with all rights on Ingesters, datastores and metadatastores	

keycloak.role.readonly	A user with readonly role to access ingester endpoints to retrieve information about ongoing data retrieval process	

Note: Please ensure to use Keycloak version 18 or later for optimal compatibility and security.

The **application.properties** file of CDH-Admin API can be used to enable authentication on Admin API through a Keycloak server. To use OAuth2 authentication through command line:

- Request a token:

```
curl -d 'client_id=<keycloak_client>' -d 'username=***' -d 'password=***' -d
'grant_type=password' "https://<keycloak_url>/auth/realms/<keycloak_realm>/protocol/openid-
connect/token"
```

You will obtain a JSON response containing the *access_token*

- use this token in your requests

```
curl -X GET -H 'authorization: Bearer <access_token>' -H 'content-type: application/json'
"http://<server.url>:<server.port>/"
```

6.3.4 Other

A sample of the configuration file is provided in the distribution, in *etc/application.properties.sample*. Configure this file according to your needs and rename it to *application.properties*.
The below lines are used to configure the port on which the application will be listening and the database properties.

```
# Port exposed by the server. If you use a dockerized launch, expose this port.
server.port = 8080

# Database
spring.datasource.url = jdbc:postgresql://<server>:<port>/<database_name>
spring.datasource.username = <username>
spring.datasource.password = <user.password>
spring.datasource.driver-class-name = org.postgresql.Driver
```

It is also important to configure which information will be used to identify users. So, you need to provide the correct value to parameter "***user-name-attribute***" in **application.properties**.

There are 2 possibilities:

- preferred_username: using the preferred user's name configured in Keycloak to identify the user in logs and for quotas management.
- user_id: using the user id defined in Keycloak to identify the user in logs and for quotas management.

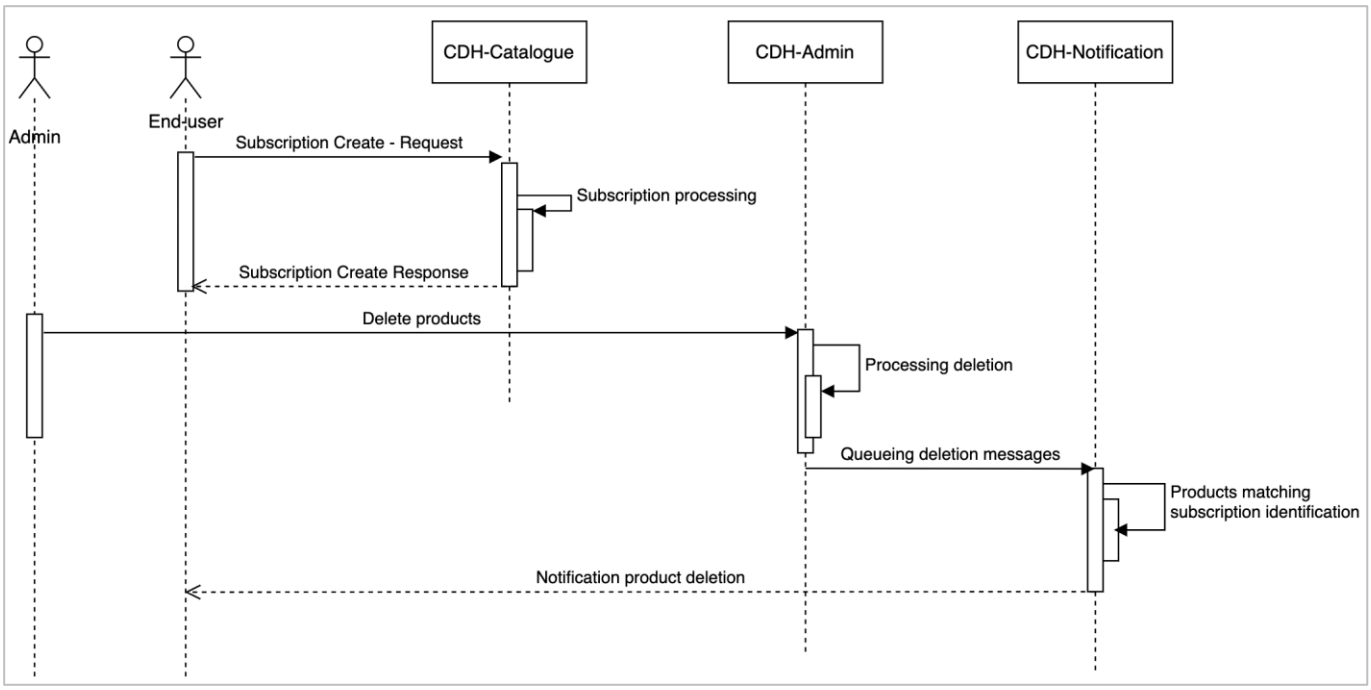
Important

```
# Which attribute to use to identify the user (displayed in logs and used for quotas)
# Possible values are: user_id, preferred_username. Default value is user_id.
user-name-attribute=user_id
```

7. Product Deletion Feature

The software offers the possibility to delete products from stores: it is like a hard eviction performed on the DHuS.

We can represent the process with the following architecture:



- End-user creates a subscription (on Catalogue) with an OData filter to be informed on deletion event on products matching the filter.
- Admin user deletes products which are matching an OData filter.
- Deletion messages are queued into a configured Kafka queue.
- Notification component will read messages in deletion queue and send notifications to all the subscriptions matching for each product.

In the *application.properties* file, the following parameters are used to control deletion behavior. From which stores products will be deleted (every store if left blank), and if notifications will be sent.

```
############ PRODUCT DELETION
# To perform deletion of products, the configuration of stores need to be in database.
# You can choose from which stores product will be deleted. If not, will delete from all
configured stores.

## Example of configuration
# Datastores name where to delete products separated by a semi-colon (;)
```

```
#deletion.datastores=HFS1-local
# Metadatastores name where to delete products separated by a semi-colon (;)
#deletion.metadatastores=solr-local

########### Notification for products
# Enable or not, default is false
notification.enabled = false
# Kafka nodes where notification messages will be sent
notification.kafka.hosts = localhost:29092;localhost:39092
# Kafka optional user/password
#notification.kafka.user = kafka
#notification.kafka.password = pass123
# Name of the topic where notification messages for deleted products will be sent, default
is notification_deleted
notification.deletion.topic=notification_deleted
```

8. Ingest

Ingest any products by following a consumer/producer paradigm using Kafka. Products will be stored inside configurable DataStores (HFS, Swift, S3) and their metadata will be saved inside MetadataStores (Solr).

Storage and ingester configuration is done in a file named **gss.xml** (described in this chapter) or using the Admin API (see chapter 10)

8.1 Pre-requisites

8.1.1 Infrastructure Requirements

We need to install:

- Docker: 20.10.12 (or later)

8.1.2 Network Requirements

Application	Default port
PostgreSQL	5432
SolR	8983

8.1.3 Software Requirements

The CDH-Ingest requirements are:

- A running Kafka instance (3.3.1 and later: https://kafka.apache.org/downloads)
- A running SolrCloud instance, (9.0.0 and later: https://lucene.apache.org/solr/downloads.html). It is the same instance created by **toolbox** (with JQ and JTS installed).
- A running PostgreSQL instance, (10.12 and later: https://www.postgresql.org/download/). It is the same instance created by **toolbox**.
- Open JDK 17

The Solr schema and the database schema update/creation scripts are inside the **toolbox** module.

8.2 Distribution

This software is available in two formats, a zip archive and a Docker image.

8.2.1 Zip Archive

Available at https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-ingest/3.4.0/cdh-ingest-3.4.0.zip

The zip archive structure is:

```
cdh-ingest/
├── etc
│   ├── database-conf
│   │   └── database-configuration-for-ingestion.properties
│   ├── folder
│   │   ├── gss-consumer.xml
│   │   └── gss-producer.xml
│   ├── JSON-samples
│   │   ├── Consumer
│   │   │   ├── CDSE
│   │   │   │   ├── consumer_cdse_source_dest_folder_error_folder.json
│   │   │   │   ├── consumer_cdse_source_dest_swift_error_swift.json
│   │   │   │   ├── consumer_cdse_stac_source_dest_folder_error_folder.json
│   │   │   │   └── consumer_cdse_source_dest_s3_error_s3.json
│   │   │   ├── DHuS
│   │   │   │   ├── consumer_dhus_source_dest_folder_error_folder.json
│   │   │   │   └── consumer_dhus_source_dest_swift_error_swift.json
│   │   │   ├── Folder
│   │   │   │   ├── consumer_folder_source_dest_folder_error_folder.json
│   │   │   │   └── consumer_folder_source_dest_swift_error_swift.json
│   │   │   ├── S3
│   │   │   │   └── consumer_s3_source_zarr_dest_s3.json
│   │   │   └── GSS
│   │   │       ├── consumer_GSS_source_dest_folder_error_folder.json
│   │   │       └── consumer_GSS_source_dest_swift_error_swift.json
│   │   ├── Datastore
│   │   │   ├── HFS
│   │   │   │   ├── datastore_quicklook.json
│   │   │   │   └── timebased_datastore.json
│   │   │   ├── S3
│   │   │   │   ├── datastore_group.json
│   │   │   │   └── datastore.json
│   │   │   └── SWIFT
│   │   │       ├── datastore_quicklook.json
│   │   │       └── timebased_datastore.json
│   │   ├── Metadatastore
│   │   │   └── metadatastore.json
│   │   ├── Producer
│   │   │   ├── producer_cdse_source.json
│   │   │   ├── producer_cdse_stac_source.json
```

```
│   │   │   ├── producer_dhus_source.json
│   │   │   ├── producer_folder_source.json
│   │   │   ├── producer_GSS_source.json
│   │   │   ├── producer_prip_source.json
│   │   │   ├── producer_s3_source.json
│   │   │   └── producer_s3_zarr_source.json
│   │   ├── Quota
│   │   │   └── quota_user.json
│   │   ├── S3-credentials
│   │   │   └── s3_credentials.json
│   │   └── SWIFT-credentials
│   │       └── swift_credentials.json
│   ├── log4j2.xml
│   ├── obj-storage
│   │   ├── gss-consumer.xml
│   │   └── gss-producer.xml
│   ├── odata-csc
│   │   ├── gss-consumer.xml
│   │   └── gss-producer.xml
│   ├── odata-dhus
│   │   ├── gss-consumer.xml
│   │   └── gss-producer.xml
│   ├── operational-samples
│   │   ├── admin-api-conf
│   │   │   ├── docker-compose.yml
│   │   │   ├── INGESTION_DHUS-HFS_S1.properties
│   │   │   ├── INGESTION_DHUS-HFS_S2.properties
│   │   │   ├── INGESTION_DHUS-HFS_S3.properties
│   │   │   ├── INGESTION_DHUS-HFS_S5.properties
│   │   │   ├── INGESTION_DHUS-SWIFT_S1.properties
│   │   │   ├── INGESTION_DHUS-SWIFT_S2.properties
│   │   │   ├── INGESTION_DHUS-SWIFT_S3.properties
│   │   │   └── INGESTION_DHUS-SWIFT_S5.properties
│   │   ├── log4j2.xml
│   │   ├── source-csc
│   │   │   ├── gss_consumer-dest-folder.xml
│   │   │   ├── gss_consumer-dest-swift.xml
│   │   │   └── gss_producer.xml
│   │   ├── source-dhus
│   │   │   ├── gss_consumer-dest-folder.xml
│   │   │   ├── gss_consumer-dest-s3.xml
│   │   │   ├── gss_consumer-dest-swift.xml
│   │   │   └── gss_producer.xml
│   │   └── source-folder
│   │   ├── gss_consumer-dest-folder.xml
│   │   ├── gss_consumer-dest-swift.xml
│   │   └── gss_producer.xml
│   ├── s3
│   │   ├── gss-consumer.xml
```

```
│   │   ├── gss-producer.xml
│   │   ├── gss-consumer-zarr.xml
│   │   └── gss-producer-zarr.xml
│   └── stac
│   ├── gss-consumer.xml
│   └── gss-producer.xml
├── lib/
├── logs/
├── README.md
├── start.sh
└── stop.sh
```

The **gss-consumer.xml** and **gss-producer.xml** files are examples for a consumer and producer **gss.xml** configuration. A **gss.xml** file must be present in the *etc/* directory to launch the ingester. You can have multiple ingesters in the same **gss.xml** file.

The **properties** files are examples for launching ingesters that have been previously configured and saved in database with **cdh-admin-api**.

The **log4j2.xml** indicates where and how the logs of the application are written. It can be modified at will. The shipped configuration writes the logs inside a *logs/* directory in a file named *cdh-ingest.log*. This files is rolled every day and older logs files are saved in the format *logs/cdh-ingest-YYYY-MM-DD.log*.

With ZIP distribution, use the *start.sh* script to launch ingester(s) and use *stop.sh* to gracefully stop it. It will use the **gss.xml** file located in **etc/** directory.

```
nohup ./start.sh &
./stop.sh
```

If you want to use a properties file to launch ingester, specify the location of this file with an environment variable.

```
export         CONF_FILE=etc/database-conf/database-configuration-for-ingestion.properties
nohup ./start.sh &
./stop.sh
```

8.2.2 Docker Image

Available on Docker Hub -> `docker pull registry.hub.docker.com/gaeldockerhub/cdh-ingest:3.4.0`.

The Docker image is based on an *openjdk:17-jdk-slim* image. The ingest application will be launched in a created */ingest* directory.

To launch the Docker container, you must provide the **gss.xml** as an attached volume. If you want to ingest products from a folder or use any folder on the host machine, these folders should be added as attached volumes to the Docker launch command.

Example with XML configuration file:

```
docker run -d --name cdh-ingest -v /path/gss.xml:/ingest/etc/gss.xml -it
registry.hub.docker.com/gaeldockerhub/cdh-ingest:{project.version}
```

Example with properties configuration file:

```
docker run -d --name cdh-ingest -v
/path/to/myFile.properties:/ingest/etc/myFile.properties -e
"CONF_FILE=/ingest/etc/myFile.properties" -it
registry.hub.docker.com/gaeldockerhub/cdh-ingest:{project.version}
```

The **-d** option is used to run the Docker container in a detached mode. If you want your Docker container to point to the *localhost* of the hosting machine, add the launch option `--network="host"` (https://docs.docker.com/network/host/). It means that the container shares the host machine network, the container will have the same IP and ports as the host.

It is possible to map the log files of the application with a local folder by adding an attached volume:

```
-v /your/path/to/logs:/ingest/logs
```

8.3 Configuration

Self-described xml example configurations can be found in the *etc/* directory. An **ingester** in **producer mode** must specify a **topic** where produced messages will be sent to **Kafka**. An **ingester** in **consumer mode** must specify a list of **topics** or a **topicPattern** from where messages will be consumed.

If you want multiple consumers to read messages from the same topic, each one of them must have the same **groupId**. Each **consumer** can ingest multiple products in parallel, with the **parallelIngests** tag.

Note: the parameter **ingestThread** has no effect and is not used in this version.

8.3.1 Producer Configuration

A producer can reproduce messages for products in error state.Add this to your producer configuration:

```
<ingester:processError active="true" retries="3">
```

The number of retries attempt is not persisted in database.

If the products in error have been put in a specific folder, you can configure a producer/consumer to handle those products. The producer must be configured to reproduce message for products in error. The consumer should be configured to not move products in error.

It's possible to re-ingest already ingested products and overwrite them. A product is considered to be ingested if its status is **INGESTED** in database. If so, this product is eligible for being reprocessed.

In the *producer* and in the *consumer*, simply add the element

```
<ingester:reprocess>true</ingester:reprocess>
```

8.3.2 Consumer Configuration

If an error occurs during the ingestion of a product, the latter can be copied to a configurable location. If you want to copy products in error in a directory:

```
<ingester:errorManager type="folder">
  <ingester:errorLocation>/your/folder/path</ingester:errorLocation>
</ingester:errorManager>
```

If you want to copy products in a swift container:

```
<ingester:errorManager type="swift">
  <ingester:credentials name="credentials">
    <ds:tenant>test</ds:tenant>
    <ds:password>***</ds:password>
    <ds:user>test</ds:user>
    <ds:url>***</ds:url>
    <ds:region>RegionOne</ds:region>
  </ingester:credentials>
  <ingester:container>error-container</ingester:container>
</ingester:errorManager>
```

If you want to copy products in a S3 bucket:

```
<ingester:errorManager type="s3">
 <ingester:s3Credentials name="s3error">
    <ds:accessKey>***</ds:accessKey>
    <ds:secretKey>***</ds:secretKey>
    <ds:region>gra</ds:region>
    <ds:endpoint>https://s3.gra.io.cloud.ovh.net</ds:endpoint>
 </ingester:s3Credentials>
 <ingester:bucket>s3-error</ingester:bucket>
</ingester:errorManager>
```

If you want to send message for products in Kafka again:

```
<ingester:errorManager type="kafka">
 <ingester:kafkaHosts>localhost:9092,localhost:9093</ingester:kafkaHosts>
 <ingester:kafkaTopic>error-topic</ingester:kafkaTopic>
 <ingester:kafkaUser>optional_username</ingester:kafkaUser>
 <ingester:kafkaPassword>optional_password</ingester:kafkaPassword>
</ingester:errorManager>
```

This error manager is a good way to retry products that cannot be downloaded or even copied to an error location, for example due to network issues. The kafka topic can be a dedicated one used only for products in error, or even the same one that is being consumed by the consumer.

8.3.3 Datastore Configuration via XML File

We can define some permissions for datastores:

- READ: allows the application to read information from the datastore
- WRITE: allows the application to write into the datastore
- DELETE: allows the application to delete information from the datastore

8.3.3.1 HFSDataStore

Store products in a folder. The internal structure of the folder is (with a depth of 2 and a granularity of 2):

```
rootPath/
    └── 0a/
        └── d5/
            └── 0ad5c26b-2ced-4ea3-96ff-3d4599eee380/
                ├── product_file
                └── ~checksum~
```

Configuration:

```
<ds:dataStore xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:type="ds:hfsDataStoreConf" name="hfs">
   <ds:permission>READ</ds:permission>
```

```
   <ds:permission>WRITE</ds:permission>
   <ds:permission>DELETE</ds:permission>
   <ds:properties>
      <!-- [M] How much time in seconds a product will be kept in this store -->
      <ds:property>
         <ds:name>KEEP_PERIOD_SECONDS</ds:name>
         <ds:value>3000</ds:value>
      </ds:property>
      <!-- [M] Assigned priority. The lower the value is, the highest is the priority -->
      <ds:property>
         <ds:name>PRIORITY</ds:name>
         <ds:value>0</ds:value>
      </ds:property>
   </ds:properties>

   <!-- [M] Abolute path to the destination folder -->
   <!-- In case of using Docker, this path is referred to an internal container path -->
   <!-- which will be mapped to an external absolute path folder of VM host into Docker run
command -->
   <!-- as follows where products will be stored: -v
/path/to/external/folder:/ingest/folder -->
   <!-- In case of using .zip package, this path is referred to an absolute path folder of
VM host where -->
   <!-- products will be stored. -->
   <ds:path>/ingest/folder</ds:path>
   <!-- [0] How many levels of directories there will be. Default is 2 -->
   <ds:depth>0</ds:depth>
   <!-- [0] How many characters of an UUID will be taken for a directory's name. Default is
2 -->
   <ds:granularity>2</ds:granularity>
</ds:dataStore>
```

8.3.3.2 SwiftDataStore

Store products in an OpenStack Swift container. The container will be automatically created. Some swift behaviour can be optionally configured.

IMPORTANT
Do not use underscore character (_) for container's name. Use "-" instead.

Configuration:

```
<!-- [0] Configuration tweaks for Swift behavior -->
<!-- [0] segmentSizeMB: size in MB of segments. Default is 500MB -->
<!-- [0] retries: number of retries done for every swift command. Default is 5 -->
<!-- [0] retryDelayMs: delay before a command is retried. Default is 50 ms -->
```

```
<!-- [0] maxConnections: maximum simultaneous connections opened to the swift storage.
Default is 20 -->

   <ds:swiftConfiguration segmentSizeMB="5000" retries="5" retryDelayMs="50"
maxConnections="20" />
   <!-- Swift credentials -->
   <ds:swiftCredentials name="SwiftCredentials">
      <!-- [M] Tenant name if using keystone V2 or Project name is using keystone V3 ->
         <ds:tenant>tenant</ds:tenant>
         <!-- [M] User password -->
      <ds:password>password</ds:password>
      <!-- [M] User name -->
      <ds:user>username</ds:user>
      <!-- [M] Connection URL -->
      <ds:url>https://auth.cloud.ovh.net/v3</ds:url>
      <!-- [M] Region -->
      <ds:region>region</ds:region>
   </ds:swiftCredentials>

   <!-- [0] This store is a swift container that is used to store quicklooks -->
   <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:swiftDataStoreConf" name="QL">
      <!-- [0] Access permissions. Default is read-only -->
      <ds:permission>WRITE</ds:permission>
      <ds:permission>READ</ds:permission>
      <ds:permission>DELETE</ds:permission>

      <ds:properties>
         <!-- [0] if true, this store will be able to store attached files. Default is
false -->
         <ds:property>
            <ds:name>STORE_ATTACHED_FILES</ds:name>
            <ds:value>true</ds:value>
         </ds:property>
         <!-- [0] if true, products will be stored by their name. Default is false (stored
by uuid) -->
         <ds:property>
            <ds:name>STORE_BY_NAME</ds:name>
            <ds:value>true</ds:value>
         </ds:property>
      </ds:properties>

      <!-- [M] credentials to access the container (defined before datastores) -->
      <ds:credentials>SwiftCredentials</ds:credentials>
      <!-- [M] name of the container -->
      <ds:container>quicklook-container-name</ds:container>
      <!-- [0] If defined, products will be stored with a prefix in containers -->
      <!-- the prefix is extracted from the product name, and can be combination of -->
      <!-- mission, instrument, productType, sensing date (year, month, day) -->
      <ds:prefixLocation>mission/instrument/productType/year/month/day</ds:prefixLocation>
</ds:dataStore>
```

8.3.3.3 SwiftDataStoreGroup

You can organize multiple containers in a group, a so-called datastore group.

A SwiftDataStoreGroup stores products in OpenStack Swift containers. The containers are automatically created according to a pattern. For example, a pattern like "S1.*:CDH-S1,S2.*:CDH-S2,S3.*:CDH-S3,S5.*:CDH-S5P" will implies the creation of 4 containers, one for each type of product.

Like the SwiftDataStore, swift behavior can be tweaked and credentials have to be defined.

IMPORTANT
Do not use underscore character (_) for container names. Use "-" instead.

`<filter>` field is used to define the pattern for product's name. Then, products with the matching name will be stored in this datastore.

Configuration:

```
<!-- [0] First swift group that store products as packages -->
<ds:dataStore xsi:type="ds:swiftDataStoreGroupConf" name="swiftPackage">
    <!-- [0] Access permissions. Default is read-only -->
    <ds:permission>WRITE</ds:permission>
    <ds:permission>READ</ds:permission>
    <ds:permission>DELETE</ds:permission>
    <!-- [0] Different properties managing the behavior of the store -->

    <ds:properties>
      <!-- [0] if true, products will be stored by their name. Default is false (stored by
uuid) -->
      <ds:property>
        <ds:name>STORE_BY_NAME</ds:name>
        <ds:value>true</ds:value>
      </ds:property>
      <!-- [0] if true, products will be stored as multiparts. Default is false (stored as
package) -->
      <ds:property>
        <ds:name>SAVE_AS_MULTIPART</ds:name>
        <ds:value>false</ds:value>
      </ds:property>
      <!-- [M] How much time in seconds a product will be kept in this store -->
      <ds:property>
        <ds:name>KEEP_PERIOD_SECONDS</ds:name>
        <ds:value>3000</ds:value>
      </ds:property>
    </ds:properties>

    <!-- [M] credentials to access the containers (defined before datastores) -->
    <ds:credentials>SwiftCredentials</ds:credentials>
```

```
    <!-- [M] This is a regex product filename filtering on kafka messages received. -->
    <!-- .* means that all messages will be processed by consumer without any filter. -->
    <ds:filter>.*</ds:filter>
    <!-- [M] the pattern used to identify/create containers in this group -->
    <!-- patternMapper is a comma separated list of key:value pairs -->
    <!-- the keys are pattern on product names and the values are the name of the
containers -->
    <!-- pattern are analyzed in declared order, first one matching wins -->
    <ds:containerPattern xsi:type="ds:mapperContainerPatternConf">
      <ds:patternMapper>S1.*:S1-container,S2.*:S2-container,S3.*:S3-container,S5.*:S5P-
container</ds:patternMapper>
    </ds:containerPattern>

    <!-- [0] If defined, products will be stored with a prefix in containers -->
    <!-- the prefix is extracted from the product name, and can be combination of -->
    <!-- mission, instrument, productType, sensing date (year, month, day) -->
    <ds:prefixLocation>mission/instrument/productType/year/month/day</ds:prefixLocation>

</ds:dataStore>
```

8.3.3.4 S3 DataStore

Store products in a S3 bucket. The bucket will be automatically created. Some s3 behaviour can be optionally configured.

```
<ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:s3DataStoreConf" name="s3QL">
   <!-- [0] Access permissions. Default is read-only -->
   <ds:permission>WRITE</ds:permission>
   <ds:permission>READ</ds:permission>
   <ds:permission>DELETE</ds:permission>

   <ds:properties>
      <!-- [0] if true, this store will be able to store attached files. Default is false
-->
      <ds:property>
         <ds:name>STORE_ATTACHED_FILES</ds:name>
         <ds:value>true</ds:value>
      </ds:property>

      <!-- [0] if true, products will be stored by their name. Default is false (stored by
uuid) -->
      <ds:property>
         <ds:name>STORE_BY_NAME</ds:name>
         <ds:value>true</ds:value>
      </ds:property>
   </ds:properties>

      <!-- [0] if true, products will have an alternate asset exposed in a STAC API.
Default is false -->
      <ds:property>
```

```
         <ds:name>EXPOSE_IN_STAC</ds:name>
         <ds:value>true</ds:value>
      </ds:property>
   </ds:properties>


   <!-- [M] credentials to access the bucket (defined before datastores) -->
   <ds:credentials>credentialsS3</ds:credentials>
   <!-- [M] name of the bucket -->
   <ds:bucket>quicklooks</ds:bucket>

   <!-- [0] If defined, products will be stored with a prefix in buckets -->
   <!-- the prefix is extracted from the product name, and can be combination of -->
   <!-- mission, instrument, productType, sensing date (year, month, day) -->
   <ds:prefixLocation>mission/instrument/productType/year/month/day</ds:prefixLocation>

</ds:dataStore>
```

8.3.3.5 S3 DataStoreGroup

S3DataStoreGroup stores products in S3 buckets. The buckets are automatically created according to a pattern. For example, a pattern like "S1.*:CDH-S1,S2.*:CDH-S2,S3.*:CDH-S3,S5.*:CDH-S5P" will implies the creation of 4 buckets, one for each type of product.

Like the S3DataStore, S3 behavior can be tweaked, and credentials must be defined.

```
<ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:s3DataStoreGroupConf" name="s3Group">

   <!-- [0] Access permissions. Default is read-only -->
 <ds:permission>WRITE</ds:permission>
   <ds:permission>READ</ds:permission>
   <ds:permission>DELETE</ds:permission>

   <!-- [0] Different properties managing the behavior of the store -->
   <ds:properties>
      <!-- [0] if true, products will be stored by their name. Default is false (stored by
uuid) -->
      <ds:property>
         <ds:name>STORE_BY_NAME</ds:name>
         <ds:value>true</ds:value>
      </ds:property>

      <!-- [0] if true, products will have an alternate asset exposed in a STAC API.
Default is false -->
      <ds:property>
         <ds:name>EXPOSE_IN_STAC</ds:name>
```

```
         <ds:value>true</ds:value>
      </ds:property>
   </ds:properties>

      <!-- [0] if true, products will be stored as multiparts. Default is false (stored as
package)-->
<ds:property>
         <ds:name>SAVE_AS_MULTIPART</ds:name>
         <ds:value>false</ds:value>
      </ds:property>
   </ds:properties>

   <!-- [M] credentials to access the buckets (defined before datastores) -->
   <ds:credentials>credentialsS3</ds:credentials>
   <!-- [M] filter which products based on their name are accepted. ".*" means all
products -->
   <ds:filter>.*</ds:filter>
   <!-- [M] the pattern used to identify/create buckets in this group -->
   <!-- patternMapper is a comma separated list of key:value pairs -->
   <!-- the keys are pattern on product names and the values are the name of the buckets -
->
   <!-- pattern are analyzed in declared order, first one matching wins -->
   <ds:containerPattern xsi:type="ds:mapperContainerPatternConf">
   <ds:patternMapper>S1.*:cdh-sentinel-1,S2.*:cdh-sentinel-2,S3.*:cdh-sentinel-3,S5.*:cdh-
sentinel-5P,.*:cdh-other</ds:patternMapper>

  </ds:containerPattern>

   <!-- [0] If defined, products will be stored with a prefix in buckets -->
   <!-- the prefix is extracted from the product name, and can be combination of -->
   <!-- mission, instrument, productType, sensing date (year, month, day) -->
   <ds:prefixLocation>mission/instrument/productType/year/month/day</ds:prefixLocation>
</ds:dataStore>
```

8.3.3.6 TimeBasedDataStoreGroup

A group of DataStore with a data circulation policy based on how much time a product can stay in a store. New products will be inserted in the highest priority store. If the last store in the hierarchy can delete products, then products will be deleted if the keep-period is reached on this store.

`<policy>` is used to configure the order to process the different stores. With value `UserDefinedPriorityPolicy`, the operator defines that he will set manually the priority for each store.

The transfer/eviction process can be launched from a catalogue instance.

Configuration:

```
<!-- Time based datastoreGroup -->
<ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:timeBasedDataStoreGroupConf" name="TimeGroup">
   <!-- [0] Access permissions. Default is read-only -->
   <ds:permission>READ</ds:permission>
   <ds:permission>WRITE</ds:permission>
   <ds:permission>DELETE</ds:permission>

   <!-- Define the properties -->
   <ds:properties>
     <!-- EVICT_REFERENCE: if true, when an eviction occurs, the product will also be
evicted from MetaStores -->
     <ds:property>
        <ds:name>STORE_BY_NAME</ds:name>
        <ds:value>true</ds:value>
     </ds:property>
     <ds:property>
        <ds:name>SAVE_AS_MULTIPART</ds:name>
        <ds:value>false</ds:value>
     </ds:property>
     <ds:property>
        <ds:name>KEEP_PERIOD_SECONDS</ds:name>
        <ds:value>432000</ds:value>
     </ds:property>
  </ds:properties>

  <!-- [M] Regex used to filter products that can be added based on their name. To accept
all products: .* -->
  <ds:filter>.*</ds:filter>
  <!-- [M] Control the orders of stores in this group. Here each store has a manually
assigned priority -->
  <ds:policy>UserDefinedPriorityPolicy</ds:policy>

  <!-- [M] DataStores that make up this group. -->
  <ds:dataStores>
     <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfsQL_DAS_S1">
        <ds:permission>READ</ds:permission>
        <ds:permission>WRITE</ds:permission>
        <ds:permission>DELETE</ds:permission>
        <ds:properties>
         <!-- EVICT_ATTACHED_FILES: If an eviction occurs, evict all attached files
(quicklooks) if set to true. Default is false -->
           <ds:property>
              <ds:name>STORE_ATTACHED_FILES</ds:name>
              <ds:value>true</ds:value>
           </ds:property>
```

```
            <!-- [M] Assigned priority. The lower the value is, the highest is the priority
-->
            <ds:property>
                <ds:name>PRIORITY</ds:name>
                <ds:value>0</ds:value>
            </ds:property>
        </ds:properties>

        <ds:path>/ingest/folder1</ds:path>
        <!-- [0] How many levels of directories there will be. Default is 2 -->
        <ds:depth>0</ds:depth>
        <!-- [0] How many characters of an UUID will be taken for a directory's name.
Default is 2 -->
        <ds:granularity>2</ds:granularity>
     </ds:dataStore>

     <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfsQL_DAS_S2">
        <ds:permission>READ</ds:permission>
        <ds:permission>WRITE</ds:permission>
        <ds:permission>DELETE</ds:permission>
        <ds:properties>
        <!-- EVICT_REFERENCE: if true, when an eviction occurs, the product will also be
evicted from MetaStores -->
            <ds:property>
               <ds:name>STORE_ATTACHED_FILES</ds:name>
               <ds:value>true</ds:value>
            </ds:property>

            <!-- [M] Assigned priority. The lower the value is, the highest is the priority
-->
            <ds:property>
               <ds:name>PRIORITY</ds:name>
               <ds:value>0</ds:value>
            </ds:property>
        </ds:properties>
        <ds:path>/ingest/folder2</ds:path>
        <!-- [0] How many levels of directories there will be. Default is 2 -->
        <ds:depth>0</ds:depth>
        <!-- [0] How many characters of an UUID will be taken for a directory's name.
Default is 2 -->
        <ds:granularity>2</ds:granularity>
        </ds:dataStore>

        <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfsQL_DAS_S3">
            <ds:permission>READ</ds:permission>
            <ds:permission>WRITE</ds:permission>
            <ds:permission>DELETE</ds:permission>
            <ds:properties>
               <!-- EVICT_REFERENCE: if true, when an eviction occurs, the product will also
be evicted from MetaStores -->
              <ds:property>
```

```
                <ds:name>STORE_ATTACHED_FILES</ds:name>
                <ds:value>true</ds:value>
             </ds:property>

             <!-- [M] Assigned priority. The lower the value is, the highest is the
priority -->
             <ds:property>
                <ds:name>PRIORITY</ds:name>
                <ds:value>0</ds:value>
             </ds:property>
          </ds:properties>
          <ds:path>/ingest/folder3</ds:path>
          <!-- [0] How many levels of directories there will be. Default is 2 -->
          <ds:depth>0</ds:depth>
          <!-- [0] How many characters of an UUID will be taken for a directory's name.
Default is 2 -->
          <ds:granularity>2</ds:granularity>
       </ds:dataStore>

       <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfsQL_DAS_S5P">
          <ds:permission>READ</ds:permission>
          <ds:permission>WRITE</ds:permission>
          <ds:permission>DELETE</ds:permission>
          <ds:properties>
             <!-- EVICT_REFERENCE: if true, when an eviction occurs, the product will also
be evicted from MetaStores -->
             <ds:property>
                <ds:name>STORE_ATTACHED_FILES</ds:name>
                <ds:value>true</ds:value>
             </ds:property>

             <!-- [M] Assigned priority. The lower the value is, the highest is the
priority -->
             <ds:property>
                <ds:name>PRIORITY</ds:name>
                <ds:value>0</ds:value>
             </ds:property>
          </ds:properties>
          <ds:path>/ingest/folder5P</ds:path>
          <!-- [0] How many levels of directories there will be. Default is 2 -->
          <ds:depth>0</ds:depth>
          <!-- [0] How many characters of an UUID will be taken for a directory's name.
Default is 2 -->
          <ds:granularity>2</ds:granularity>
       </ds:dataStore>
    </ds:dataStores>
</ds:dataStore>
```

8.3.4 DataStore Options

The behavior of DataStores can be customized with different properties.

8.3.4.1 Store Product by Name

Property STORE_BY_NAME, default value false. If **STORE_BY_NAME** is true, product will be stored as FILENAME.zip (e.g. S2B_MSIL2A_20230104T071309_N0509_R106_T39SXT_20230104T084623.zip) instead of default value uuid.zip (e.g. 5a13997b-a360-40e0-a5a7-54629013ce54.zip)

```
<ds:property>
   <ds:name>STORE_BY_NAME</ds:name>
   <ds:value>true</ds:value>
</ds:property>
```

NOTE: This property has no effect on a HFSDataStore. Indeed, this kind of DataStore stores products in a specific directory structure by using product uuids.

8.3.4.2 Store Multipart Product

The next properties are only available for Swift and S3 DataStores.

- Property **SAVE_AS_MULTIPART,** default value false. Indicates if this store will store product as multipart. If STORE_AS_MULTIPART is true, product will be stored as inspectable folder instead of zipped file.

```
<ds:property>
   <ds:name>SAVE_AS_MULTIPART</ds:name>
   <ds:value>true</ds:value>
</ds:property>
```

- Property **MULTIPART_THREADS**, default value 4. Controls how many threads to use to upload parts of one product.

```
<ds:property>
   <ds:name>MULTIPART_THREADS</ds:name>
   <ds:value>4</ds:value>
</ds:property>
```

- Property **MULTIPART_CONTAINER_SUFFIX**. Customize the container that will store parts. Default is no value, i.e. same container.

```
<ds:property>
   <ds:name>MULTIPART_CONTAINER_SUFFIX</ds:name>
   <ds:value>-parts</ds:value>
</ds:property>
```

Products Stored as Multiparts in Swift/S3

A product stored unzipped in swift is composed of multiple subparts and on JSON manifest referencing those parts. For example, a product stored as zip and as multipart will look like this in a container/bucket named **Sentinel-1**:

```
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/...-20210322T145217.pdf
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/calibration/calibration...
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/calibration/calibration...
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/libration/noise...
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/calibration/noise...
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/s1b-s6-slc-vh-...
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/s1b-s6-slc-vv-...
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/manifest.safe
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/measurement/s1b-s6-slc-vh-...
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/measurement/s1b-s6-slc-vv-...
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/preview/icons/logo.png
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/preview/map-overlay.kml
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/preview/product-preview.html
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/preview/quick-look.png
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-calibration.xsd
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-measurement.xsd
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-noise.xsd
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-product.xsd
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-quicklook.xsd
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-map-overlay.xsd
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-object-types.xsd
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-product-preview.xsd
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.zip
S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.zip.json
```

- S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.zip: is the zip version of the product.
- S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.zip.json: is the json manifest.
- other files are the subparts of the product.

The manifest content is:

```
{
  "root": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/",
  "parts": {
    "06cfed05-0862-3f9e-ae91-b1e72e9c44c0": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/preview/quick-look.png",
    "40a40fe5-65eb-3ef4-a238-d7b70f801d34": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-quicklook.xs
    "6095ab03-be62-3d6c-a022-3c0ccd5c214a": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-calibration.
    "40416f42-1b20-334c-a8e0-b81c85504fdd": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-object-types.xsd",
    "506b6acd-54dc-3fdc-a436-c9786c33464c": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/measurement/s1b-s6-slc-vv-20210
    "0d01909e-dd25-31a4-a4d7-99365d261071": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/manifest.safe",
    "4caeeb37-202d-35f9-b108-360fc1d2b1e0": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/calibration/calibrat
    "b64e61bb-ab13-31ae-adba-f0c136f7839a": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/preview/product-preview.html",
    "770b418f-ae2d-3c9c-a282-3544e62d0476": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/S1B_S6_SLC__1SDV_20210322T14200
    "14cda01b-d23d-3264-be8e-29de90085062": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/calibration/noise-s1
    "d7c61a43-a6d1-395d-bd94-c8f1ba505145": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-product-preview.xsd"
    "e2c9c644-fb18-3d6f-9694-08f0ebbe02c7": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/s1b-s6-slc-vv-202103
    "82a1d515-b038-3f7a-a8f8-cfc4e13b30e3": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/calibration/calibrat
    "d0dae257-636d-37b8-a97d-32308168512c": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/annotation/s1b-s6-slc-vh-202103
    "7e9ee09a-d18d-3006-a8fa-5a9aa97c0bbf": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-noise.xsd",
    "e420f2e2-cffd-318f-8c54-4b40d440ea26": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/preview/map-overlay.kml",
    "041cd13c-7206-3e96-880c-58170f480f48": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-map-overlay.xsd",
    "5c91633a-5407-3a53-b155-fc2fffda3962": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/preview/icons/logo.png",
    "2063b273-82bf-316d-b30a-5c27494cbe0e": "S1B_S6_SLC__1SDV_20210322T142000_20210322T142024_026128_031E2D_3A93.SAFE/support/s1-level-1-product.xsd"
  },
  "partsContainer": "Sentinel-1",
  "prefix": "S1B"
}
```


- **root:** root directory of the product
- **parts:** all sub parts of the product
- **partsContainer:** name of the container/bucket containing the parts of the product, can be the same as the one containing the JSON manifest
- **prefix:** prefix location if the product is stored with a prefix in the container

8.3.4.3 Store Attached Files (quicklook)

Property **STORE_ATTACHED_FILES**, default value *false*. Indicates if this store can store attached files.

IMPORTANT
For Collaborative scenario, this property must be set to *true*.

```
<ds:property>
   <ds:name>STORE_ATTACHED_FILES</ds:name>
   <ds:value>true</ds:value>
</ds:property>
```

8.3.4.4 Expose in STAC

Only avaible for S3DataStore and S3DataStoreGroup. If set to true, and if the datastore is used by cdh-stac-api, products stored in this store will expose an alternate asset for downloading them. It will be a link that will redirect to the S3 storage.

```
<ds:property>
   <ds:name>EXPOSE_IN_STAC</ds:name>
   <ds:value>true</ds:value>
</ds:property>
```

8.3.5 Eviction/Transfer Scheduling

The following property is only available for DataStores contained in a TimeBasedDataStoreGroup.

```
<!-- [M] How much time in seconds a product will be kept in this store -->
<ds:property>
   <ds:name>KEEP_PERIOD_SECONDS</ds:name>
   <ds:value>86400</ds:value>
</ds:property>
```

8.3.6 Ingestion Workflow

The ingestion workflow can be customized in the consumer. Each task in the workflow has at least 4 common parameters with default values:

- **pattern:** only applies this task for products (name) matching this pattern. Default is .* for all products. This pattern is defined for each tasks to configure which products will be processed.
- **tryLimit:** number of tries done for this task if an error occurs. Default is 1.
- **stopOnFailure:** if an error occurs during this task, stop the workflow and consider the ingestion as failed. Default is true.
- **active:** whether or not to execute this task. Default is true.

Tasks can be declared in any order. Their priority is programmatically set and cannot be changed. Any number of the same tasks can be added.

8.3.6.1 Available Tasks

- **ingestInDataStores**: This task is used to ingest a product in specific DataStores or in all configured DataStores.

```
<ingester:task
  xsi:type="ingester:ingestInDataStores"
  pattern=".*" tryLimit="1" stopOnFailure="true" targetStores="Swift"/>
```

If the parameter **targetStores** is omitted, the product will be ingested in all configured DataStores.

- **extractMetadata**: This task is used to extract or retrieve (like a DHuS synchronisation without copy. Need to configure *fetchAttributes* to *false* in producer) product's metadata.

```
<ingester:task
  xsi:type="ingester:extractMetadata"
  pattern=".*" tryLimit="1" stopOnFailure="true"/>
```

- **ingestInMetadataStores**: This task is used to ingest a product in specific MetadataStores or in all configured MetadataStores.

```
<ingester:task
  xsi:type="ingester:ingestInMetadataStores"
  pattern=".*" tryLimit="1" stopOnFailure="true" targetStores="Solr"/>
```

If the parameter **targetStores** is omitted, the product will be ingested in all configured MetadataStores.

- **createQuicklook**: This task is used to create and save a product quicklook.

```
<ingester:task
   xsi:type="ingester:createQuicklook"
   pattern=".*" tryLimit="1" stopOnFailure="false" height="512" width="512"
targetStores="QL"/>
```

The parameter **targetStores** is mandatory and should point to store(s) able to store attached files. If the product has no quicklook, it will not be considered as an error.

Description of parameters can be found in the sample configuration files.

To be able to perform some ingestion steps, products must be copied to a local directory (only one read will be done from the source). This directory has to be specified with the **tmpPath** attribute of the **ingester:tasks** element. Default value is */tmp* directory.

IMPORTANT
If products are ingested in a folder, make sure to attach it as volume if you use the Docker distribution.

Add `-v /path/to/external/folder:/ingest/folder` to the Docker image.

To summarize the filters / patterns

The ingester has:

- A **filter for producer** to define the product which name matches the pattern it will check after scanning the source. To process all the products, use ".*". (For all type of producers). For example, if only want to process ZIP files in a S3 producer, use ".*.zip"
- An **OData filter** only for OData source (on Producer) to apply on OData source (CSC or DHuS)
- A **STAC filter** for STAC source in producer. This filter is a combination of key/value pairs, supporting any query parameter available in the source.
- A **geographical filter** only for OData source (on Producer) to apply on an OData source that doesn't support geographic filters (like DHuS) after OData filter. Note that this filter will be used after fetching products matching the OData filter. So only use it if the source doesn't natively support a geographic filter in its OData filter.
- A **pattern** defined **for each task of the consumer**. The products matching the pattern are processed by the given task.

STAC filter example for producer

For xml configuration:

```
<ingester:filter>
   <ingester:param name="collections" value="SENTINEL-2" />
   <ingester:param name="bbox" value="-80.673805,-0.52849,-78.060341,1.689651" />
</ingester:filter>
<ingester:sortby>+datetime</ingester:sortby>
```

For JSON configuration:

```
"filter": {
  "param": [
    {
      "name": "collections",
      "value": "SENTINEL-2"
```

```
    },
    {
    "name": "bbox",
    "value": "-80.673805,-0.52849,-78.060341,1.689651"
    }
  ]
},
"sortby": "+datetime"
```

The available filter parameters are the ones available in any STAC API:
- bbox
- intersects (cannot be used if bbox is present)
- collections
- ids
- datetime

9. Catalogue

Catalogue is a CSC OData REST API exposing earth observation products and allowing them to be downloaded. Products data can be stored in different stores: HFS, Swift and S3. Products metadata will be saved inside a Solr Cloud instance by Ingest.

Storage configuration is done in a file named **gss.xml**. This application supports authentication through a Keycloak server configured in a file named **application.properties**.

The catalogue is by default deployed on the context **/odata/v1** on the port **8080**.

9.1 Pre-requisites

9.1.1 Infrastructure Requirements

We need to install:

- Docker: 20.10.12 (or later)

9.1.2 Network Requirements

Application	Default port
PostgreSQL	5432
SolR	8983
Keycloak version 18 or later	8443

9.1.3 Software Requirements

Some tools must be installed to use the catalogue:

- A running SolrCloud instance, (9.0.0 and after: https://lucene.apache.org/solr/downloads.html) with a collection (use the **toolbox**)
- A running PostgreSQL instance, (10.12 and after: https://www.postgresql.org/download/) and a database initialize with **toolbox**.
- A running Keycloak instance if you want to use authentication (18.0.0 or later: https://www.keycloak.org/downloads)
- A storage depending on your scenario: folder, openstack swift, S3

The Solr schema and the database schema update/creation scripts are inside the **CDH-toolbox** module. The database and the solr should be updated before launching a new version of the catalogue.

9.2 Distribution

This software is available in two formats, a zip archive and a Docker image.

9.2.1 Zip Archive

Available at the following address, the archive must be downloaded and unzipped to launch the catalogue: https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-catalogue/3.4.0/cdh-catalogue-3.4.0.zip

The zip archive is presented as follows:

```
cdh-catalogue/
├── etc/
│   ├── operational-samples/
│   │   ├── application.properties.sample
│   │   ├── docker-compose.yml
│   │   ├── log4j2.xml
│   │   ├── gss_dest-folder.xml
│   │   ├── gss_dest-swift.xml
│   │   └── xml-conf
│   │       ├── application.properties.sample
│   │       ├── docker-compose.yml
│   │       └── gss-catalogue.xml
│   │
```

```
│   ├── JSON-Samples/
│   │   └── Subscription
│   │       ├── subscription_deleted_product.json
│   │       └── subscription_ingested_product.json
│   │
│   ├── gss_sample.xml
│   ├── application.properties.sample
│   └── log4j2.xml
│
├── lib/
├── logs/
├── start.sh
└── stop.sh
```

The gss_sample.xml file is an example of a **gss.xml** configuration. A **gss.xml** file must be present in the *etc/* directory to launch the catalogue, unless you rely on database configuration.

The **application.properties** is used for the Keycloak server configuration and application configuration. This file is self described. An **application.properties** file must be present in the *etc/* directory to launch the catalogue.

The **log4j2.xml** indicates where and how the logs of the application are written. It can be modified at page format.

With ZIP distribution, use the `start.sh` script to launch the catalogue and use `stop.sh` to gracefully stop it.

```
nohup ./start.sh &
./stop.sh
```

9.2.2 Docker Image

Available on Docker Hub -> `docker pull registry.hub.docker.com/gaeldockerhub/cdh-catalogue:3.4.0`

You can check that the Docker image has been downloaded with the command `docker image list | grep cdh-catalogue.`

The Docker image is based on an **openjdk:17-jdk-slim** image. The catalogue application will be launched in a created /catalogue directory. Don't forget to expose the port of the catalogue when launching the Docker container.

To launch the Docker container, you have to provide the **gss.xml** and the **application.properties** files as attached volumes. Examples for those files are available in the Zip archive.

Example:

```
docker run -d --name cdh-catalogue -v /path/gss.xml:/catalogue/etc/gss.xml
-v /path/application.properties:/catalogue/etc/application.properties -it
registry.hub.docker.com/gaeldockerhub/cdh-catalogue:3.4.0
```

The *-d* option is used to run the Docker container in detached mode. If you want your Docker container to point to the localhost of the hosting machine, add the launch option *--network="host"* (https://docs.docker.com/network/host/). It means that the container shares the host machine network, the container will have the same IP and ports as the host. For example, if you launch the container and you want to access the catalogue on localhost:8080, you must use this option.

You can map the log files of the application with a local folder by adding an attached volume:

```
-v /your/path/to/logs:/catalogue/logs.
```

To gracefully stop the container, add a timeout (in seconds) to the stop command:

```
docker stop -t 10 cdh-catalogue
```

9.3 Catalogue Configuration

Configure the **application.properties** according to your needs.

9.3.1 Catalogue Keycloak Configuration

CDH-Catalogue requires Keycloak version 18 or later. To ensure the users access to the Catalogue two Keycloak configuration should be enabled through the Keycloak console page.

- Authentication via a web API.
- Authentication via a web browser.

9.3.1.1 Keycloak Web-API Authentication Configuration

The Web-API authentication uses the OAuth 2.0 protocol to protect resource requests by verifying authorized access tokens. To enable this authentication, it is essential to add user/users to both the Keycloak client-access role and realm-access role.

To do so, the following steps must be taken:

- Access the Keycloak admin console.
- Select the relevant Keycloak realm.

1. **Keycloak Realm-access:**
   - Within your desired realm, navigate to the "Realm" section in the left sidebar.
   - To ensure that the user is listed in roles for realm-access, please follow the appropriate steps:
     - `Admin-console → Realm roles → <your-role> → Users in role → <desired-user>`

2. **Keycloak Client-access:**
   - Within your desired realm, navigate to the "Clients" section in the left sidebar.
   - Here, you'll see a list of clients associated with the current realm. Initially, you might see some default clients like "account" and "realm-management."
   - To ensure that the user is listed in roles for client-access, please follow the appropriate steps:
     - `Admin-console → Clients → <your-client> → roles → <your-role> → Users in role → <desired-user>`

3. **Keycloak Users:**
   - To begin, go to the "Users" section in the left sidebar of your desired realm.
   - You will find a list of users here.
   - It's important to note that there are two types of roles
     - The `realm-access role`.
     - The `client-access role`.

   - Role mapping for the realm-access role:
     - `Admin-console → Users → <desired-user> → Role Mapping → Assign role → Filter by realm roles → Filter by clients → <your-role> → Assign`

   - Role mapping for the client-access role:
     - `Admin-console → Users → <desired-user> → Role Mapping → Assign role → <your-role> → Assign`

Note:

- Please ensure to see the user/users in (Users in role) of realm-access and client-access.
- The settings in the Keycloak console may differ depending on the version used.
- The following website allows to see the users and roles:
    - Generate a token.
    - Decode the token.
    - Verification of the token containing the roles at the https://jwt.io
- Please refer to RD-4 for the complete Keycloak configuration.

User role mapping example in realm and client access below:

```
{
  "realm_access": {
    "roles": [
      "user-role",              // realm-access role
      "default-roles-admin-realm",
      "offline_access",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "user-client": {
      "roles": [
        "user-role"             // client-access role
      ]
    },
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "preferred_username": "cdh-user"
}
```

9.3.1.2 Keycloak Web-browser Authentication Configuration

CDH-Catalogue uses Oauth2 client with Single Sign-On (SSO) as an authentication mechanism to allow user access to the Catalogue via a web browser. SSO on CDH-Catalogue works as follows:

- **Authentication:** When a user attempts to access the Catalogue, the user is redirected to Keycloak's authentication server.
- **Login:** The user enters their credentials (username and password) on Keycloak's authentication login page.
- **Token Generation:** Upon successful authentication, Keycloak authentication server generates a security token that represents the user's identity and contains some relevant information about the user.
- **Token Validation:** The Catalogue then receives the security token for further verification.

Enabling Single Sign-On and accessing the catalogue via a web browser requires that the userInfo checkbox setting be selected in the following way:

- Log in to the Keycloak admin console.
- Choose the Keycloak realm.
- `Navigate to Client Scopes → roles → Mappers → client roles → Add to userinfo (checkbox)`
- `Proceed to Client Scopes → roles → Mappers → realm roles → Add to userinfo (checkbox)`

In Keycloak, userInfo is a pre-defined scope that allows clients to request access to fundamental user information from Keycloak's userinfo during the OAuth 2.0 token exchange process.

9.3.2 Keycloack Authentication

The table below describes the Keycloak roles needed by users to perform operations on the system.

User Role	Description	DHS Actor (admin / user)
end-user	End user with just right to list products, have details on them, download them	

The **application.properties** can be used to enable authentication through a Keycloak server. To use OAuth2 authentication through command line:

- Request a token:

```
curl -d 'client_id=<keycloak_client>' -d 'username=***' -d 'password=***' -d
'grant_type=password'
"https://<keycloak_url>/auth/realms/<keycloak_realm>/protocol/openid-connect/token"
```

You will obtain a JSON response containing the *access_token*

- To use the token in your requests:

```
curl -X GET -H 'authorization: Bearer <access_token>' -H 'content-type: application/json'
"http://<server.url>:<server.port>/odata/v1/Products"
```

It is also important to configure which information will be used to identify users. So, you need to provide the correct value to parameter "***user-name-attribute***" in **application.properties**.

There are 2 possibilities:

- prefered_username: using the prefered user's name configured in Keycloak to identify the user in logs and for quotas management.
- user_id: using the user id defined in Keycloack to identify the user in logs and for quotas management.

IMPORTANT

```
# Which attribute to use to identify the user (displayed in logs and used for quotas)

# Possible values are: user_id, preferred_username. Default value is user_id.

user-name-attribute=preferred_username
```

9.3.3 Configuration via XML file

For configuration, you can refer to §8.3.3

9.3.3.1 HFSDataStore

See 8.3.3.1

9.3.3.2 SwiftDataStore

See 8.3.3.2

9.3.3.3 SwiftDataStoreGroup

See 8.3.3.3

9.3.3.4 S3DataStore

See 8.3.3.4

9.3.3.5 S3DataStoreGroup

See 8.3.3.5

9.3.3.6 TimeBasedDataStoreGroup

See 8.3.3.6 for general description.

The transfer/eviction process can be launched according to configuration in Catalogue:

```
<conf:process evictionByTime="true" />
```

In the new release deployment configuration, this parameter is not present in Admin API Timebased datastore JSON and must be set in **application.properties** (*process.evictionByTime*)

Note: The evictionByTime parameter should be set to true either in the **application.properties** file of the Admin API, or in the **application.properties** file of the Catalogue, but not in both components at the same time.

9.3.4 DataStore Options

See §8.3.4 for details

9.4 Eviction

The two following properties are only available for **TimeBasedDataStoreGroup** and control eviction behaviour.

9.4.1 Evict Metadata

If an eviction occurs, evict the product from all **MetadataStore** if set to *true*. Default is *false*.

IMPORTANT
For Collaborative scenario, this property must be set to *true*.

```
<ds:property>
   <ds:name>EVICT_REFERENCE</ds:name>
   <ds:value>true</ds:value>
</ds:property>
```

9.4.2 Evict Attached Files (quicklooks)

If an eviction occurs, evict all **attached files** (quicklooks) if set to *true*. Default is *false*.

IMPORTANT
For Collaborative scenario, this property must be set to *true*.

```
<ds:property>
   <ds:name>EVICT_ATTACHED_FILES</ds:name>
   <ds:value>true</ds:value>
</ds:property>
```

9.4.3 Retention Policy – only for TimeBasedDataStoreGroup

The following property is only available for DataStores contained in a **TimeBasedDataStoreGroup**.

IMPORTANT
It's recommended to use the same value as the one defined for ingestion.

```
<!-- [M] How much time in seconds a product will be kept in this store -->
<ds:property>
   <ds:name>KEEP_PERIOD_SECONDS</ds:name>
   <ds:value>86400</ds:value>
</ds:property>
```

9.5 Catalogue Processes/Functions Activation

Catalogue background processes and functions can be turned on or off in the `<configuration>` attribute.

EvictionByTime is used to activate product eviction. Only one cdh-catalogue or one cdh-admin-api should have it activated.

ProductNotification is used to send notification messages to Kafka when a product is evicted.

DirectDownloadLink is used to enable/disable a possible download from a swift/S3 storage directly, when using a cloud storage. If it is set to *false*, the download will be perform through catalogue.

QuotaDisabled allows the system to enable/disabled all user's quota Management. So, if it is set to *false*, no quota is used for downloads.

```
<!-- [0] evictionByTime activate the data transfer for TimeBasedDataStoreGroups. Default
is false -->
<conf:process evictionByTime="false" />

<!-- [0] Activate notifications for evicted products. Default is false -->
<conf:productNotification enabled="false" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="notification:kafkaProductNotificationConf">
   <!-- [M] comma separated kafka nodes (1 to n nodes) -->
   <notification:hosts>localhost:29092</notification:hosts>
   <!-- [0] Kafka username -->
   <notification:user>***</notification:user>
   <!-- [0] Kafka password -->
   <notification:password>***</notification:password>
   <!-- [0] Topic name where messages for evicted products are sent. Default is
notification_evicted -->
   <notification:evictionTopic>notification_evicted</notification:evictionTopic>
</conf:productNotification>

<!-- [0] Configuration tweaks -->
<!-- directDownloadLink: enable the direct download from cloud stores (swift/S3) when
using $value. Default is true. -->
<!-- quotaDisabled: disable quotas. Default is false. -->
<conf:download directDownloadLink="true" quotaDisabled="false" />
```

9.6 Default quotas

If no quotas are assigned to a user, default ones will be used. These default values can be customized.

If using the gss.xml file:

```
<conf:download directDownloadLink="false" quotaDisabled="false">
   <dl:quota name="TOTAL_DOWNLOAD" value="300000000" durationMinutes="60" />
   <dl:quota name="PARALLEL_DOWNLOAD" value="2" />
   <dl:quota name="TOTAL_DOWNLOAD_LINK" value="10" durationMinutes="10" />
</conf:download>
```

If using the db configuration, add in application.properties:

```
##custom default quotas values if no quotas assigned to a user
#number of concurrent download, default = 2
download.quota.parallel.download.value = 2
```

```
#max volume of download in bytes/min, default is 3GB/hour
download.quota.total.download.value = 3000000000
download.quota.total.download.duration = 60
#max number of download links in numbers/min, default is 10/10 min
download.quota.total.download.link.value = 10
download.quota.total.download.link.duration = 1
```

Note that if authentication is disabled and quotas are enabled, a default user named JohnDoe will be used in all requests. This fake user comes with specific quota values configured in database. You can edit them or remove them if you want to use default quotas as above.
These quotas are defined in the table *quota* with *user_id='JohnDoe'*.

10. Catalogue endpoints

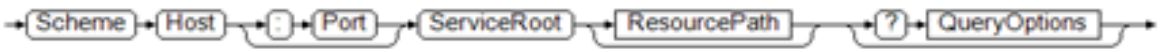
Here are some URLs for accessing catalogue functionalities.

10.1 Example of OData Service Root URL

Example of OData service Root:

https://cdh.catalogue.gael.fr:8080/odata/v1

For more details on the URL convention, you can use this link: http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part2-url-conventions.html



Query Options are like following

Option	Description
`$format`	Specifies the HTTP response format e.g. XML or JSON.
`$expand`	Directs that related records should be retrieved in the record or collection being retrieved.
`$filter`	Specifies an expression or function that must evaluate to **true** for a record to be returned in the collection.
`$orderby`	Determines what values are used to order a collection of records.
`$select`	Specifies a subset of properties to return.
`$skip`	Sets the number of records to skip before it retrieves records in a collection.

`$top`	Determines the maximum number of records to return.
`$skiptoken`	Uses server-side paging. First value should be * and then use the new value returned in the response to go to next page.

10.2 OData Product Entity

The following is a representation of a product in JSON format:


```
{
  "@odata.mediaContentType": "application/octet-stream",
  "Id": "6226d126-d71b-48e0-aa24-67764d031af8",
  "Name":
"S3B_SL_2_LST___20230930T222958_20230930T223258_20231001T000059_0179_084_300_4500_PS2_O_NR_004.zip",
  "ContentType": "application/zip",
  "ContentLength": 59192366,
  "OriginDate": "2023-10-01T00:06:43.563Z",
  "PublicationDate": "2023-10-06T13:01:44.590Z",
  "ModificationDate": "2023-10-06T13:01:44.590Z",
  "Online": true,
  "EvictionDate": null,
  "Checksum": [
    {
      "Algorithm": "MD5",
      "Value": "511089e72500a6faa58f28b609345c9b",
      "ChecksumDate": "2023-10-06T13:01:47.071Z"
    }
  ],
  "ContentDate": {
    "Start": "2023-09-30T22:29:58.072536Z",
    "End": "2023-09-30T22:32:58.072536Z"
  },
  "Footprint": "geography'SRID=4326;Polygon((112.6421350418926 -85.05115,110.248 -84.6594,108.14 -84.2486,106.372 -
83.8257,104.801 -83.4092,103.451 -82.98,102.21 -82.557,101.126 -82.1183,100.105 -81.684,99.249 -81.2481,98.4687 -
80.8115,97.6909 -80.3655,97.0353 -79.9222,96.4288 -79.4827,95.8654 -79.0348,95.3801 -78.5945,94.8844 -78.1438,94.4498 -
77.6993,94.0415 -77.2473,93.6366 -76.8024,93.288 -76.3486,92.9299 -75.9028,92.615 -75.4502,92.3461 -75.0031,92.0543 -
74.5448,91.7917 -74.0959,91.523 -73.6504,91.2907 -73.1926,91.053 -72.7461,90.8436 -72.2911,90.8143 -72.2924,81.8875 -
72.4692,72.9566 -72.2402,64.4296 -71.6191,56.7125 -70.6624,56.1462 -71.075,55.4906 -71.4869,54.8938 -71.9011,54.1935 -
72.3062,53.5018 -72.7171,52.7657 -73.1187,51.9951 -73.519,51.2198 -73.9133,50.3698 -74.3127,49.4683 -74.7054,48.5624 -
75.0931,47.5903 -75.4766,46.5609 -75.8533,45.475 -76.2333,44.3322 -76.5951,43.117 -76.9623,41.8506 -77.3186,40.5014 -
77.67,39.1143 -78.0178,37.5852 -78.3574,35.993 -78.6828,34.3295 -79.0064,32.5157 -79.3193,30.6709 -79.6158,28.6505 -
79.9035,26.546 -80.1827,24.3087 -80.4374,21.9757 -80.6818,19.4853 -80.9169,18.5624 -80.9996,28.8487 -83.2007,47.8407 -
85.0356,48.426782712735275 -85.05115,112.6421350418926 -85.05115))'",
  "GeoFootprint": {
    "type": "Polygon",
    "coordinates": [
```

```
        [
          [ 112.6421350418926, -85.05115 ], [ 110.248, -84.6594 ], [ 108.14, -84.2486 ], [ 106.372, -83.8257 ], [ 104.801, -83.4092 ],
[ 103.451, -82.98 ], [ 102.21, -82.557 ], [ 101.126, -82.1183 ], [ 100.105, -81.684 ], [ 99.249, -81.2481 ], [ 98.4687, -80.8115 ], [
97.6909, -80.3655 ], [ 97.0353, -79.9222 ], [ 96.4288, -79.4827 ], [ 95.8654, -79.0348 ], [ 95.3801, -78.5945 ], [ 94.8844, -78.1438 ],
[ 94.4498, -77.6993 ], [ 94.0415, -77.2473 ], [ 93.6366, -76.8024 ], [ 93.288, -76.3486 ], [ 92.9299, -75.9028 ], [ 92.615, -75.4502 ], [
92.3461, -75.0031 ], [ 92.0543, -74.5448 ], [ 91.7917, -74.0959 ], [ 91.523, -73.6504 ], [ 91.2907, -73.1926 ], [ 91.053, -72.7461 ], [
90.8436, -72.2911 ], [ 90.8143, -72.2924 ], [ 81.8875, -72.4692 ], [ 72.9566, -72.2402 ], [ 64.4296, -71.6191 ], [ 56.7125, -70.6624 ],
[ 56.1462, -71.075 ], [ 55.4906, -71.4869 ], [ 54.8938, -71.9011 ], [ 54.1935, -72.3062 ], [ 53.5018, -72.7171 ], [ 52.7657, -73.1187 ],
[ 51.9951, -73.519 ], [ 51.2198, -73.9133 ], [ 50.3698, -74.3127 ], [ 49.4683, -74.7054 ], [ 48.5624, -75.0931 ], [ 47.5903, -75.4766 ],
[ 46.5609, -75.8533 ], [ 45.475, -76.2333 ], [ 44.3322, -76.5951 ], [ 43.117, -76.9623 ], [ 41.8506, -77.3186 ], [ 40.5014, -77.67 ], [
39.1143, -78.0178 ], [ 37.5852, -78.3574 ], [ 35.993, -78.6828 ], [ 34.3295, -79.0064 ], [ 32.5157, -79.3193 ], [ 30.6709, -79.6158 ], [
28.6505, -79.9035 ], [ 26.546, -80.1827 ], [ 24.3087, -80.4374 ], [ 21.9757, -80.6818 ], [ 19.4853, -80.9169 ], [ 18.5624, -80.9996 ], [
28.8487, -83.2007 ], [ 47.8407, -85.0356 ], [ 48.426782712735275, -85.05115 ], [ 112.6421350418926, -85.05115 ]
        ]
    ]
  }
}
```


10.3 Search – OData Queries

You can search for products using OData queries.

10.3.1 List of Products (GET)

- Description: Allows the user to display all the products in the catalogue.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products
- Return: List of products in JSON.

10.3.2 Details of a Product (GET)

- Description: Get details of a product by Id.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)
- Return: A product in JSON, if it exists.

10.3.3 Query with Filter on Properties (GET)

- Description: Query with filter on properties (name and publication date)

- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)?$filter=startswith(Name,'S2') and PublicationDate lt 2023-01-25T09:00:00.000Z
- Return: A list of products in JSON, if it exists and matches the condition.

10.3.4 Query with Filter on Integer Attribute (GET)

- Description: Query with filter on Integer Attribute.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products?$filter=Attributes/OData.CSC.IntegerAttribute/any(att:att/Name eq '<attribute.name>' and att/OData.CSC.IntegerAttribute/Value eq <attribute.value>)
- Return: All the products matching the filter on the Integer property.

10.3.5 Query with Filter on String Attribute (GET)

- Description: Query with filter on String Attribute.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products?$filter=Attributes/OData.CSC.StringAttribute/any(att:att/Name eq '<attribute.name>' and att/OData.CSC.StringAttribute/Value eq '<attribute.value>')
- Return: All the products matching the filter on the String property.

10.3.6 Query with Filter on Date Attribute (GET)

- Description: Query with filter on Date Attribute.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products?$filter=Attributes/OData.CSC.DateTimeOffsetAttribute/any(att:att/Name eq '<attribute.name>' and att/OData.CSC.DateTimeOffsetAttribute/Value lt <attribute.value>)

- The date value is in format: **YYYY-MM-DDTHH:MM:SS.000000Z** (Example: 2022-11-28T12:52:35.000000Z)
- Return: List of products, if it exists.

10.3.7 Query with Filter on a specific year of Date property (GET)

- Description: Query with filter on a specific year of a Date property.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products?$filter=year(ContentDate/Start) eq 2025
- Return: List of products, if it exists.

10.3.8 Geographic Query (GET)

- Description: Geographic query.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products?$filter=OData.CSC.Intersects(area=geography'SRID=<value>;Polygon(<polygon values>)')
- Return: List of products matching the filter, if it exists.

```
http://<server.url>:<server.port>/odata/v1/Products?$filter=OData.CSC.Intersects(location=Footprint,area=geography'SRID=4326;POLYGON((2.590271552251986 48.83797852010584,2.5905505019895347 48.83797852010584,2.5905505019895347 48.83755835202525,2.590271552251986 48.83755835202525,2.590271552251986 48.83797852010584))')
```

An alternative version is:

```
http://<server.url>:<server.port>/odata/v1/Products?$filter=OData.CSC.Intersects(area=geography'SRID=4326;POLYGON((2.590271552251986 48.83797852010584,2.5905505019895347 48.83797852010584,2.5905505019895347 48.83755835202525,2.590271552251986 48.83755835202525,2.590271552251986 48.83797852010584))')
```

Note: the OData.CSC.Intersects function can be combined with the NOT operator to exclude all products intresecting a particular geometry.

10.4 Download

The download is feasible on a product or one of its attached files. You can perform this operation using an OData query.

For Zarr products, their size is set to 0 as they are a directory. Each file making the product will however have the correct size. If a user downloads a Zarr product, it will be zipped.

For products that are unpacked, like a zip file where all entries are stored unzipped, the server will zip the entries on the fly. So, the downloaded size may differ each time the zipped product is downloaded, as some metadata are added into the zip, like dates, as it is being built by the server.

10.4.1 Download a Product (GET)

- Description: Download a product by Id.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)/$value

10.4.2 Download the Product by Range (GET)

- Description: Download the product by range.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)/$value
- Header: **Range** bytes=0-1023
- Result: Download part of the product.
- It will not work for product stored as unpacked, like ZARR products

10.4.3 Download Attached File or a Product Part (quicklook) (GET)

- Description: Download an attached file or product part (quicklook)
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)/AttachedFiles('quick-look.jpg')/$value
- Result: Download the defined attached file.

10.5 Product Navigation

10.5.1 List Product Node (GET)

- Description: List a produce node.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)/Nodes

10.5.2 Product Node (GET)

- Description: Get a product Node.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)/Nodes('<productNameNodes'>)/Nodes

10.5.3 Product Manifest (GET)

- Description: Get a product manifest.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)/Nodes('<productNameNodes>'/Nodes('<manifestNode>')

10.5.4 Download Manifest (GET)

- Description: Download a product manifest.
- HTTP Method: GET
- URL: http://<server.url>:<server.port>/odata/v1/Products(<product.uuid>)/Nodes('<productNameNodes>'/Nodes('<manifestNode>')/$value

10.6 Create a Subscription (POST)

Subscriptions can be created for two events:
- SubscriptionEvent: "deleted" -> for deleted and evicted products
- SubscriptionEvent: "created" -> for ingested products

Description: We can create a subscription on the catalogue using the request:

HTTP Method: POST

URL: http://<server.url>:<server.port>/odata/v1/Subscriptions

Body:


```
{
  "Status": "running",
  "SubscriptionEvent": "<deleted|created>",
  "FilterParam": "Products?$filter=contains(Name,'R014')",
  "NotificationEndpoint": "https://typedwebhook.tools/webhook/9edb1881-059e-44c2-9d5f-984061c1db35"
}
```


Response:


```
{
  "@odata.context": "$metadata#Subscriptions",
  "Id": "8d22ad95-fa9f-47fd-b90c-ab2769cb4ec2",
  "Status": "running",
  "SubscriptionEvent": "DELETED",
  "FilterParam": "Products?$filter=contains(Name,'R014')",
  "SubmissionDate": "2023-11-07T11:18:36.926669Z",
  "LastQueryDate": "2023-11-07T11:18:36.926673Z",
  "NotificationEndpoint": "https://typedwebhook.tools/webhook/9edb1881-059e-44c2-9d5f-984061c1db35",
  "NotificationEpUsername": null,
  "NotificationEpPassword": null
}
```


cURL

```
curl -v -X POST 'https://<server.url>:<port>/odata/v1/Subscriptions' -d '{
    "Status": "running",
    "SubscriptionEvent": "deleted",
    "FilterParam": "Products?$filter=startswith(Name,'\''S'\'')",
    "NotificationEndpoint": "https://webhook.site/8c0e7526-3ef1-4fae-b87a-
886e5f911546"
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" |
jq
```

RECOMMENDATION
You can find JSON configuration sample files in the cdh-catalogue-3.4.0.zip file. These files can be used for catalogue subscription configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Subscription**" directory.

10.7 List of Subscriptions (GET)

Description: Get a list of available subscriptions.

HTTP Method: GET

URL: http://<server.url>:<server.port>/odata/v1/Subscriptions

Response:


```
{
  "@odata.context": "$metadata#Subscriptions",
  "value": [
    {
      "Id": "7be6dad7-e21a-49b2-9a00-f12d8bf4bd74",
      "Status": "running",
      "SubscriptionEvent": "DELETED",
      "FilterParam": "Products?$filter=startswith(Name,'S')",
      "SubmissionDate": "2023-11-06T22:04:21.427611Z",
      "LastQueryDate": "2023-11-06T22:04:21.427613Z",
      "NotificationEndpoint": "https://webhook.site/79c43a12-fb76-49ad-8c48-5885f12e8e7f",
      "NotificationEpUsername": null,
      "NotificationEpPassword": null
    },
    {
      "Id": "8d22ad95-fa9f-47fd-b90c-ab2769cb4ec2",
      "Status": "running",
      "SubscriptionEvent": "DELETED",
      "FilterParam": "Products?$filter=contains(Name,'R014')",
      "SubmissionDate": "2023-11-07T11:18:36.926669Z",
      "LastQueryDate": "2023-11-07T11:18:36.926673Z",
      "NotificationEndpoint": "https://typedwebhook.tools/webhook/9edb1881-059e-44c2-9d5f-984061c1db35",
      "NotificationEpUsername": null,
      "NotificationEpPassword": null
    }
  ]
}
```

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/odata/v1/Subscriptions" | jq
```

10.8 Details of a Subscription (GET)

Description: Retrieve details about a given subscription

HTTP Method: GET

URL: http://<server.url>:<server.port>/odata/v1/Subscriptions(<subscription.id>)

Response:


```
{
  "@odata.context": "$metadata#Subscriptions/$entity",
  "Id": "8d22ad95-fa9f-47fd-b90c-ab2769cb4ec2",
  "Status": "running",
  "SubscriptionEvent": "DELETED",
  "FilterParam": "Products?$filter=contains(Name,'R014')",
  "SubmissionDate": "2023-11-07T11:18:36.926669Z",
  "LastQueryDate": "2023-11-07T11:18:36.926673Z",
  "NotificationEndpoint": "https://typedwebhook.tools/webhook/9edb1881-059e-44c2-9d5f-984061c1db35",
  "NotificationEpUsername": null,
  "NotificationEpPassword": null
}
```


cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/odata/v1/Subscriptions(<subscription-id>)" | jq
```

10.9 Cancel a subscription (POST)

Description: Cancel a specific subscription

HTTP Method: POST

URL: http://<catalogue.url>/Subscriptions(<subscription.id>)/OData.CSC.Cancel

Response:


```
{
  "@odata.context": "$metadata#OData.CSC.Subscription",
  "Id": "8d22ad95-fa9f-47fd-b90c-ab2769cb4ec2",
  "Status": "cancelled",
  "SubscriptionEvent": "DELETED",
  "FilterParam": "Products?$filter=contains(Name,'R014')",
  "SubmissionDate": "2023-11-07T11:18:36.926669Z",
  "LastQueryDate": "2023-11-07T11:18:36.926673Z",
  "NotificationEndpoint": "https://typedwebhook.tools/webhook/9edb1881-059e-44c2-9d5f-984061c1db35",
  "NotificationEpUsername": null,
  "NotificationEpPassword": null
}
```


cURL

```
curl -v -X POST
'https://<server.url>:<port>/odata/v1/Subscriptions(<subscription-
id>)/OData.CSC.Cancel' -H 'Content-Type: application/json' -H
"Authorization: Bearer ${AT}" | jq
```

10.10 Pause a Subscription (POST)

Description: Pause a specific subscription so it can change status from "Running" to "Paused"

HTTP Method: POST

URL: http://<catalogue.url>/Subscriptions(<subscription.id>)/OData.CSC.Pause

Response:


```
{
  "@odata.context": "$metadata#OData.CSC.Subscription",
  "Id": "7be6dad7-e21a-49b2-9a00-f12d8bf4bd74",
  "Status": "paused",
  "SubscriptionEvent": "DELETED",
  "FilterParam": "Products?$filter=startswith(Name,'S')",
  "SubmissionDate": "2023-11-06T22:04:21.427611Z",
  "LastQueryDate": "2023-11-06T22:04:21.427613Z",
  "NotificationEndpoint": "https://webhook.site/79c43a12-fb76-49ad-8c48-5885f12e8e7f",
  "NotificationEpUsername": null,
```

```
  "NotificationEpPassword": null
}
```


An error can occurred: **Cannot Pause and Resume a cancelled subscription.**


```
{
  "error": {
    "code": null,
    "message": "Cannot pause/resume Cancelled subscription"
  }
}
```


cURL

```
curl -v -X POST
'https://<server.url>:<port>/odata/v1/Subscriptions(<subscription-
id>)/OData.CSC.Pause' -H 'Content-Type: application/json' -H
"Authorization: Bearer ${AT}" | jq
```

10.11 Resume a Subscription (POST)

Description: Resume a specific subscription so it can change status from "Paused" to "Running".

HTTP Method: POST

NOTE: Notifications are sent only for RUNNING subscriptions.

URL: http://<catalogue.url>/Subscriptions(<subscription.id>)/OData.CSC.Resume

Response:


```
{
  "@odata.context": "$metadata#OData.CSC.Subscription",
  "Id": "7be6dad7-e21a-49b2-9a00-f12d8bf4bd74",
  "Status": "running",
  "SubscriptionEvent": "DELETED",
  "FilterParam": "Products?$filter=startswith(Name,'S')",
  "SubmissionDate": "2023-11-06T22:04:21.427611Z",
  "LastQueryDate": "2023-11-06T22:04:21.427613Z",
  "NotificationEndpoint": "https://webhook.site/79c43a12-fb76-49ad-8c48-5885f12e8e7f",
  "NotificationEpUsername": null,
  "NotificationEpPassword": null
}
```

cURL
curl -v -X POST 'https://<server.url>:<port>/odata/v1/Subscriptions(<subscription-id>)/OData.CSC.Resume' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq

10.12 Delete a Subscription (DELETE)

Description: Delete a subscription by id.

HTTP Method: DELETE

URL: http://<catalogue.url>:<catalogue.port>/Subscriptions(<subscription.id>)

cURL
curl -X DELETE -H 'content-type: application/json' 'http://localhost:8081/odata/v1/Subscriptions(<subscription-id>)'

11. Notification

GSS Notification component is named CDH-Notification. It is a process which will consume deletion messages from a configured Kafka queue and send notifications to some endpoints configured in the database.

11.1 Pre-requisites

11.1.1 Infrastructure Requirements

We recommend to install:

- Docker: 20.10.12 (or later)

11.1.2 Network Requirements

Application	Default port
PostgreSQL	5432

11.1.3 Software Requirements

Some tools must be installed to use the notification:

- A running PostgreSQL instance, (10.12 and after: https://www.postgresql.org/download/) with a database for the software
- A running Kafka instance (3.3.1 and later: https://kafka.apache.org/downloads)
- A running SolrCloud instance, (9.0.0 and later: https://lucene.apache.org/solr/downloads.html). It is the same instance created by **toolbox** (with JQ and JTS installed).
- Open JDK 17

The database schema update/creation scripts are inside the **toolbox** module.

11.2 Distribution links

This software is available in two formats, a zip archive and a Docker image.

11.2.1 Zip archive

Available at https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-notification/3.4.0/cdh-notification-3.4.0.zip
The archive must be downloaded and unzipped to launch Notification.

The zip archive is presented as follows:

```
notification/
├── etc/
│   ├── operational-samples/
│   │   ├── consumer-for-notification.properties
│   │   └── docker-compose.yml
│   │
│   ├── consumer.properties.sample
│   └── log4j2.xml
│
├── lib/
├── logs/
├── start.sh
└── stop.sh
```

The **consumer.properties** file of notification is used to configure the Kafka queue where to find the messages. The content of this file is described further in the sample file: *consumer.properties.sample*. A **consumer.properties** file must be present in the *etc/* directory to launch Notification application.

The **log4j2.xml** indicates where and how the logs of the application are written. It can be modified at will. The shipped configuration writes the logs inside a *logs/* directory in a file named *cdh-notification.log*. This file is rolled every day and older logs files are saved in the format *logs/cdh-notification-YYYY-MM-DD.log*.

With ZIP distribution, use the *start.sh* script to launch the catalogue and use *stop.sh* to gracefully stop it.

```
nohup ./start.sh &
./stop.sh
```

11.2.2 Docker image

Available at Docker Hub `docker pull registry.hub.docker.com/gaeldockerhub/cdh-nt:3.4.0`

You can check that the Docker image has been downloaded with the command `docker image list | grep cdh-notification`. The Docker image is based on an *openjdk:17-jdk-slim* image.

The notification application will be launched in a created */notification* directory inside the Docker image.

To launch the Docker container, the **consumer.properties** files as attached volumes. Examples for those files are available in the Zip archive.

Example:

```
docker run -d --name cdh-notification -v
/path/consumer.properties:/notification/etc/consumer.properties -it
registry.hub.docker.com/gaeldockerhub/cdh-notification:3.4.0
```

The **-d** option is used to run the Docker container in detached mode. If you want your Docker container to point to the *localhost* of the hosting machine, add the launch option `--network="host"` (https://docs.docker.com/network/host/). It means that the container shares the host machine network, the container will have the same IP and ports as the host.

For example, if you launch the container and you want to access the cdh-admin-api on localhost:8080, you must use this option. You can map the log files of the application with a local folder by adding an attached volume:

```
-v /your/path/to/logs:/notification/logs
```

To gracefully stop the container, add a timeout (in seconds) to the stop command:

```
docker stop -t 60 cdh-notification
```

11.3 Configuration

A sample of the configuration file is provided in the distribution (**consumer.properties.sample**).
The below lines, are used to configure the kafka queue where the messages are sent.

```
# MANDATORY - Semi-colon separated list of kafka brokers (ip:port)
kafka.hosts = <kafka-1>:<port>;<kafka-2>:<port>

# OPTIONAL - If your Kafka cluster is protected with authentication, provide the username
and password
#kafka.user = ***
#kafka.password = ***
```

```
# MANDATORY if kafka.topics.prefix is empty - Semi-colon separated list of topics
kafka.topics =
OPTIONAL - If configured will read from all topics starting by this value instead
of using kafka.topics.
kafka.topics.prefix = notification_

# MANDATORY - Consumer group id
group.id = notification-consumer-group

# MANDATORY - DB configuration - Database information
db.url=jdbc:postgresql://<postgres.url>:<postgres.port>/<db.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=2
#db.socketTimeout=
#db.networkTimeout=

# To encrypt/decrypt secret information in database (password for example)
secret.key.password = <secret.password>
```

RECOMMENDATION

You can find operational configuration sample files in the cdh-notification-3.4.0.zip file. These files can be used for notification configuration. They fit the main operational scenarios known. The files are stored in the "**operational-samples**" directory.

12. Admin API endpoints

This is the main scenario to use for the ingestion process (creation of all the quotas, datastores, metadatastores, producers & consumers) and, also, for the creation of catalogue instance using the cdh-admin-api.

The creation, update and deletion f a quota is applied on the fly.

12.1 Quota Schema

Quotas are assigned per user to limit three different kinds of actions for product downloads:

- TotalDownloadsQuota: Maximum volume in bytes for product download in a defined period. Default is 3GB per hour. It is called *TOTAL_DOWNLOAD* in the API.
- ParallelDownloadsQuota: Maximum number of parallel direct downloads. Default is 2. Use for *PARALLEL_DOWNLOAD* Admin API.
- TotalDownloadsLinkQuota: Maximum number of generated download links for a period in minutes. Default is 10 links in 10 minutes. Download links are generated when we ingest products from an object storage. It is named *TOTAL_DOWNLOAD_LINK* in the API.

Quotas can be assigned and updated through the user management end point /Users. If no quotas are assigned for a user, default values will be used.

We can describe a quota as below:

```
Quota
{
     name    string
     userId  string
     value   integer($int64)
     durationinteger($int64)
}
```

The key is a concatenation of *name* and *userID*. So, we cannot have 2 quotas with the same name and user.

Quota name must be one of the pre-defined following:

- *TOTAL_DOWNLOAD*
- *PARALLEL_DOWNLOAD*
- *TOTAL_DOWNLOAD_LINK*

12.2 Quota API

12.2.1 Quotas List

Description: List all quotas registered

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/quotas

Body: None

Response: List of quotas stored in the database (HTTP 200)

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/quotas" | jq`

12.2.2 Quota List for a User

Description: List of a user's quota

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/quotas/{userId}

The *userId* is a parameter of the request.

Body: None

Response: List of the quotas for the user (HTTP 200)

Error: Server error (HTTP 500) when the user doesn't exist.

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/quotas/<user>" | jq`

12.2.3 Create a Quota (POST)

Description: Create a quota.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/quotas

Body: Json description of the quota.

Response: Quota created (HTTP 200)

Error: Server error (HTTP 500) when:

the quota name is not one of the pre-defined *(TOTAL_DOWNLOAD, PARALLEL_DOWNLOAD, TOTAL_DOWNLOAD_LINK*

the *userId* is missing or empty.

cURL
`curl -X POST -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.ulr>:<port>/quotas" -d '{"name":"TOTAL_DOWNLOAD", "userId": "user", "value":2000000000000, "duration":1}' | jq`

RECOMMENDATION
You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for quotas configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Quota**" directory.

12.2.4 Create a Bulk Quota (POST)

Description: Create a set of quotas.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/quotas/bulk

Input: Json list of quotas

Response: Quotas created (HTTP 200)

Error: Server error (HTTP 500) when

there is only one quota in the list and:

- The quota name is not one of the pre-defined *(TOTAL_DOWNLOAD, PARALLEL_DOWNLOAD, TOTAL_DOWNLOAD_LINK*
- The *userId* is missing or empty.
- All the quotas of the list could not be created.

cURL
`curl -X POST -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/quotas/bulk" -d '[{"name":"TOTAL_DOWNLOAD", "userId": "user", "value":2000000000000,`

```
"duration":1},{ "name":"TOTAL_DOWNLOAD", "userId": "user2",
"value":2000000000000, "duration":1}]' | jq
```

12.2.5 Update a Quota (PATCH)

Description: Update the value or the duration of a quota by user id and name.

HTTP Method: PATCH

URL: http://<server.url>:<server.port>/<context.path>/quotas/{userId}/{name}

UserId and *name* of the quotas are the parameters.

Input: Json describing the new values for fields value and duration.

Response: The updated quota (HTTP 200)

Error:

Not Found (HTTP 404) when the quota does not exist.

Server Error (HTTP 500) for server errors.

cURL

```
curl -X PATCH -H "Authorization: Bearer ${AT}" -H "content-
type:application/json"
"https://<server.url>:<port>/quotas/{user}/{PARALLEL_DOWNLOAD}" -d
'{"value":15}' | jq
```

12.2.6 Delete a Quota (DELETE)

Description: Delete a quota by user id and name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/quotas/{userId}/{name}

UserId and *name* of the quotas are the parameters.

Input: None

Response: Informative message (HTTP 200)

Error:

Not Found (HTTP 404) when the quota does not exist.

Server Error (HTTP 500) for server errors.

cURL
`curl -X DELETE -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/quotas/{user}/{PARALLEL_DOWNLOAD}" | jq`

12.2.7 Delete all User's Quota (DELETE)

Description: Delete all the quotas for a user by user id.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/quotas/{userId}

UserId is the parameter.

Input: None

Response: Informative message (HTTP 200)

Error:

Not Found (HTTP 404) when the quotas do not exist.

Server Error (HTTP 500) for server errors.

cURL
`curl -X DELETE -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/quotas/{user}" | jq`

12.3 Swift Credentials API

This endpoint allows to perform CRUD operation on swift credentials.

12.3.1 List of all Swift Credentials (GET)

Description: Get a list of all available credentials or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/swiftcredentials

Response: List of the swift credentials created on the system or an empty list

Error: Server Error (HTTP 500) for server errors.

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-
type:application/json" "https://<server.url>:<port>/swiftcredentials | jq
```

12.3.2 Get a Swift Credential (GET)

Description: Get a swift credential by name.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/swiftcredentials/{credential.name}

Response: Get a swift credential if it exists

Error: Not Found (HTTP 404) when the swift credential does not exist.

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-
type:application/json"
"https://<server.url>:<port>/swiftcredentials/<credential-name>" | jq
```

12.3.3 Create a Swift Credential (POST)

Description: Create a swift credential.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/swiftcredentials

Body:

```
{
  "tenant": "myTenant",
  "password": "****",
  "user": "user_name",
  "url": "http://server.port/url",
  "region": "GG",
  "domain": "myDomain",
  "name": "credential2"
}
```

Response: Created credential (HTTP 200)

Error:

Conflict (HTTP 409) when the credential already exists.

Server Error (HTTP 500) for server errors.

cURL

```
curl -v -X POST 'https://<server.url>:<port>/swiftcredentials' -d '{
    "tenant": "<tenant>",
    "password": "***",
    "user": "***",
    "url": "<url>",
    "region": "<region>",
    "domain": "Default",
    "name": "<credenial-name>"
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" |
jq
```

RECOMMENDATION

You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for credential configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/SWIFT-credentials**" directory.

12.3.4 Update a Swift Credential (PATCH)

Description: Update a swift credential by name.

HTTP Method: PATCH

URL: http://<server.url>:<server.port>/<context.path>/swiftcredentials/{credential.name}

Body:

```
{
  "tenant": "tenant1",
  "password": "****",
  "user": "user_name",
  "url": "http://server.port/url",
  "region": "GG",
  "domain": "myDomain",
  "name": "credential2"
}
```

Response: Updated credential (HTTP 200)

```
{
  "tenant": "tenant1",
  "password": "*****",
  "user": "user_name",
  "url": "http://server.port/url",
```

```
  "region": "GG",
  "domain": "myDomain",
  "name": "credential2"
}
```

Error: Not found (HTTP 404) when the credential is not found.

12.3.5 Delete a Swift Credential (DELETE)

Description: Delete a swift credential by name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/swiftcredentials/{credential.name}

Response: Delete credential (HTTP 200)

Error: Not found (HTTP 404) when the credential is not found.

12.4 S3 Credentials API

This endpoint allows to perform CRUD operation on S3 credentials.

12.4.1 List all S3 Credentials (GET)

Description: Get a list of all available credentials or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/s3credentials

Response: List of S3 credentials created on the system or an empty list

Error: Server Error (HTTP 500) for server errors.

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/s3credentials | jq
```

12.4.2 Get S3 Credentials (GET)

Description: Get S3 credentials by name.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/s3credentials/{credential.name}

Response: Get a S3 credentials if it exists

Error: Not Found (HTTP 404) when the credentials do not exist.

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/s3credentials/<credential-name>" | jq`

12.4.3 Create S3 Credentials (POST)

Description: Create S3 credentials.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/s3credentials

Body:

```
{
  "accessKey": "****",
  "secretKey": "****",
  "region": "gra",
  "endpoint": "https://s3.gra.io.cloud.ovh.net",
  "name": "s3Credentials"
}
```

Response: Created credential (HTTP 200)

Error:

Conflict (HTTP 409) when the credential already exists.

Server Error (HTTP 500) for server errors.

cURL
`curl -v -X POST 'https://<server.url>:<port>/s3credentials' -d '{ "accessKey": "<access_key>", "secretKey": "<secret_key>", "region": "<region>", "endpoint": "<endpoint>", "name": "<credential_name>" }' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq`

RECOMMENDATION

You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for credential configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/S3-credentials**" directory.

12.4.4 Update S3 Credentials (PATCH)

Description: Update S3 credentials by name.

HTTP Method: PATCH

URL: http://<server.url>:<server.port>/<context.path>/s3credentials/{credential.name}

Body:

```
{
  "accessKey": "****",
  "secretKey": "****",
  "region": "gra",
  "endpoint": "https://s3.gra.io.cloud.ovh.net",
  "name": "s3Credentials"
}
```

Response: Updated credential (HTTP 200)

```
{
  "accessKey": "****",
  "secretKey": "****",
  "region": "gra",
  "endpoint": "https://s3.gra.io.cloud.ovh.net",
  "name": "s3Credentials"
}
```

Error: Not found (HTTP 404) when the credential is not found.

12.4.5 Delete S3 Credentials (DELETE)

Description: Delete S3 credentials by name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/s3credentials/{credential.name}

Response: Delete credential (HTTP 200)

Error: Not found (HTTP 404) when the credential is not found.

12.5 Datastores API

We can configure datastores using the API.

Property	Type	Mandatory	Description / Default value
permission	Array of Strings	true	"READ" Example: "permission": ["READ", "WRITE", "DELETE"]

properties	Contains a field "property"		Example: "properties": {
			"property": [
			{
			"name": "STORE_ATTACHED_FILES",
			"value": "true"
			},
			{
			"name": "KEEP_PERIOD_SECONDS",
			"value": "300"
			},
			{
			"name": "EXPOSE_IN_STAC",
			"value": "true"
			}
			]
			}

name	String	true	Name of the datastore
type	String	true	Type of the datastore. Automatically filled when creating the datastore according to its type.
parentGroup	String		Datastore group which contains the children

According to the type of Datastore, we can have more fields to fill.

HFS Datastore

Property	Type	Mandatory	Description / Default value
path	String	true	Path where to store products. It refers to the internal path configured in docker compose file
granularity	int	true	Default: 2
depth	int	true	Default: 2

Swift Datastore

Property	Type	Mandatory	Description / Default value
container	string	true	Container name
credentials	string	true	Credential name
prefixLocation	string	true	Prefix to add to a product location in an object storage

Swift Datastore group

Property	Type	Mandatory	Description / Default value
filter	String		Filter for product's name
credentials	String	true	Credential name
prefixLocation	String	true	Prefix to add to a product location in an object storage
containerPattern	String		the pattern used to identify/create containers

S3 Datastore

Property	Type	Mandatory	Description / Default value
bucket	string	true	Bucket name
credentials	string	true	S3 Credentials name
prefixLocation	string	true	Prefix to add to a product location in an object storage

S3 Datastore group

Property	Type	Mandatory	Description / Default value
filter	String		Filter for product's name
credentials	String	true	Credential name
prefixLocation	String	true	Prefix to add to a product location in an object storage
containerPattern	String		Pattern used to identify/create buckets

TimebasedGroup Datastore

Property	Type	Mandatory	Description / Default value
filter	String	true	Path where to store products
policy	enum	true	Datastore policy used on the datastore. Possible values: "USER_DEFINED_PRIORITY_POLICY" or "BASIC_STORE_PRIORITY_POLICY"

children	A set of datastores which are contained by the timebased datastore group	true	A child of the timebased datastore group

12.5.1 Generic Search

Description: Get all available Datastores .

HTTP Method: GET

URL:
http://<server.url>:<server.port>/<context.path>/datastores?type={type}&top={top.value}&skip={skip.value}

http://<server.url>:<server.port>/<context.path>/datastores?name={name.value}

Response: Get all datastores matching parameters (HTTP 200)

- Name
- Type
- Top (some datastores)
- Skip (Skip some datastores)

12.5.2 HFS Datastore API

12.5.2.1 List of all HFS Datastores (GET)

Description: Get a list of all available HFS Datastores or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/hfs

Response: Get an empty list or the list of HFS datastores (HTTP 200)

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/datastores/hfs | jq`

RECOMMENDATION

You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for HFS datastore configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Datastore/HFS**" directory.

12.5.2.2 Get a HFS Datastore (GET)

Description: Get a HFS Datastore by name.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/hfs/{hfs.datastore.name}

Response: Get a HFS datastore (HTTP 200)

Error:

Not found (HTTP 404) when the datastore is not found.

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-
type:application/json"
"https://<server.url>:<port>/datastores/hfs/<datastore-name> | jq
```

12.5.2.3 Create a HFS Datastore (POST)

Description: Create a HFS Datastore.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/datastores/hfs

Body:

```
{
  "permission": [
    "READ",
    "WRITE"
  ],
  "properties": {
    "property": [
      {
        "name": "MAX_SIZE",
        "value": "5000"
      }
    ]
  },
  "name": "HFS3",
```

```
  "path": "/path/to/datastore",  // internal path in docker compose file
  "depth": 2,
  "granularity": 2
}
```

Response:

```
{
  "permission": [
    "READ",
    "WRITE"
  ],
  "properties": {
    "property": [
      {
        "name": "MAX_SIZE",
        "value": "5000"
      }
    ]
  },
  "name": "HFS3",
  "path": "/path/to/datastore",
  "depth": 2,
  "granularity": 2,
  "parentGroup": null,
  "type": "HFS"
}
```

Error:

HTTP 400 - A field or the object is not correct.

HTTP 500 - An error occurred. See the logs.

cURL

```
curl -v -X POST 'http://localhost:8082/datastores/hfs' -d '{
    "permission": [
        "READ",
        "WRITE",
        "DELETE"
    ],
    "properties": {
        "property": [
            {
                "name": "MAX_SIZE",
```

```
                "value": "5000"
            },
            {
                "name": "STORE_ATTACHED_FILES",
                "value": "true"
            },
            {
                "name": "KEEP_PERIOD_SECONDS",
                "value": "0"
            }
        ]
    },
    "name": "hfs",
    "path": "/Users/JohnDoe/datastore",
    "depth": 2,
    "granularity": 2,
    "type": "HFS"
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}"| jq
```

12.5.2.4 Update a HFS Datastore (PATCH)

Description: Update a HFS Datastore by name.

HTTP Method: PATCH

URL: http://<server.url>:<server.port>/<context.path>/datastores/hfs/{hfs.datastore.name}

Optional param: restartIngesters=<true/false>
If true and any consumer instance is running and use this DataStore, restarts them.

Body:

```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "KEEP_PERIOD_SECONDS",
        "value": "6000"
      },
      {
        "name": "STORE_ATTACHED_FILES",
        "value": "true"
      },
```

```
      {
        "name": "PRIORITY",
        "value": "0"
      }
    ]
  },
  "name": "HFS3",
  "path": "/path/to/datastore",
  "depth": 2,
  "granularity": 2,
  "parentGroup": null,
  "type": "HFS"
}
```


Response: Get the update HFS datastore (HTTP 200)


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "KEEP_PERIOD_SECONDS",
        "value": "6000"
      },
      {
        "name": "STORE_ATTACHED_FILES",
        "value": "true"
      },
      {
        "name": "PRIORITY",
        "value": "0"
      }
    ]
  },
  "name": "HFS3",
  "path": "/path/to/datastore",
  "depth": 2,
  "granularity": 2,
  "parentGroup": null,
  "type": "HFS"
}
```

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the log.

12.5.2.5 Delete a HFS Datastore (DELETE)

Description: Delete a HFS Datastore by name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/datastores/hfs/{hfs.datastore.name}

Response: A message of deletion successfully done (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

cURL
`curl -v -X DELETE -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/datastores/hfs/<datastore-name> | jq`

12.5.3 Swift Datastore API

12.5.3.1 List of all Swift Datastores (GET)

Description: Get a list of all available Swift Datastores or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/swift

Response: Get an empty list or the list of Swift datastores (HTTP 200)

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/datastores/swift" | jq`

RECOMMENDATION

You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for swift datastore configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Datastore/SWIFT**" directory.

12.5.3.2 Get a Swift Datastore (GET)

Description: Get a swift Datastore by name.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/swift/<swift.datastore.name>

Response: Get a Swift datastores (HTTP 200)

Error:

Not found (HTTP 404) when the datastore is not found.

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-
type:application/json"
"https://<server.url>:<port>/datastores/swift/<swift-datastore-name>" | jq
```

12.5.3.3 Create a Swift Datastore (POST)

Description: Create a swift Datastore.

HTTP Method: POST

NOTE: The credential used must be created previously.

URL: http://<server.url>:<server.port>/<context.path>/datastores/swift

Response: Created a Swift datastores (HTTP 200)

Body:

```
{
  "permission": [
    "READ",
    "WRITE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_ATTACHED_FILES",
        "value": "true"
      }
    ]
  },
```

```
  "name": "SwiftDS",
  "credentials": "credential1",
  "container": "containerSwift",
  "prefixLocation": "instrument"
}
```


Response:


```
{
  "permission": [
    "READ",
    "WRITE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_ATTACHED_FILES",
        "value": "true"
      }
    ]
  },
  "name": "SwiftDS",
  "credentials": "credential1",
  "container": "containerSwift",
  "prefixLocation": "instrument",
  "parentGroup": null,
  "type": "SWIFT"
}
```


Error:

HTTP 400 - A field or the object is not correct.

HTTP 500 - An error occurred. See the logs.

cURL

```
curl -v -X POST 'https://<server.url>:<port>/datastores/swift' -d '{
  "permission": [
    "READ",
    "WRITE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_ATTACHED_FILES",
        "value": "true"
      }
    ]
  },
  "name": "SwiftDS",
```

```
  "credentials": "credential1",
  "container": "containerSwift",
  "prefixLocation": "instrument",
  "parentGroup": null,
  "type": "SWIFT"
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" |
jq
```

12.5.3.4 Update a Swift Datastore (PATCH)

Description: Update a swift Datastore by name.

HTTP Method: PATCH

URL: http://<server.url>:<server.port>/<context.path>/datastores/swift/<swift.datastore.name>

Optional param: restartIngesters=<true/false>
If true and any consumer instance is running and use this DataStore, restarts them.

Body:

```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "KEEP_PERIOD_SECONDS",
        "value": "50000"
      }
    ]
  },
  "name": "SwiftDS",
  "credentials": "credential1",
  "container": "containerSwift",
  "prefixLocation": "instrument",
  "parentGroup": null,
  "type": "SWIFT"
}
```

Response:

```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
```

```
  ],
  "properties": {
    "property": [
      {
        "name": "KEEP_PERIOD_SECONDS",
        "value": "50000"
      }
    ]
  },
  "name": "SwiftDS",
  "credentials": "credential1",
  "container": "containerSwift",
  "prefixLocation": "instrument",
  "parentGroup": null,
  "type": "SWIFT"
}
```


Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.5.3.5 Delete a Swift Datastore (DELETE)

Description: Delete a swift Datastore by name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/datastores/swift/<swift.datastore.name>

Response: A message to confirm the deletion of the Swift datastore (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

cURL

```
curl -v -X DELETE -H "Authorization: Bearer ${AT}" -H "content-
type:application/json"
"https://<server.url>:<port>/datastores/swift/<swift-datastore-name>" | jq
```

12.5.4 SwiftGroup Datastore API

12.5.4.1 List of all SwiftGroup Datastores (GET)

Description: Get a list of all available Swift Datastores groups or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/swiftgroup

Response: Get an empty list or the list of Swift datastores groups (HTTP 200)

12.5.4.2 Get a Swift Datastore Group (GET)

Description: Get a Swift Datastores group by name.

HTTP Method: GET

URL:
http://<server.url>:<server.port>/<context.path>/datastores/swiftgroup/<swiftgroup.datastore.name>

Response: Get an a Swift datastores group (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

12.5.4.3 Create a Swift Datastore Group (POST)

Description: Create a Swift Datastores group.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/datastores/swiftgroup

NOTE: the credential must be created previously.

Body:

```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      },
      {
        "name": "SAVE_AS_MULTIPART",
        "value": "false"
      },
      {
```

```
        "name": "MULTIPART_THREADS",
        "value": "4"
      },
      {
        "name": "KEEP_PERIOD_SECONDS",
        "value": "172800"
      }
    ]
  },
  "name": "SwiftGroup",
  "credentials": "credential1",
  "filter": ".*",
  "containerPattern": {
    "type": "mapperPattern",
    "patternMapper": "S1.*:CDH-Sentinel-1,S2.*:CDH-Sentinel-2,S3.*:CDH-Sentinel-3,S5.*:CDH-Sentinel-5P,.*:CDH-Other"
  },
  "prefixLocation": "instrument",
  "parentGroup": null
}
```


Response:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      },
      {
        "name": "SAVE_AS_MULTIPART",
        "value": "false"
      },
      {
        "name": "MULTIPART_THREADS",
        "value": "4"
      },
      {
        "name": "KEEP_PERIOD_SECONDS",
        "value": "172800"
      }
```

```
    ]
  },
  "name": "SwiftGroup",
  "credentials": "credential1",
  "filter": ".*",
  "containerPattern": {
    "type": "mapperPattern",
    "patternMapper": "S1.*:CDH-Sentinel-1,S2.*:CDH-Sentinel-2,S3.*:CDH-Sentinel-3,S5.*:CDH-Sentinel-5P,.*:CDH-Other"
  },
  "prefixLocation": "instrument",
  "parentGroup": null,
  "type": "SWIFT_GROUP"
}
```


Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - An error occurred. See the logs.

12.5.4.4 Update a Swift Datastore Group (PATCH)

Description: Update a swift Datastores group by name.

HTTP Method: PATCH

URL:
http://<server.url>:<server.port>/<context.path>/datastores/swiftgroup/<swiftgroup.datastore.name>

Optional param: restartIngesters=<true/false>
If true and any consumer instance is running and use this DataStore, restarts them.

Body:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      },
      {
        "name": "SAVE_AS_MULTIPART",
```

```
          "value": "false"
        },
        {
          "name": "MULTIPART_THREADS",
          "value": "4"
        },
        {
          "name": "KEEP_PERIOD_SECONDS",
          "value": "172800"
        }
      ]
    },
    "credentials": "credential1",
    "filter": ".*",
    "containerPattern": {
      "type": "mapperPattern",
      "patternMapper": "S1.*:Test-Sentinel-1,S2.*:Test-Sentinel-2,S3.*:Test-Sentinel-3,S5.*:Test-Sentinel-5P,.*:Other"
    },
    "prefixLocation": "instrument/productType/year/month/day"
}
```


Response:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      },
      {
        "name": "SAVE_AS_MULTIPART",
        "value": "false"
      },
      {
        "name": "MULTIPART_THREADS",
        "value": "4"
      },
      {
        "name": "KEEP_PERIOD_SECONDS",
        "value": "172800"
      }
```

```
    ]
  },
  "name": "SwiftGroup2",
  "credentials": "credential1",
  "filter": ".*",
  "containerPattern": {
    "type": "mapperPattern",
    "patternMapper": "S1.*:Test-Sentinel-1,S2.*:Test-Sentinel-2,S3.*:Test-Sentinel-3,S5.*:Test-Sentinel-5P,.*:Other"
  },
  "prefixLocation": "instrument/productType/year/month/day",
  "parentGroup": null,
  "type": "SWIFT_GROUP"
}
```


Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.5.4.5 Delete a Swift Datastore Group (DELETE)

Description: Delete a swift datastore group by name.

HTTP Method: DELETE

URL:
http://<server.url>:<server.port>/<context.path>/datastores/swiftgroup/<swiftgroup.datastore.name>

Response: Store successfully deleted (HTTP 200)

Error:

HTTP 404 - If datastore does not exists.

HTTP 500 - Internal error. See the logs.

12.5.5 S3 Datastore API

For such datastores, the property *STORE_AS_MULTIPART* should be set to *false*.

12.5.5.1 List of all S3 Datastores (GET)

Description: Get a list of all available S3 Datastores or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/s3

Response: Get an empty list or the list of S3 datastores (HTTP 200)

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/datastores/s3" | jq`

RECOMMENDATION
You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for s3 datastore configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Datastore/S3**" directory.

12.5.5.2 Get a S3Datastore (GET)

Description: Get a S3 Datastore by name.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/s3/<s3.datastore.name>

Response: Get a S3 datastore (HTTP 200)

Error:

Not found (HTTP 404) when the datastore is not found.

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/datastores/s3/<s3-datastore-name>" | jq`

12.5.5.3 Create a S3 Datastore (POST)

Description: Create a S3 Datastore.

HTTP Method: POST

NOTE: The credentials used must have been created previously.

URL: http://<server.url>:<server.port>/<context.path>/datastores/s3

Response: Created a S3 datastore (HTTP 200)

Body:


```
{
  "permission": [
    "READ",
    "WRITE",
    “DELETE”
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_ATTACHED_FILES",
        "value": "true"
      },
      {
        "name": "EXPOSE_IN_STAC",
        "value": "true"
      }
    ]
  },
  "name": "S3Store",
  "credentials": "s3credentials",
  "bucket": "test-S3",
  "prefixLocation": "instrument",
  "type": "S3"
}
```


Response:


```
{
  "permission": [
    "READ",
    "WRITE",
    “DELETE”
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_ATTACHED_FILES",
        "value": "true"
      },
      {
        "name": "EXPOSE_IN_STAC",
```

```
        "value": "true"
      }
    ]
  },
  "name": "S3Store",
  "credentials": "s3credentials",
  "bucket": "test-S3",
  "prefixLocation": "instrument",
  "parentGroup": null,
  "type": "S3"
}
```

Error:

HTTP 400 - A field or the object is not correct.

HTTP 500 - An error occurred. See the logs.

cURL

```
curl -v -X POST 'https://<server.url>:<port>/datastores/s3' -d @store.json -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

12.5.5.4 Update a S3 Datastore (PATCH)

Description: Update a S3 Datastore by name.

HTTP Method: PATCH

URL: http://<server.url>:<server.port>/<context.path>/datastores/s3/<s3.datastore.name>

Optional param: restartIngesters=<true/false>
If true and any consumer instance is running and use this DataStore, restarts them.

Body:

```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      }
    ]
  },
```

```
  "name": "S3Store",
  "credentials": "s3Credentials",
  "bucket": "test-S3",
  "prefixLocation": "instrument"
}
```


Response:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      }
    ]
  },
  "name": "S3Store",
  "credentials": "s3Credentials",
  "bucket": "test-S3",
  "prefixLocation": "instrument",
  "parentGroup": null,
  "type": "S3"
}
```


Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.5.5.5 Delete a S3 Datastore (DELETE)

Description: Delete a S3 Datastore by name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/datastores/s3/<s3.datastore.name>

Response: A message to confirm the deletion of the S3 datastore (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

cURL
`curl -v -X DELETE -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/datastores/s3/<s3-datastore-name>" | jq`

12.5.6 S3Group Datastore API

12.5.6.1 List of all S3Group Datastores (GET)

Description: Get a list of all available S3 Datastore groups or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/s3group

Response: Get an empty list or the list of S3 datastore groups (HTTP 200)

12.5.6.2 Get a S3 Datastore Group (GET)

Description: Get a S3 Datastores group by name.

HTTP Method: GET

URL:
http://<server.url>:<server.port>/<context.path>/datastores/s3group/<s3group.datastore.name>

Response: Get a S3 datastore group (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

12.5.6.3 Create a S3 Datastore Group (POST)

Description: Create a S3 Datastore group.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/datastores/s3group

NOTE: the credentials must have been created previously.

Body:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      },
      {
        "name": "SAVE_AS_MULTIPART",
        "value": "true"
      },
      {
        "name": "MULTIPART_THREADS",
        "value": "4"
      },
      {
        "name": "EXPOSE_IN_STAC",
        "value": "true"
      }
    ]
  },
  "name": "S3Group",
  "credentials": "s3credentials",
  "filter": ".*",
  "containerPattern": {
    "type": "mapperPattern",
    "patternMapper": "S1.*:CDH-Sentinel-1,S2.*:CDH-Sentinel-2,S3.*:CDH-Sentinel-3,S5.*:CDH-Sentinel-5P,.*:CDH-Other"
  },
  "prefixLocation": "instrument",
  "parentGroup": null
}
```


Response:


```
{
  "permission": [
    "READ",
```

```
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      },
      {
        "name": "SAVE_AS_MULTIPART",
        "value": "true"
      },
      {
        "name": "MULTIPART_THREADS",
        "value": "4"
      },
      {
        "name": "EXPOSE_IN_STAC",
        "value": "true"
      }
    ]
  },
  "name": "S3Group",
  "credentials": "s3credentials",
  "filter": ".*",
  "containerPattern": {
    "type": "mapperPattern",
    "patternMapper": "S1.*:CDH-Sentinel-1,S2.*:CDH-Sentinel-2,S3.*:CDH-Sentinel-3,S5.*:CDH-Sentinel-5P,.*:CDH-Other"
  },
  "prefixLocation": "instrument",
  "parentGroup": null,
  "type": "S3_GROUP"
}
```


Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - An error occurred. See the logs.

12.5.6.4 Update a S3 Datastore Group (PATCH)

Description: Update a S3 Datastore group by name.

HTTP Method: PATCH

URL:
http://<server.url>:<server.port>/<context.path>/datastores/s3group/<s3group.datastore.name>

Optional param: restartIngesters=<true/false>
If true and any consumer instance is running and use this DataStore, restarts them.

Body:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      },
      {
        "name": "SAVE_AS_MULTIPART",
        "value": "true"
      },
      {
        "name": "MULTIPART_THREADS",
        "value": "4"
      }
    ]
  },
  "credentials": "s3credentials",
  "filter": ".*",
  "containerPattern": {
    "type": "mapperPattern",
    "patternMapper": "S1.*:Test-Sentinel-1,S2.*:Test-Sentinel-2,S3.*:Test-Sentinel-3,S5.*:Test-Sentinel-5P,.*:Other"
  },
  "prefixLocation": "mission/instrument/productType/year/month/day"
}
```


Response:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
```

```
  ],
  "properties": {
    "property": [
      {
        "name": "STORE_BY_NAME",
        "value": "true"
      },
      {
        "name": "SAVE_AS_MULTIPART",
        "value": "true"
      },
      {
        "name": "MULTIPART_THREADS",
        "value": "4"
      }
    ]
  },
  "name": "S3Group",
  "credentials": "s3credentials",
  "filter": ".*",
  "containerPattern": {
    "type": "mapperPattern",
    "patternMapper": "S1.*:Test-Sentinel-1,S2.*:Test-Sentinel-2,S3.*:Test-Sentinel-3,S5.*:Test-Sentinel-5P,.*:Other"
  },
  "prefixLocation": "mission/instrument/productType/year/month/day",
  "parentGroup": null,
  "type": "S3_GROUP"
}
```


Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.5.6.5 Delete a S3 Datastore Group (DELETE)

Description: Delete a S3 datastore group by name.

HTTP Method: DELETE

URL:
http://<server.url>:<server.port>/<context.path>/datastores/s3group/<s3group.datastore.name>

Response: Store successfully deleted (HTTP 200)

Error:

HTTP 404 - If datastore does not exists.

HTTP 500 - Internal error. See the logs.

12.5.7 Timebased Datastore API

Policy can have the following values:

- BASIC_STORE_PRIORITY_POLICY
- USER_DEFINED_PRIORITY_POLICY

RECOMMENDATION
You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for timebased datastore configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Datastore**" directory.

12.5.7.1 List of all Timebased Datastore (GET)

Description: Get a list of all available timebased datastore groups or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/datastores/timebased

Response: Get a list of timebased datastore groups or an empty list (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.5.7.2 Get a Timebased Datastore (GET)

Description: Get a timebased datastore group by name.

HTTP Method: GET

URL:
http://<server.url>:<server.port>/<context.path>/datastores/timebased/<timebased.datastore.name>

Response: Get a timebased datastore group (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.5.7.3 Create a Timebased Datastore Group (POST)

Description: Create a timebased datastore group.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/datastores/timebased

NOTE: Children datastores will be created at the same time. There will be an error if a children datastore already exists.

Body:

```
{
  "name": "hfsTimeGroup_S1",
  "type": "TIME_GROUP",
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "filter": ".*",
  "policy": "BASIC_STORE_PRIORITY_POLICY",
  "children": [
    {
      "permission": [
        "READ",
        "WRITE",
        "DELETE"
      ],
      "properties": {
        "property": [
          {
            "name": "STORE_BY_NAME",
            "value": true
          },
          {
            "name": "SAVE_AS_MULTIPART",
```

```
            "value": false
          },
          {
            "name": "KEEP_PERIOD_SECONDS",
            "value": "432000"
          }
        ]
      },
      "name": "hfs_S1",
      "path": "/ingest/folder1",
      "depth": 0,
      "granularity": 2,
      "parentGroup": "hfsTimeGroup_S1",
      "type": "HFS"
    }
  ]
}
```


Response: Create a timebased datastore group (HTTP 200)


```
{
  "name": "hfsTimeGroup_S1",
  "type": "TIME_GROUP",
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "filter": ".*",
  "policy": "BASIC_STORE_PRIORITY_POLICY",
  "children": [
    {
      "permission": [
        "READ",
        "WRITE",
        "DELETE"
      ],
      "properties": {
        "property": [
          {
            "name": "STORE_BY_NAME",
            "value": true
          },
          {
            "name": "SAVE_AS_MULTIPART",
            "value": false
          },
```

```
            {
              "name": "KEEP_PERIOD_SECONDS",
              "value": "432000"
            }
          ]
        },
        "name": "hfs_S1",
        "path": "/ingest/folder1",
        "depth": 0,
        "granularity": 2,
        "parentGroup": "hfsTimeGroup_S1",
        "type": "HFS"
    }
  ]
}
```

Error:

HTTP 400 - A field or the object is not correct.

HTTP 500 - Internal error. See the logs.

12.5.7.4 Update a Timebased Datastore Group (PATCH)

Description: Update a timebased datastore group by name.

HTTP Method: PATCH

URL:
http://<server.url>:<server.port>/<context.path>/datastores/timebased/<timebased.datastore.name>

Optional param: restartIngesters=<true/false>
If true and any consumer instance is running and use this DataStore, restarts them.

Body:

```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "properties": {
    "property": [
      {
```

```
        "name": "MAX_SIZE",
        "value": "500000000"
      }
    ]
  },
  "name": "TB3",
  "filter": ".*",
  "policy": "BASIC_STORE_PRIORITY_POLICY",
  "children": [
    {
      "permission": [
        "READ",
        "WRITE"
      ],
      "properties": {
        "property": [
          {
            "name": "KEEP_PERIOD_SECONDS",
            "value": "300"
          }
        ]
      },
      "name": "HFS3ForGroup",
      "path": "/tmp/path",
      "depth": 2,
      "granularity": 2,
      "parentGroup": "TB3",
      "type": "HFS"
    },
    {
      "permission": [
        "READ",
        "WRITE",
        "DELETE"
      ],
      "properties": {
        "property": [
          {
            "name": "KEEP_PERIOD_SECONDS",
            "value": "300"
          },
          {
            "name": "STORE_BY_NAME",
            "value": "true"
          },
          {
```

```
            "name": "SAVE_AS_MULTIPART",
            "value": false
          },
          {
            "name": "MULTIPART_THREADS",
            "value": "4"
          }
        ]
      },
      "name": "SwiftDsForGroup",
      "credentials": "credential1",
      "container": "container1-grp",
      "prefixLocation": "instrument",
      "parentGroup": "TB3",
      "type": "SWIFT"
    },
    {
      "permission": [
        "READ",
        "WRITE",
        "DELETE"
      ],
      "properties": {
        "property": [
          {
            "name": "STORE_BY_NAME",
            "value": "true"
          },
          {
            "name": "SAVE_AS_MULTIPART",
            "value": false
          },
          {
            "name": "MULTIPART_THREADS",
            "value": "4"
          },
          {
            "name": "KEEP_PERIOD_SECONDS",
            "value": "172800"
          }
        ]
      },
      "name": "SwiftGroupForGroup",
      "credentials": "credential1",
      "filter": ".*",
      "containerPattern": {
```

```
          "type": "mapperPattern",
          "patternMapper": "S1.*:Test-Sentinel-1,S2.*:Test-Sentinel-2,S3.*:Test-Sentinel-3,S5.*:Test-Sentinel-5P,.*:Other"
        },
        "prefixLocation": "instrument/productType/year/month/day",
        "parentGroup": "TB3",
        "type": "SWIFT_GROUP"
      }
    ],
    "type": "TIME_GROUP"
}
```


Response: Update a timebased datastore group (HTTP 200)


```
{
    "permission": [
      "READ",
      "WRITE",
      "DELETE"
    ],
    "properties": {
      "property": [
        {
          "name": "MAX_SIZE",
          "value": "500000000"
        }
      ]
    },
    "name": "TB3",
    "filter": ".*",
    "policy": "BASIC_STORE_PRIORITY_POLICY",
    "children": [
      {
        "permission": [
          "READ",
          "WRITE"
        ],
        "properties": {
          "property": [
            {
              "name": "KEEP_PERIOD_SECONDS",
              "value": "300"
            }
          ]
        },
        "name": "HFS3ForGroup",
        "path": "/tmp/path",
```

```
      "depth": 2,
      "granularity": 2,
      "parentGroup": "TB3",
      "type": "HFS"
    },
    {
      "permission": [
        "READ",
        "WRITE",
        "DELETE"
      ],
      "properties": {
        "property": [
          {
            "name": "KEEP_PERIOD_SECONDS",
            "value": "300"
          },
          {
            "name": "STORE_BY_NAME",
            "value": "true"
          },
          {
            "name": "SAVE_AS_MULTIPART",
            "value": false
          },
          {
            "name": "MULTIPART_THREADS",
            "value": "4"
          }
        ]
      },
      "name": "SwiftDsForGroup",
      "credentials": "credential1",
      "container": "container1-grp",
      "prefixLocation": "instrument",
      "parentGroup": "TB3",
      "type": "SWIFT"
    },
    {
      "permission": [
        "READ",
        "WRITE",
        "DELETE"
      ],
      "properties": {
        "property": [
```

```
            {
              "name": "STORE_BY_NAME",
              "value": "true"
            },
            {
              "name": "SAVE_AS_MULTIPART",
              "value": false
            },
            {
              "name": "MULTIPART_THREADS",
              "value": "4"
            },
            {
              "name": "KEEP_PERIOD_SECONDS",
              "value": "172800"
            }
          ]
        },
        "name": "SwiftGroupForGroup",
        "credentials": "credential1",
        "filter": ".*",
        "containerPattern": {
          "type": "mapperPattern",
          "patternMapper": "S1.*:Test-Sentinel-1,S2.*:Test-Sentinel-2,S3.*:Test-Sentinel-3,S5.*:Test-Sentinel-5P,.*:Other"
        },
        "prefixLocation": "instrument/productType/year/month/day",
        "parentGroup": "TB3",
        "type": "SWIFT_GROUP"
      }
    ],
    "type": "TIME_GROUP"
}
```


Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.5.7.5 Delete a Timebased Datastore Group (DELETE)

Description: Delete a timebased datastore group by name.

HTTP Method: DELETE

URL:
http://<server.url>:<server.port>/<context.path>/datastores/timebased/<timebased.datastore.name>

Response: A message of confirmation of deletion (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.6 Metadatastores API

12.6.1 Solr Metadatastore API

RECOMMENDATION
You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for the metadata store configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Metadatastore**" directory.

12.6.1.1 List of all Solr Metadatastores (GET)

Description: Get a list of all available Solr metadatastores or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/metadatastores/solr

Response: Get a list of solr metadatastores or an empty list (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-
type:application/json" "https://<server.url>:<port>/metadatastores/solr" |
jq
```

12.6.1.2 Get a Solr Metadatastore (GET)

Description: Get a Solr metadatastore by name.

HTTP Method: GET

URL:
http://<server.url>:<server.port>/<context.path>/metadatastores/solr/<solr.metadatastore.name>

Response: Get a Solr metadatastores (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/metadatastores/solr/<metadatastore-name>" | jq
```

12.6.1.3 Create a Solr Metadatastore (POST)

Description: Create a Solr metadatastore.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/metadatastores/solr

Body:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "strategies": {
    "strategy": [
      {
        "name": "REQUEST_BY_ID",
        "value": 0
      }
    ]
  },
  "name": "SolrMetadatastore2",
  "hosts": "https://solr.url.test/solr",
  "clientType": "LBHttp",
  "user": "solaR",
  "password": "******",
  "collection": "cdh-main",
  "defaultSort": "name desc",
  "defaultTop": 100,
```

```
  "maxSkip": 100
}
```


Note:

- The number of maxSkip should be greater than the number of products needing deletion.
- Starting from release 1.4.2, while creating the metadata store, it will include three additional internal properties in the response that should not be modified.


```
  "properties": null,
  "visitorBuilder": "fr.gael.gss.core.store.solr.ProductSolrVisitorBuilder",
  "transformer": "fr.gael.gss.core.store.solr.ProductSolrTransformer"
```


Response: A created metadatastore response (HTTP 200)

Error:

HTTP 400 - A field or the object is not correct.

HTTP 500 - Internal error. See the logs.

cURL

```
curl -v -X POST 'https://<server.url>:<port>/metadatastores/solr' -d '{
    "permission": [
        "READ",
        "WRITE",
        "DELETE"
    ],
    "strategies": {
        "strategy": [
            {
                "name": "REQUEST_BY_ID",
                "value": 0
            }
        ]
    },
    "name": "SolrMetadatastore2",
    "hosts": "https://solr.url.test/solr",
    "clientType": "LBHttp",
    "user": "solaR",
    "password": "*****",
    "collection": "cdh-main",
    "defaultSort": "name desc",
    "defaultTop": 100,
    "maxSkip": 100
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" |
jq
```

12.6.1.4 Update a Solr Metadatastore (PATCH)

Description: Update a Solr metadatastore by name.

HTTP Method: PATCH

URL:
http://<server.url>:<server.port>/<context.path>/metadatastores/solr/<solr.metadatastore.name>

Optional param: restartIngesters=<true/false>
If true and any consumer instance is running and use this Metadatastore, restarts them.

Body:


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
  "strategies": {
    "strategy": [
      {
        "name": "REQUEST_BY_ID",
        "value": 0
      }
    ]
  },
  "name": "SolrMetadatastore2",
  "hosts": "https://solr.url.test/solr",
  "clientType": "LBHttp",
  "user": "solaR",
  "password": "*****",
  "collection": "cdh-main",
  "defaultSort": "name desc",
  "defaultTop": 100,
  "maxSkip": 100,
  "storage": "SolrMetadatastore_test"
}
```


Response: Updated solr metadatastore (HTTP 200)


```
{
  "permission": [
    "READ",
    "WRITE",
    "DELETE"
  ],
```

```
  "strategies": {
    "strategy": [
      {
        "name": "REQUEST_BY_ID",
        "value": 0
      }
    ]
  },
  "name": "SolrMetadatastore2",
  "hosts": "https://solr.url.test/solr",
  "clientType": "LBHttp",
  "user": "solaR",
  "password": "*****",
  "collection": "cdh-main",
  "defaultSort": "name desc",
  "defaultTop": 100,
  "maxSkip": 100,
  "storage": "SolrMetadatastore_test"
}
```


Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.6.1.5 Delete a Solr Metadatastore (DELETE)

Description: Delete a Solr metadatastore by name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/metadatastores/solr/<solr.metadatastore.name>

Response: A confirmation of deletion of Solr metadatastore (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

cURL

```
curl -v -X DELETE -H "Authorization: Bearer ${AT}" -H "content-type:application/json"
"https://<server.url>:<port>/metadatastores/solr/<metadatastore-name>" | jq
```

12.7 Ingester Producer API

For producers, the property source's type will have the following value:

Source Type	Description
fr.gael.gss.ingest.ingester.ProducerFolderConf	Source folder
fr.gael.gss.admin.ingester.payload.ProducerSwiftConfPayload	Source swift
fr.gael.gss.ingest.ingester.ProducerOdataConf	Source Odata
fr.gael.gss.admin.ingester.payload.ProducerS3ConfPayload	Source S3
fr.gael.gss.ingest.ingester.ProducerStacConf	Source STAC

12.7.1 Reprocess

It's possible to re-ingest already ingested products and overwrite them. A product is ingested if its status is INGESTED in database. If so, this product is eligible to be reprocessed.

In the producer and in the consumer, simply add the element

<ingester:reprocess>true</ingester:reprocess> in XML or *"reprocess": true,* in JSON.

12.7.2 ProcessError

A producer can reproduce messages for products in error state. Add this to your producer configuration:

in XML:

<ingester:processError active="true" retries="3">

in JSON:

"processError": { "active": true, "retries": 3 }

The number of retries attempt is not persisted in database.

If the products in error have been put in a specific folder, you can configure a producer/consumer to handle those products. The producer must be configured to reproduce message for products in error. The consumer should be configured to not move products in error.

Property	Type	Mandatory	Default value

hosts	String	true	
user	String		
password	String		
topic	String	true	
pushInterval	int		60
filter	String		
processError			
reprocess	boolean		false
dataSource	String		Unknown
source		true	

Folder source

Property	Type	Mandatory	Default value
path	String	true	Refers to the internal path configured in Docker compose file.

Swift source

Property	Type	Mandatory	Default value
credentials	String	true	
containers	String	true	
infiniteLoop	boolean		false

S3 source

Property	Type	Mandatory	Default value
credentials	String	true	
buckets	String	true	
infiniteLoop	boolean		false

OData source

Property	Type	Mandatory	Default value
serviceRootUrl	String	true	
top	int		10
lastPublicationDate	String		
filter	String		
type	String		csc Other value: dhus, cdse (note on the page about cdse)
assumedFormat	String		Mandatory for type cdse

fetchAttributes	boolean		false
fetchQuicklook	boolean		false
useDateFromDb	boolean		true
geoPostFilter	String		
auth.user	String	true	
auth.password	String	true	
auth.clientId	String		
auth.tokenEndpoint	String		
auth.type	String	true	Values: basic or oauth2

Note:

- The assumedFormat property is mandatory to ingest with both dhus and cdse type.

STAC source

Property	Type	Mandatory	Default value
serviceRootUrl	String	true	
top	int		10
pivotDate	String		datetime
pivotDateValue	String		Current date
filter	Array of name/value pairs		
collection	String		
assumedFormat	String		Mandatory for type cdse
fetchQuicklook	boolean		false
useDateFromDb	boolean		true
sortby	String		
auth.user	String	true	
auth.password	String	true	
auth.clientId	String		
auth.tokenEndpoint	String		
auth.type	String	true	Values: basic or oauth2

Note:

- The assumed format property is mandatory to ingest from CDSE.

WARNING: In the current implementation, all the default values will be applied if the property is not present during update.

12.7.3 List of all Producers (GET)

Description: Get a list of all available producers or return empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/producers

NOTE: We can also use the following parameters for the search:

- **sourceType:** to retrieve all the producers with a specific type. Its value are FOLDER, SWIFT, ODATA
- **top:** to retrieve a specific number or producers
- **skip:** to skip some objects in the response

We obtained the URL
http://<server.url>:<server.port>/<context.path>/producers?sourceType=<source.type>&top=<top.value>&skip=<skip.value>

Response: A list of all producers or an empty list (HTTP 200)

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/producers" | jq`

12.7.4 Get a Producer (GET)

Description: Get a producer by name.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/producers/<producer.name>

Response: The producer found (HTTP 200)

Error:

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

cURL
`curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/producers/<producer-name>" | jq`

12.7.5 Create a Producer (POST)

Description: Create a producer.

The following four properties are mandatory and must not be null or empty:

- name
- hosts
- topics
- source

RECOMMENDATION
You can find some JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for ingester configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Producer**" directory. The "JSON-samples/Producer" directory contains the operational scenarios for the following sources: CDSE, DHuS, Folder and GSS.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/producers

Body: Folder source

```
{
  "name": "producer1",
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "exampleTopic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 1
  },
  "reprocess": false,
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerFolderConf",
    "path": "/path/for/source"
  }
}
```

Swift source:

NB: Credential with the correct name must exist before creating a Swift producer.

```
{
  "name": "producerSwift",
  "hosts": "host1,host2",
  "topic": "ingestion_topic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 1
  },
  "reprocess": false,
  "source": {
    "sourceType": "fr.gael.gss.admin.ingester.payload.ProducerSwiftConfPayload",
    "containers": "test",
    "infiniteLoop": false,
    "credentials": "credential2"
  }
}
```

OData source with basic authentication:

For property **source.type** , we can have either "*dhus*" (for DHuS source) or "*csc*" for GSS or "cdse" (for CDSE source) according to the service we want to connect.

Note:

- The assumed format shall be set to ".zip" for DHuS and CDSE data sources.
- The assumed format should not be used for a GSS data source.

For OData authentication (**source.auth**), we can have "*basic*" or "*oauth2*" values for property `type`

```
{
  "name": "producerOdata",
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "my_topic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 0
  },
  "reprocess": false,
```

```
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
    "filter": ".*",
    "lastPublicationDate": "",
    "serviceRootUrl": "http://my.server.url",
    "type": "dhus",
    "assumedFormat": ".zip",
    "fetchQuicklook": true,
    "fetchAttributes": false,
    "useDateFromDb": false,
    "geoPostFilter": null,
    "auth": {
      "user": "user1",
      "password": "****",
      "type": "basic"
    }
  }
}
```


OData source with OAuth2 authentication:


```
{
  "name": "producerOdataOAuth2",
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "my_topic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 0
  },
  "reprocess": false,
  "dataSource": "Unknown",
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
    "filter": ".*",
    "lastPublicationDate": "",
    "serviceRootUrl": "http://my.server.url",
    "type": "dhus",
    "assumedFormat": ".zip",
    "fetchQuicklook": true,
    "fetchAttributes": false,
    "useDateFromDb": false,
    "geoPostFilter": "",
```

```
    "auth": {
      "user": "user1",
      "password": "****",
      "clientId": "clientIdForauthentication",
      "tokenEndpoint": "http://my.endpoint/auth",
      "type": "oauth2"
    }
  }
}
```


Response: Created producer (HTTP 200)

OData PRIP source with OAuth2 authentication:


```
{
  "name": "producerOdataPrip",
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "my_topic",
  "pushInterval": 60,
  "filter": ".*",
  "dataSource": "Prip",
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
    "filter": "not contains(Name, '.xml')",
    "lastPublicationDate": "2025-09-01T00:00:00Z",
    "serviceRootUrl": "http://my.server.url",
    "type": "csc",
    "fetchQuicklook": false,
    "fetchAttributes": false,
    "useDateFromDb": true
    "auth": {
      "user": "user1",
      "password": "****",
      "clientId": "clientIdForauthentication",
      "tokenEndpoint": "http://my.endpoint/auth",
      "type": "oauth2"
    }
  }
}
```

Response: Created producer (HTTP 200)

Note that the PRIP may serve xml products, so you may want to filter them out like in the above example.

Folder source:


```
{
  "name": "producer1",
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "exampleTopic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 1
  },
  "reprocess": false,
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerFolderConf",
    "path": "/path/for/source"
  }
}
```


Swift source:


```
{
  "name": "producerSwift",
  "hosts": "host-1",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "string",
  "pushInterval": 60,
  "filter": "string",
  "processError": {
    "active": true,
    "retries": 1
  },
  "reprocess": false,
  "dataSource": "Unknown",
  "source": {
    "sourceType": "fr.gael.gss.admin.ingester.payload.ProducerSwiftConfPayload",
    "containers": "test",
    "infiniteLoop": false,
    "credentials": "credential2"
```

```
  }
}
```

S3 source:

```
{
  "name": "producerS3,
  "hosts": "host-1",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "string",
  "pushInterval": 60,
  "filter": "string",
  "processError": {
    "active": true,
    "retries": 1
  },
  "reprocess": false,
  "dataSource": "Unknown",
  "source": {
    "sourceType": "fr.gael.gss.admin.ingester.payload.ProducerS3ConfPayload",
    "buckets": "test",
    "infiniteLoop": false,
    "credentials": "s3Credentials"
  }
}
```

S3 source for Zarr products:

```
{
  "name": "producerS3,
  "hosts": "host-1",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "string",
  "pushInterval": 60,
  "filter": "string",
  "processError": {
    "active": true,
    "retries": 1
  },
  "reprocess": false,
  "dataSource": "Unknown",
  "source": {
    "sourceType": "fr.gael.gss.admin.ingester.payload.ProducerS3ConfPayload",
```

```
    "buckets": "test",
    "infiniteLoop": false,
    "credentials": "s3Credentials",
    "productDirectories": true,
    "suffix": ".zarr"
  }
}
```

OData source with basic authentication:

```
{
  "name": "producerOdata",
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "my_topic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 0
  },
  "reprocess": false,
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
    "filter": ".*",
    "lastPublicationDate": "",
    "serviceRootUrl": "http://my.server.url",
    "type": "dhus",
    "assumedFormat": ".zip",
    "fetchQuicklook": true,
    "fetchAttributes": false,
    "useDateFromDb": false,
    "geoPostFilter": "",
    "auth": {
      "user": "user1",
      "password": "****",
      "type": "basic"
    }
  }
}
```

OData source with OAuth2 authentication:

```
{
  "name": "producerOdataOAuth2",
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "my_topic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 0
  },
  "reprocess": false,
  "dataSource": "Unknown",
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
    "filter": ".*",
    "lastPublicationDate": "",
    "serviceRootUrl": "http://my.server.url",
    "type": "dhus",
    "assumedFormat": ".zip",
    "fetchQuicklook": true,
    "fetchAttributes": false,
    "useDateFromDb": false,
    "geoPostFilter": "",
    "auth": {
      "user": "user1",
      "password": "****",
      "clientId": "clientIdForauthentication",
      "tokenEndpoint": "http://my.endpoint/auth",
      "type": "oauth2"
    }
  }
}
```


Error:

HTTP 400 - A mandatory field is missing, or the object is not complete.

HTTP 409 – A producer with the same name already exists.

HTTP 500 - Internal error. See the logs.

cURL:

```
curl -v -X POST 'https://<server.url>:<port>/producers' -d @producer.json -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

STAC source:

```
"source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerStacConf",
    "serviceRootUrl": "https://catalogue.dataspace.copernicus.eu/stac",
    "top": 100,
    "pivotDate": "datetime",
    "pivotDateValue": "2024-03-21T02:00:00.000Z",
    "filter": {
    "param": [
      {
        "name": "collections",
        "value": "SENTINEL-2"
      },
      {
        "name": "bbox",
        "value": "-80.673805,-0.52849,-78.060341,1.689651"
      }
     ]
    },
   "sortby": "+datetime",
   "assumedFormat": ".zip",
    "fetchQuicklook": true,
    "useDateFromDb": true
}
```

12.7.6 Update a Producer (PATCH)

Description: Update a producer by name.

NOTE: The user cannot change the type of the source for each producer.

HTTP Method: PATCH

URL: http://<server.url>:<server.port>/<context.path>/producers/<producer.name>

Optional param: restartIngesters=<true/false>
If true and an existing producer instance is running, restarts it.

Body: Folder source:

```
{
  "name": "producer1",
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
```

```
  "password": "optional.kafka.password",
  "topic": "exampleTopic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 1
  },
  "reprocess": false,
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerFolderConf",
    "path": "/path/for/source"
  }
}
```

Swift source:

```
{
  "name": "producerSwift",
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "string",
  "pushInterval": 60,
  "filter": "string",
  "processError": {
    "active": true,
    "retries": 1
  },
  "reprocess": false,
  "dataSource": "Unknown",
  "source": {
    "sourceType": "fr.gael.gss.admin.ingester.payload.ProducerSwiftConfPayload",
    "containers": "test",
    "infiniteLoop": false,
    "credentials": "credential2"
  }
}
```

OData source with basic authentication

```
{
  "name": "producerOdata",
  "hosts": "host1, host2",
```

```
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "my_topic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 0
  },
  "reprocess": false,
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
    "filter": ".*",
    "lastPublicationDate": "",
    "serviceRootUrl": "http://my.server.url",
    "type": "dhus",
    "assumedFormat": ".zip",
    "fetchQuicklook": true,
    "fetchAttributes": false,
    "useDateFromDb": false,
    "geoPostFilter": "",
    "auth": {
      "user": "user1",
      "password": "*****",
      "type": "basic"
    }
  }
}
```


Response: Update the producer (HTTP 200)

OData source with OAuth2 authentication:


```
{
  "name": "producerOdataOAuth2",
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "topic": "my_topic",
  "pushInterval": 60,
  "filter": ".*",
  "processError": {
    "active": true,
    "retries": 0
  },
```

```
  "reprocess": false,
  "dataSource": "Unknown",
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
    "filter": ".*",
    "lastPublicationDate": "",
    "serviceRootUrl": "http://my.server.url",
    "type": "dhus",
    "assumedFormat": ".zip",
    "fetchQuicklook": true,
    "fetchAttributes": false,
    "useDateFromDb": false,
    "geoPostFilter": "",
    "auth": {
      "user": "user1",
      "password": "****",
      "clientId": "clientIdForauthentication",
      "tokenEndpoint": "http://my.endpoint/auth",
      "type": "oauth2"
    }
  }
}
```


Error:

HTTP 404 - If producer does not exist.

HTTP 500 - Internal error. See the logs.

12.7.7 Delete a Producer (DELETE)

Description: Delete a producer by name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/producers/<producer.name>

Response: A confirmation of the deletion (HTTP 200)

Error:

HTTP 404 - If producer does not exist.

HTTP 500 - Internal error. See the logs.

cURL

```
curl -v -X DELETE -H "Authorization: Bearer ${AT}" -H "content-
type:application/json" "https://<server.url>:<port>/producers/<producer-
name>" | jq
```

12.8 Ingester Consumer API

The **sourceType** will have the following value:

Source Type	Description
fr.gael.gss.ingest.ingester.ConsumerFolderConf	Source folder
fr.gael.gss.admin.ingester.payload.ConsumerSwiftConfPayload	Source swift
fr.gael.gss.ingest.ingester.ConsumerOdataConf	Source Odata
fr.gael.gss.ingest.ingester.ConsumerStacConf	Source STAC
fr.gael.gss.admin.ingester.payload.ConsumerS3ConfPayload	Source S3

The property **type** for Ingestion task has the following value:

Task type	Description
fr.gael.gss.ingest.ingester.IngestInDataStores	Ingestion in datastore task
fr.gael.gss.ingest.ingester.ExtractMetadata	Extract metadatastore task
fr.gael.gss.ingest.ingester.IngestInMetadataStores	Ingestion in metadatastore task
fr.gael.gss.ingest.ingester.CreateQuicklook	Creation of quicklook task

NOTE: An error Manager can be either a Swift container, a S3 bucket, a folder or a Kafka topic. Credentials must exist if using Swift or S3.

Swift container:

```
"errorManager": {
  "container": "containerError",
  "type": "swift",
```

```
  "credentials": "credential1"
}
```

S3 bucket:

```
"errorManager": {
  "bucket": "bucket-error",
  "type": "s3",
  "credentials": "s3Credentials"
}
```

Folder:

```
"errorManager": {
  "errorLocation": "/path/to/store/products/in/error", // internal path in Docker compose file
  "type": "folder"
}
```

Kafka topic:

```
"errorManager": {
  "kafkaHosts": "kafka-host:9092",
  "kafkaTopic": "recovery",
  "kafkaUser": "optional.kafka.username",
  "kafkaPassword": "optional.kafka.password",
  "type": "kafka"
}
```

JSON Properties for consumers

Property	Type	Mandatory	Default value
hosts	String	true	
topics	String	true	
user	String		
password	String		
pollIntervalMs	int		1200000
filter	String		
parallelIngests	int		4
groupId	String	true	
topicPattern	String		
reprocess	boolean		false
sourceDelete	object		{ "value": false,

			"pattern": ".*" }
ingestThreads	int		1
tasks		true	
tasks.tmpPath	String		
source		true	

Ingest in Datastore task properties

Property	Type	Mandatory	Default value
pattern	String		
stopOnFailure	boolean		
tryLimit	int		
active	boolean		
targetStores	String		

Ingest in Metadatastore task properties

Property	Type	Mandatory	Default value
pattern	String		
stopOnFailure	boolean		
tryLimit	int		
active	boolean		
targetStores	String		

Extract Metadata task properties

Property	Type	Mandatory	Default value
pattern	String		
stopOnFailure	boolean		
tryLimit	int		
active	boolean		
forceOnline	boolean		

Create quicklook task properties

Property	Type	Mandatory	Default value
pattern	String		
stopOnFailure	boolean		
tryLimit	int		
active	boolean		
onlyUseProvidedQL	boolean		

height	int		
width	int		
targetStores	String	true	

Folder source

Property	Type	Mandatory	Default value
path	String	true	refers to the internal path configured in Docker compose file

Swift source

Property	Type	Mandatory	Default value
credentials	String	true	
containers	String	true	

S3 source

Property	Type	Mandatory	Default value
credentials	String	true	
buckets	String	true	

OData source

Property	Type	Mandatory	Default value
serviceRootUrl	String	true	
type	String		csc Other values: dhus, cdse
retriesOn429	int		0
retryWaitOn429Ms	int		100
auth.user	String	true	
auth.password	String	true	
auth.clientId	String		
auth.tokenEndpoint	String		
auth.type	String	true	Values: basic or oauth2

STAC source

Property	Type	Mandatory	Default value
serviceRootUrl	String	true	
retriesOn429	int		0
retryWaitOn429Ms	int		100
auth.user	String	true	

auth.password	String	true	
auth.clientId	String		
auth.tokenEndpoint	String		
auth.type	String	true	Values: basic or oauth2

WARNING: In the current implementation, all the default values will be applied if the property is not present.

12.8.1 List of all Consumers (GET)

Description: Get a list of all available consumers or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/consumers

NOTE: We can also use the following parameters for the search:

- **sourceType:** to retrieve all the consumer with a specific type. Its values are FOLDER, SWIFT, S3, ODATA, STAC
- **top:** to retrieve a specific number or consumers
- **skip:** to skip some objects in the response

We obtained the URL
http://<server.url>:<server.port>/<context.path>/consumers?sourceType=<source.type>&top=<top.value>&skip=<skip.value>

Response: A list of all consumers or an empty list (HTTP 200)

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-
type:application/json" "https://<server.url>:<port>/consumers" | jq
```

12.8.2 Get a Consumer (GET)

Description: Get a consumer by name.

HTTP Method: GET

URL: http://<server.url>:<server.port>/<context.path>/consumers/<consumer.name>

Response: A consumer with the given name (HTTP 200)

Error:

HTTP 404 - If consumer does not exist.

HTTP 500 - Internal error. See the logs.

cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/consumers/<consumer-name>" | jq
```

12.8.3 Create a Consumer (POST)

Description: Create a consumer.

HTTP Method: POST

URL: http://<server.url>:<server.port>/<context.path>/consumers

RECOMMENDATION

You can find JSON configuration sample files in the cdh-ingest-3.4.0.zip file. These files can be used for the consumer configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Consumer**" directory.

The "JSON-samples/Consumer" directory contains operational scenarios for the following sources: CDSE, DHuS, Folder, Swift, S3 and GSS.

Body: Folder source

```
{
  "name": "consumerFolder1",
  "parallelIngests": 4,
  "hosts": "hots1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "group-consumer",
  "topics": "topic-ingestion-S2,topic-ingestion-S3",
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "errorManager": {
    "errorLocation": "/path/to/error",
    "type": "folder"
  },
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ConsumerFolderConf",
```

```
    "path": "/path/to/consumer"
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "forceOnline": false
    },
    {
      "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
      "pattern": ".*",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
      "onlyUsedProvidedQL": true,
      "height": 45,
      "width": 45,
      "targetStores": "HFS1"
    }
  ],
  "tmpPath": "/path/to/tmp/for/tasks"  // internal path in Docker compose file
}
```

Response: The created consumer (HTTP 200)

Body: Swift source

```
{
  "name": "consumerSwift",
  "parallelIngests": 4,
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_group_consumer",
  "topics": "topic1,topic2",
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "errorManager": {
    "errorLocation": "/path/to/store/products/in/error", // internal path in Docker compose file
  "type": "folder"
  },
  "source": {
    "sourceType": "fr.gael.gss.admin.ingester.payload.ConsumerSwiftConfPayload",
    "containers": "toto",
    "credentials": "credentials2"
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
```

```
        "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "forceOnline": false
      },
      {
        "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
        "pattern": ".*",
        "stopOnFailure": false,
        "tryLimit": 0,
        "active": true,
        "onlyUsedProvidedQL": true,
        "height": 45,
        "width": 45,
        "targetStores": "HFS1"
      }
    ],
    "tmpPath": "/path/to/tmp/for/tasks"

}
```

Response: The created consumer (HTTP 200)

Body: S3 source

```
{
  "name": "consumerS3",
  "parallelIngests": 4,
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_group_consumer",
  "topics": "topic1,topic2",
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "errorManager": {
    "errorLocation": "/path/to/store/products/in/error", // internal path in Docker compose file
  "type": "folder"
  },
  "source": {
```

```
    "sourceType": " fr.gael.gss.admin.ingester.payload.ConsumerS3ConfPayload",
    "buckets": "toto",
    "credentials": "credentialsS3"
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "forceOnline": false
    },
    {
      "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
      "pattern": ".*",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
      "onlyUsedProvidedQL": true,
      "height": 45,
      "width": 45,
      "targetStores": "HFS1"
    }
  ],
  "tmpPath": "/path/to/tmp/for/tasks"

}
```

Response: The created consumer (HTTP 200)

Note that for Zarr ingestion there is no quicklook extraction task. Moreover a size of 0 will be displayed for zarr products, it's a normal behavior as these products are directories.

Body: OData source with basic authentication

```
{
  "name": "consumerOdata",
  "parallelIngests": 4,
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_consumer_group",
  "topics": "topic1",
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "errorManager": {
    "errorLocation": "/path/to/store/products/in/error", // internal path in Docker compose file
  "type": "folder"
  },
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
    "type": "dhus",
    "serviceRootUrl": "https://colhub.copernicus.eu/dhus",
    "retriesOn429": 2,
    "retryWaitOn429Ms": 100,
    "auth": {
      "type": "basic",
      "user": "user_name",
      "password": "****"
    }
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
```

```
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "forceOnline": false
    },
    {
      "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
      "pattern": ".*",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
      "onlyUsedProvidedQL": true,
      "height": 45,
      "width": 45,
      "targetStores": "HFS1"
    }
  ],
  "tmpPath": "/path/to/tmp/for/tasks" // internal path in Docker compose file
}
```

Response: The created consumer (HTTP 200)

Body: OData source with OAuth2 authentication

```
{
  "name": "consumerOdataOAuth2",
  "parallelIngests": 4,
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_consumer_group",
  "topics": "topic1,topic3",
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
```

```
    "value": false,
    "pattern": ".*"
  },
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
    "type": "cdse",
    "serviceRootUrl": "https://catalogue.dataspace.copernicus.eu/odata/v1",
    "retriesOn429": 2,
    "retryWaitOn429Ms": 100,
    "auth": {
      "type": "oauth2",
      "user": "user_name",
      "password": "****",
      "clientId": "clientIdForauthentication",
      "tokenEndpoint": "http://my.endpoint/auth"
    }
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "forceOnline": false
    },
    {
      "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
      "pattern": ".*",
      "stopOnFailure": false,
```

```
        "tryLimit": 0,
        "active": true,
        "onlyUsedProvidedQL": true,
        "height": 45,
        "width": 45,
        "targetStores": "HFS1"
      }
    ],
    "tmpPath": "/path/to/tmp/for/tasks"  // internal path in Docker compose file
}
```

Response: The created consumer (HTTP 200)

Body: STAC source with OAuth2 authentication

```
{
    "name": "consumerStac",
    "parallelIngests": 4,
    "hosts": "host1,host2",
    "user": "optional.kafka.username",
    "password": "optional.kafka.password",
    "groupId": "my_consumer_group",
    "topics": "topic1,topic3",
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": {
      "value": false,
      "pattern": ".*"
    },
    "source": {
      "sourceType": "fr.gael.gss.ingest.ingester.ConsumerStacConf",
       "serviceRootUrl": "https://catalogue.dataspace.copernicus.eu/stac/",
      "retriesOn429": 2,
      "retryWaitOn429Ms": 100,
      "auth": {
        "type": "oauth2",
        "user": "user_name",
        "password": "****",
        "clientId": "clientIdForauthentication",
        "tokenEndpoint": "http://my.endpoint/auth"
      }
    },
    "taskList": [
      {
        "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
```

```
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "targetStores": "HFS1"
      },
      {
        "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "targetStores": "solr"
      },
      {
        "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "forceOnline": false
      },
      {
        "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
        "pattern": ".*",
        "stopOnFailure": false,
        "tryLimit": 0,
        "active": true,
        "onlyUsedProvidedQL": true,
        "height": 45,
        "width": 45,
        "targetStores": "HFS1"
      }
    ],
    "tmpPath": "/path/to/tmp/for/tasks"  // internal path in Docker compose file
}
```

Response: The created consumer (HTTP 200)

Body: Folder source

```
{
  "name": "consumerFolder1",
  "parallelIngests": 4,
  "user": "optional.kafka.username",
```

```
"password": "optional.kafka.password",
"hosts": "hots1,host2",
"groupId": "group-consumer",
"topics": "topic-ingestion-S2,topic-ingestion-S3",
"topicPattern": null,
"reprocess": false,
"pollIntervalMs": 1200000,
"sourceDelete": {
  "value": false,
  "pattern": ".*"
},
"errorManager": {
  "errorLocation": "/path/to/error",  // internal path in Docker compose file
"type": "folder"
},
"source": {
  "sourceType": "fr.gael.gss.ingest.ingester.ConsumerFolderConf",
  "path": "/path/to/consumer"
},
"taskList": [
  {
    "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
    "pattern": ".*",
    "stopOnFailure": true,
    "tryLimit": 0,
    "active": true,
    "targetStores": "HFS1"
  },
  {
    "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
    "pattern": ".*",
    "stopOnFailure": true,
    "tryLimit": 0,
    "active": true,
    "targetStores": "solr"
  },
  {
    "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
    "pattern": ".*",
    "stopOnFailure": true,
    "tryLimit": 0,
    "active": true,
    "forceOnline": false
  },
  {
    "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
```

```
        "pattern": ".*",
        "stopOnFailure": false,
        "tryLimit": 0,
        "active": true,
        "onlyUsedProvidedQL": true,
        "height": 45,
        "width": 45,
        "targetStores": "HFS1"
      }
  ],
  "tmpPath": "/path/to/tmp/for/tasks" // internal path in Docker compose file
}
```

Response: The created consumer (HTTP 200)

Swift source:

```
{
  "name": "consumerSwift",
  "parallelIngests": 4,
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_group_consumer",
  "topics": "topic1,topic2",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "source": {
    "sourceType": "fr.gael.gss.admin.ingester.payload.ConsumerSwiftConfPayload",
    "containers": "toto",
    "credentials": "credentials2"
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
```

```
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "forceOnline": false
    },
    {
      "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
      "pattern": ".*",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
      "onlyUseProvidedQL": false,
      "height": 45,
      "width": 45,
      "targetStores": "HFS1"
    }
  ],
  "tmpPath": "/path/to/tmp/for/tasks", // internal path in Docker compose file
  "errorManager": {
  errorLocation": "/path/to/store/products/in/error", // internal path in Docker compose file
   "container": null,
     "type": "folder",
     "credentials": null
  }
}
```

Response: The created consumer (HTTP 200)

12.8.4 Update a Consumer (PATCH)

Description: Update a consumer by name.

HTTP Method: PATCH

URL: http://<server.url>:<server.port>/<context.path>/consumers/<consumer.name>

Optional param: restartIngesters=<true/false>
If true and an existing consumer instance is running, restarts it.

NOTE: User cannot change the type of the source for each consumer. The new taskList, if defined, will replace the existing one.

Body: Folder source

```
{
  "name": "consumerFolder1",
  "parallelIngests": 4,
  "hosts": "hots1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "group-consumer",
  "topics": "topic-ingestion-S2,topic-ingestion-S3",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "errorManager": {
    "errorLocation": "/path/to/error",  // internal path in Docker compose file
  "type": "folder"
  },
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ConsumerFolderConf",
    "path": "/path/to/consumer"
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
```

```
        "targetStores": "solr"
      },
      {
        "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "forceOnline": false
      },
      {
        "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
        "pattern": ".*",
        "stopOnFailure": false,
        "tryLimit": 0,
        "active": true,
        "onlyUsedProvidedQL": true,
        "height": 45,
        "width": 45,
        "targetStores": "HFS1"
      }
    ],
    "tmpPath": "/path/to/tmp/for/tasks" // internal path in Docker compose file
}
```

Response: The updated consumer (HTTP 200)

Swift source:

```
{
  "name": "consumerSwift",
  "parallelIngests": 4,
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_group_consumer",
  "topics": "topic1,topic2",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "source": {
    "sourceType": "fr.gael.gss.admin.ingester.payload.ConsumerSwiftConfPayload",
```

```
    "containers": "toto",
    "credentials": "credentials2"
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "forceOnline": false
    },
    {
      "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
      "pattern": ".*",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
      "onlyUseProvidedQL": false,
      "height": 45,
      "width": 45,
      "targetStores": "HFS1"
    }
  ],
  "tmpPath": "/path/to/tmp/for/tasks", // internal path in Docker compose file
 "errorManager": {
    "errorLocation": "/path/to/store/products/in/error", // internal path in Docker compose file
  "container": null,
    "type": "folder",
    "credentials": null
```

```
  }
}
```

Response: The updated consumer (HTTP 200)

S3 source:

```
{
  "name": "consumerS3",
  "parallelIngests": 4,
  "hosts": "host1, host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_group_consumer",
  "topics": "topic1,topic2",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
   "sourceDelete": {
     "value": false,
     "pattern": ".*"
  },
  "source": {
     "sourceType": "fr.gael.gss.admin.ingester.payload.ConsumerS3ConfPayload",
     "buckets": "toto",
     "credentials": "credentialsS3"
  },
  "taskList": [
     {
        "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "targetStores": "HFS1"
     },
     {
        "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "targetStores": "solr"
     },
     {
        "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
```

```
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "forceOnline": false
      },
      {
        "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
        "pattern": ".*",
        "stopOnFailure": false,
        "tryLimit": 0,
        "active": true,
        "onlyUseProvidedQL": false,
        "height": 45,
        "width": 45,
        "targetStores": "HFS1"
      }
    ],
    "tmpPath": "/path/to/tmp/for/tasks", // internal path in Docker compose file
  "errorManager": {
      "errorLocation": "/path/to/store/products/in/error", // internal path in Docker compose file
    "container": null,
      "type": "folder",
      "credentials": null
    }
}
```

Response: The updated consumer (HTTP 200)

OData source with basic authentication:

```
{
  "name": "consumerOdata",
  "parallelIngests": 4,
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_consumer_group",
  "topics": "topic1",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
```

```
  },
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
    "serviceRootUrl": "https://colhub.copernicus.eu/dhus",
    "auth": {
      "user": "user_name",
      "password": "****",
      "clientId": null,
      "tokenEndpoint": null,
      "type": "basic"
    },
    "type": "dhus",
    "retriesOn429": 2,
    "retryWaitOn429Ms": 100
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "forceOnline": false
    },
    {
      "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
      "pattern": ".*",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
```

```
        "onlyUseProvidedQL": false,
        "height": 45,
        "width": 45,
        "targetStores": "HFS1"
      }
  ],
  "tmpPath": "/path/to/tmp/for/tasks",  // internal path in Docker compose file
 "errorManager": {
    "errorLocation": "/path/to/store/products/in/error", // internal path in Docker compose file
  "container": null,
    "type": "folder",
    "credentials": null
  }
}
```

Response: The updated consumer (HTTP 200)

OData source with oauth2 authentication:

```
{
  "name": "consumerOdataOAuth2",
  "parallelIngests": 4,
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_consumer_group",
  "topics": "topic1,topic3",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
    "serviceRootUrl": "https://catalogue.dataspace.copernicus.eu/odata/v1/",
    "auth": {
      "user": "user_name",
      "password": "****",
      "clientId": "clientIdForauthentication",
      "tokenEndpoint": "http://my.endpoint/auth",
      "type": "oauth2"
    },
    "type": "dhus",
    "retriesOn429": 2,
```

```
    "retryWaitOn429Ms": 100
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "solr"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "forceOnline": false
    },
    {
      "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
      "pattern": ".*",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
      "onlyUseProvidedQL": false,
      "height": 45,
      "width": 45,
      "targetStores": "HFS1"
    }
  ],
   "tmpPath": "/path/to/tmp/for/tasks",  // internal path in Docker compose file
  "errorManager": null
}
```

Error:

HTTP 404 - If consumer does not exist.

HTTP 500 - Internal error. See the logs.

STAC source with oauth2 authentication:

```
{
  "name": "consumerStac",
  "parallelIngests": 4,
  "hosts": "host1,host2",
  "user": "optional.kafka.username",
  "password": "optional.kafka.password",
  "groupId": "my_consumer_group",
  "topics": "topic1,topic3",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": {
    "value": false,
    "pattern": ".*"
  },
  "source": {
    "sourceType": "fr.gael.gss.ingest.ingester.ConsumerStacConf",
    "serviceRootUrl": "https://catalogue.dataspace.copernicus.eu/stac/",
    "auth": {
      "user": "user_name",
      "password": "****",
      "clientId": "clientIdForauthentication",
      "tokenEndpoint": "http://my.endpoint/auth",
      "type": "oauth2"
    },
    "retriesOn429": 2,
    "retryWaitOn429Ms": 100
  },
  "taskList": [
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
      "pattern": ".*",
      "stopOnFailure": true,
      "tryLimit": 0,
      "active": true,
      "targetStores": "HFS1"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
      "pattern": ".*",
```

```
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "targetStores": "solr"
      },
      {
        "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
        "pattern": ".*",
        "stopOnFailure": true,
        "tryLimit": 0,
        "active": true,
        "forceOnline": false
      },
      {
        "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
        "pattern": ".*",
        "stopOnFailure": false,
        "tryLimit": 0,
        "active": true,
        "onlyUseProvidedQL": false,
        "height": 45,
        "width": 45,
        "targetStores": "HFS1"
      }
    ],
    "tmpPath": "/path/to/tmp/for/tasks",  // internal path in Docker compose file
  "errorManager": null
}
```

Error:

HTTP 404 - If consumer does not exist.

HTTP 500 - Internal error. See the logs.

12.8.5 Delete a Consumer (DELETE)

Description: Delete a consumer by name.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/<context.path>/consumers/<consumer.name>

Response: Confirmation of (HTTP 200)

Error:

HTTP 404 - If consumer does not exist.

HTTP 500 - Internal error. See the logs.

cURL
`curl -v -X DELETE -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/consumers/<consumer-name>" | jq`

12.9 Deletion Job

This object represents the deletion job which deletes products in the stores (metadatastores and datastores)

RECOMMENDATION
You can find JSON configuration sample files in the cdh-admin-api-3.4.0.zip file. These files can be used for deletion configuration. They fit the main operational scenarios known. The files are stored in the "**JSON-samples/Deletion**" directory.

12.9.1 Schema

Property	Type	Mandatory	Description / Default value
jobId	uuid	yes	autogenerated
name	String		A name to keep for history
odataFilter	String		Filter for products in stores
notificationMessage	String		A message to send to notified users
reason	**Enum** CORRUPTED_PRODUCT WRONG_CHECKSUM DUPLICATED_PRODUCT OBSOLETE_PRODUCT	yes	• CORRUPTED_PRODUCT: A product that is damaged or incomplete • WRONG_CHECKSUM: An incorrect checksum, indicates a corrupt or tampered product • DUPLICATED_PRODUCT: Same or identical product • OBSOLETE_PRODUCT: A product that is no longer available or relevant
status	**Enum** CREATED RUNNING PAUSED	yes	Default status: CREATED CREATED: initial deletion job status RUNNING: to execute a deletion job

	DONE CANCELLED		PAUSED: to pause a running job DONE: a job execution completion CANCELLED: to cancel a previously created job
creationDate	Timestamp		
updatedDate	Timestamp		
errors	int		During deletion, the number of products in error
nbProducts	int		Number of products to delete / deleted
nbThreads	int		1 Number of threads to use for the job

12.9.2 Create a Deletion Job (POST)

Description: Create a deletion job.

HTTP Method: POST

URL: http://<server.url>:<server.port>/jobs

Body:

```
{
  "name": "My testing",
  "reason": "CORRUPTED_PRODUCT",
  "status": "CREATED",
  "notificationMessage": "Products are corrupted",
  "odataFilter": "$filter=contains(Name,'R142')",
  "nbThreads": 2
}
```

Response:

```
{
  "name": "My testing",
  "reason": "CORRUPTED_PRODUCT",
  "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "notificationMessage": "Products are corrupted",
  "errors": 0,
  "nbProducts": 3,
  "creationDate": "2023-11-12T19:00:48.235Z",
  "updatedDate": null,
  "status": "CREATED",
  "odataFilter": "$filter=contains(Name,'R142')",
  "nbThreads": 2
}
```

cURL

```
curl -v -X POST ''https://<server.url>:<port>/jobs' -d @dletion_job.json -H
'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

12.9.3 Dry run a Deletion Job (POST)

Description: Dryrun a deletion job without actual execution.

HTTP Method: POST

URL: http://<server.url>:<server.port>/jobs/<id>/$dryrun

Response:


```
{
  "name": "My testing",
  "reason": "CORRUPTED_PRODUCT",
  "notificationMessage": "Products are corrupted",
  "jobId": null,
  "errors": 0,
  "nbProducts": 21,
  "creationDate": null,
  "updatedDate": null,
  "status": null,
  "odataFilter": "$filter=contains(Name,'R142')",
  "nbThreads": 2
}
```


cURL

```
curl -v -X POST 'https://<server.url>:<port>/jobs/<job-id>/$dryrun' -H
'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

12.9.4 List of all Deletion Jobs (GET)

Description: Get a list of all available deletion jobs or return an empty list.

HTTP Method: GET

URL: http://<server.url>:<server.port>/jobs

Response:


```
[
  {
    "name": "job 1",
    "reason": "CORRUPTED_PRODUCT",
    "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
```

```
    "notificationMessage": "Products are corrupted",
    "errors": 0,
    "nbProducts": 3,
    "creationDate": "2023-11-12T19:00:48.235Z",
    "updatedDate": null,
    "status": "RUNNING",
    "odataFilter": "$filter=contains(Name,'R142')",
    "nbThreads": 2
  },
  {
    "name": "job 2”,
    "reason": "CORRUPTED_PRODUCT",
    "jobId": "3fa85f64-5717-4563-b3fc-2c963f66afa6",
    "notificationMessage": "Products are corrupted again:(",
    "errors": 0,
    "nbProducts": 3,
    "creationDate": "2023-11-12T19:00:48.235Z",
    "updatedDate": null,
    "status": "CREATED",
    "odataFilter": "$filter=contains(Name,'R142')",
    "nbThreads": 2
  }
]
```


cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-
type:application/json" "https://<server.url>:<port>/jobs" | jq
```

12.9.5 Get a Deletion Job (GET)

Description: Get a particular deletion job by job id.

HTTP Method: GET

URL: http://<server.url>:<server.port>/jobs/<id>

Response:


```
{
  "name": "My testing",
  "reason": "CORRUPTED_PRODUCT",
  "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "notificationMessage": "Products are corrupted",
  "errors": 0,
```

```
  "nbProducts": 0,
  "creationDate": "2023-11-12T19:00:48.235Z",
  "updatedDate": null,
  "status": "DONE",                    // job already executed status
  "odataFilter": "$filter=contains(Name,'R142')",
  "nbThreads": 2
}
```


cURL

```
curl -v -X GET -H "Authorization: Bearer ${AT}" -H "content-type:application/json" "https://<server.url>:<port>/jobs/<job-id>" | jq
```

12.9.6 Cancel a Deletion Job (POST)

Description: Cancel a deletion job by id.

HTTP Method: POST

URL: http://<server.url>:<server.port>/jobs/<id>/cancel

NOTE: If a deletion job is cancelled:

- We cannot resume a cancelled job.
- We cannot run a cancelled job.
- We cannot pause a cancelled job.

Response:


```
{
  "name": "My testing",
  "reason": "CORRUPTED_PRODUCT",
  "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "notificationMessage": "Products are corrupted",
  "errors": 0,
  "nbProducts": 0,
  "creationDate": "2023-11-12T19:00:48.235Z",
  "updatedDate": "2023-11-13T01:08:38.760061Z",
  "status": "CANCELED",
  "odataFilter": "$filter=contains(Name,'R142')",
  "nbThreads": 2
}
```


cURL

```
curl -v -X POST 'https://<server.url>:<port>/jobs/<job-id>/cancel' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

12.9.7 Run a Deletion Job (POST)

Description: Run and begin the deletion process by job id.

HTTP Method: POST

URL: http://<server.url>:<server.port>/jobs/<id>/run

Response:


```
{
  "name": "My testing",
  "reason": "CORRUPTED_PRODUCT",
  "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "notificationMessage": "Products are corrupted",
  "errors": 0,
  "nbProducts": 0,
  "creationDate": "2023-11-12T19:00:48.235Z",
  "updatedDate": "2023-11-13T01:20:39.542541Z",
  "status": "RUNNING",
  "odataFilter": "$filter=contains(Name,'R142')",
  "nbThreads": 2
}
```


Note: The number of maxSkip [defined while creating the metadata store] should be greater than the number of products needing deletion.

cURL

```
curl -v -X POST 'https://<server.url>:<port>/jobs/<job-id>/run' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

12.9.8 Resume a Deletion Job (POST)

Description: Resume a paused deletion job by id.

HTTP Method: POST

URL: http://<server.url>:<server.port>/jobs/<id>/resume

NOTE: If a deletion job has the resume status:

- We can pause a resume job.
- We can cancel a resume job.

Response:


```
{
  "name": "My testing",
  "reason": "CORRUPTED_PRODUCT",
  "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "notificationMessage": "Products are corrupted",
  "errors": 0,
  "nbProducts": 0,
  "creationDate": "2023-11-12T19:00:48.235Z",
  "updatedDate": "2023-11-13T01:08:38.760061Z",
  "status": "RUNNING",
  "odataFilter": "$filter=contains(Name,'R142')",
  "nbThreads": 2
}
```


cURL

```
curl -v -X POST 'https://<server.url>:<port>/jobs/<job-id>/resume' -H
'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

12.9.9 Pause a Deletion Job (POST)

Description: Pause an existing running deletion job by id.

HTTP Method: POST

URL: http://<server.url>:<server.port>/jobs/<id>/pause

NOTE: If a deletion job has the pause status:

- We can resume a pause job.
- We can run a pause job.
- We can cancel a pause job.

Response:


```
{
  "name": "My testing",
  "reason": "CORRUPTED_PRODUCT",
  "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "notificationMessage": "Products are corrupted",
  "errors": 0,
  "nbProducts": 0,
  "creationDate": "2023-11-12T19:00:48.235Z",
  "updatedDate": "2023-11-13T01:08:38.760061Z",
```

```
  "status": "PAUSED",
  "odataFilter": "$filter=contains(Name,'R142')",
  "nbThreads": 2
}
```


cURL

```
curl -v -X POST 'https://<server.url>:<port>/jobs/<job-id>/pause' -H
'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

12.9.10 Delete a Deletion Job (DELETE)

Description: Delete a deletion job by job id.

HTTP Method: DELETE

URL: http://<server.url>:<server.port>/jobs/<id>

Response:


```
{
  "message": "Delete job 3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "details": ""
}
```


NOTE:

If the job does not exist, we will have a HTTP 404 error.

cURL

```
curl -v -X DELETE -H "Authorization: Bearer ${AT}" -H "content-
type:application/json" "https://<server.url>:<port>/jobs/<job-id>" | jq
```

12.10 Eviction Process

Since version 2.1.1, the eviction process can be managed in the Admin API. The eviction can be automatically launched on startup with the property **process.evictionByTime=true** in the *application.properties* file. In this case the **process.evictionByTime=true** parameter should be set as false in the Catalogue.

The eviction process will use DataStores configured in database to manage products eviction.

12.10.1 Eviction Endpoint (GET)

Description: Monitor the eviction process.

HTTP Method: GET

URL: http://<server.url>:<server.port>/eviction

Response:


```
{
  "status": "RUNNING",
  "previousEvictions": [
    {
      "type": "EVICTED",
      "product": {
        "productId": "b7a05f77-3e5f-4cf0-8b59-081ba59c15b3",
        "productName": "S2A_MSIL1C_20181005T154111_N0500_R011_T18TWL_20230624T144249.zip",
        "storeName": "timeGroup",
        "source": "swiftTime",
        "evictionDate": "2024-09-04T11:07:25.517Z",
        "keepPeriod": 864000,
        "processing": false,
        "productReference": {
          "id": "b7a05f77-3e5f-4cf0-8b59-081ba59c15b3",
          "name": "S2A_MSIL1C_20181005T154111_N0500_R011_T18TWL_20230624T144249.zip"
        }
      },
      "productEvicted": true,
      "referenceEvicted": true,
      "attacheFilesEvicted": true,
      "eventDate": "2024-09-05T08:21:31.294908576Z",
      "errorMessage": null
    }
  ],
  "nextEvictions": [
    {
      "productId": "e3fcf341-88c1-47c8-bb5d-3a4d706118ce",
      "productName": "S2A_MSIL1C_20181007T075811_N0500_R035_T35KMQ_20230728T185340.zip",
      "storeName": "timeGroup",
      "source": "swiftTime",
      "evictionDate": "2024-09-13T11:08:16.637839Z",
      "keepPeriod": 864000,
      "processing": false,
      "productReference": {
        "id": "e3fcf341-88c1-47c8-bb5d-3a4d706118ce",
        "name": "S2A_MSIL1C_20181007T075811_N0500_R035_T35KMQ_20230728T185340.zip"
      }
    },
    {
      "productId": "088749c7-6dde-440f-b8c7-8c24e2a7078a",
      "productName": "S2A_MSIL1C_20181007T075811_N0500_R035_T35JLK_20230728T185340.zip",
```

```
            "storeName": "timeGroup",
            "source": "swiftTime",
            "evictionDate": "2024-09-13T11:09:21.100119Z",
            "keepPeriod": 864000,
            "processing": false,
            "productReference": {
               "id": "088749c7-6dde-440f-b8c7-8c24e2a7078a",
               "name": "S2A_MSIL1C_20181007T075811_N0500_R035_T35JLK_20230728T185340.zip"
            }
         }
      ],
      "evictionPolicies": [
         {
            "storeGroupName": "timeGroup",
            "storeName": "swiftTime",
            "eviction": true,
            "keepPeriodSeconds": 864000,
            "evictReference": true,
            "evictAttachedFiles": true
         }
      ]
   }
```


cURL

```
curl -X GET -H "Authorization: Bearer ${AT}" -H "content-
type:application/json" "https://<server.url>:<port>/eviction"
```

The response contains the following fields:

Field	Description
status	Eviction process status, can be RUNNING or STOPPED.
previousEvictions	An array of last evicted products (1000 max.).
nextEvictions	An array of next scheduled evictions (1000 max.).
evictionPolicies	An array of all existing policies fetched from database.

12.10.2 Start Eviction (POST)

Description: Start the eviction process.

HTTP Method: POST

URL: http://<server.url>:<server.port>/eviction?action=start

Response: 200 if eviction has started, 400 if the action cannot be executed

12.10.3 Stop Eviction (POST)

Description: Stop the eviction process.

HTTP Method: POST

URL: http://<server.url>:<server.port>/eviction?action=stop

Response: 200 if eviction has stopped, 400 if the action cannot be executed

12.11 Ingesters management

It's possible to launch/stop and monitor producers and consumers on the Admin API. The ingesters will run on the same JVM as the Admin API. Each ingester is identified by a unique name, previously created on the producers or consumers endpoint of the Admin API.

Ingester states - An ingester can be in one of the following states:

State	Description
STARTING	A start action has been received and the ingester is starting.
RUNNING	The ingester is running.
STOPPING	A stop action has been received and the ingester is stopping. An ingester can stay in this state for several minutes, waiting for current ingestions to end.
SHUTDOWN	The ingester is stopped.
FAILED	The ingester is failed. It can happen if an error occurred after a start or stop action. The **lastMessage** field of the ingester will be updated with the exception message.

12.11.1 Start an Ingester (POST)

Description: Start a producer or a consumer.

HTTP Method: POST

URL: http://<server.url>:<server.port>/ingesters-management/<ingester-name>?action=start

Response:


```
{
  "name": "cdse-p",
  "events": [],
  "counters": {},
  "state": "RUNNING",
  "type": "PRODUCER",
  "updatedDate": "2024-09-12T11:16:00.882383720Z",
  "lastMessage": "Ingester cdse-p is running"
}
```


12.11.2 Stop an ingester (POST)

Description: Stop a producer or a consumer.

HTTP Method: POST

URL: http://<server.url>:<server.port>/ingesters-management/<ingester-name>?action=stop

Response:


```
{
  "name": "cdse-p",
  "events": [
    {
      "status": "QUEUED",
      "productId": "834be3f0-bbcf-4c87-9ea0-1e81f6f8c081",
      "productName": "S2B_MSIL2A_20231012T085959_N0509_R121_T29CNR_20231012T112256.zip",
      "date": "2024-09-12T11:40:44.011426595Z"
    },
    {
      "status": "QUEUED",
      "productId": "83d3481e-17d5-4a47-a782-8d63b00b6754",
      "productName": "S2B_MSIL2A_20200309T073729_N0500_R092_T35JQK_20230628T180704.zip",
      "date": "2024-09-12T11:40:44.016454939Z"
    }
```

```
  ],
  "counters": {
    "QUEUED": 2
  },
  "state": "STOPPING",
  "type": "PRODUCER",
  "updatedDate": "2024-09-12T11:40:44.566374651Z",
  "lastMessage": "Ingester cdse-p is running"
}
```


12.11.3 Get an ingester (GET)

Description: Get a producer or a consumer.

HTTP Method: GET

URL: http://<server.url>:<server.port>/ingesters-management/<ingester-name>

Response:


```
{
  "name": "cdse-c",
  "events": [
    {
      "status": "INGESTED",
      "productId": "f03176fd-dfb6-4e0a-b5b4-30c6c39ea688",
      "productName": "S2A_MSIL2A_20200319T032531_N0500_R018_T47NQF_20230628T105144.zip",
      "date": "2024-09-12T12:05:18.998506776Z"
    }
  ],
  "counters": {
    "INGESTED": 1
  },
  "state": "RUNNING",
  "type": "CONSUMER",
  "updatedDate": "2024-09-12T12:05:18.998520192Z",
  "lastMessage": "cdse-c"
}
```


12.11.4 Filter all Ingesters (GET)

Description: Filter/search ingesters.

HTTP Method: GET

URL: http://<server.url>:<server.port>/ingesters-management

Optional params: they can be added with a "?" and separated by a "&".
Example: http://<server.url>:<server.port>/ingesters-management?type=PRODUCER&top=10

Parameter	Description	Example
name	Name of the ingester	name=cdse-p
type	Type of the ingesters. Can be PRODUCER or CONSUMER	type=CONSUMER
state	State of the ingesters. • Please refer to Section 12.10 to Ingester States table	state=RUNNING
top	Limit the number of results.	top=10
skip	Skip some results	skip=5

Response:


```
{
  "count": 2,
  "instances": [
    {
      "name": "cdse-p",
      "events": [
        {
          "status": "QUEUED",
          "productId": "834be3f0-bbcf-4c87-9ea0-1e81f6f8c081",
          "productName": "S2B_MSIL2A_20231012T085959_N0509_R121_T29CNR_20231012T112256.zip",
          "date": "2024-09-12T11:40:44.011426595Z"
        },
        {
          "status": "QUEUED",
          "productId": "83d3481e-17d5-4a47-a782-8d63b00b6754",
          "productName": "S2B_MSIL2A_20200309T073729_N0500_R092_T35JQK_20230628T180704.zip",
          "date": "2024-09-12T11:40:44.016454939Z"
        },
        {
```

```
            "status": "QUEUED",
            "productId": "ab2d643a-fd9c-4b4f-8b36-778a7fd47b12",
            "productName": "S2A_MSIL2A_20200320T075621_N0500_R035_T44XNM_20230419T042038.zip",
            "date": "2024-09-12T11:40:44.020983087Z"
          },
          {
            "status": "QUEUED",
            "productId": "2208b2fa-6529-4e19-9b48-97339a808d3b",
            "productName": "S2B_MSIL2A_20200316T054639_N0500_R048_T46WFA_20230502T051038.zip",
            "date": "2024-09-12T11:40:44.030151297Z"
          },
          {
            "status": "QUEUED",
            "productId": "d4926f21-9d19-41ba-b904-4d630d9fa157",
            "productName": "S2B_MSIL2A_20200324T082609_N0500_R021_T37SBB_20230602T182526.zip",
            "date": "2024-09-12T11:40:44.036748583Z"
          },
          {
            "status": "QUEUED",
            "productId": "37fddc34-17bd-4eb7-b479-0cbe371966f1",
            "productName": "S2B_MSIL2A_20200322T130029_N0500_R138_T25SFC_20230419T051258.zip",
            "date": "2024-09-12T11:40:44.047121769Z"
          }
        ],
        "counters": {
          "QUEUED": 6
        },
        "state": "SHUTDOWN",
        "type": "PRODUCER",
        "updatedDate": "2024-09-12T11:40:47.833525157Z",
        "lastMessage": "Ingester cdse-p is shutdown"
      },
      {
        "name": "cdse-c",
        "events": [
          {
            "status": "INGESTED",
            "productId": "f03176fd-dfb6-4e0a-b5b4-30c6c39ea688",
            "productName": "S2A_MSIL2A_20200319T032531_N0500_R018_T47NQF_20230628T105144.zip",
            "date": "2024-09-12T12:05:18.998506776Z"
          },
          {
            "status": "INGESTED",
            "productId": "81dc8d8e-c3a9-4e48-9395-04bf1690837a",
            "productName": "S2A_MSIL2A_20231012T090911_N0509_R050_T34QBL_20231012T133754.zip",
            "date": "2024-09-12T12:06:09.960401581Z"
```

```
        },
        {
          "status": "INGESTED",
          "productId": "a418e776-b62e-4909-8f83-0a74fabaf3d6",
          "productName": "S2B_MSIL2A_20231012T081809_N0509_R121_T35PPL_20231012T113156.zip",
          "date": "2024-09-12T12:07:55.787635871Z"
        },
        {
          "status": "INGESTED",
          "productId": "3a85b188-b88d-4ae8-bf5a-909e92fb5a64",
          "productName": "S2B_MSIL2A_20200309T073729_N0500_R092_T37MBN_20230628T180704.zip",
          "date": "2024-09-12T12:08:32.946614824Z"
        },
        {
          "status": "INGESTED",
          "productId": "7abae324-230e-4f54-a4f5-418090d79d50",
          "productName": "S2A_MSIL2A_20200322T205031_N0500_R071_T04KFB_20230419T052751.zip",
          "date": "2024-09-12T12:08:55.423526989Z"
        }
      ],
      "counters": {
        "INGESTED": 5
      },
      "state": "STOPPING",
      "type": "CONSUMER",
      "updatedDate": "2024-09-12T12:08:55.423533455Z",
      "lastMessage": "cdse-c"
    }
  ]
}
```


12.12 Number of events

The events returned by a consumer and producer instance are not persisted. They will not be saved between restarts.

You can control the maximum number of events returned by changing the property **ingester.events.queue.size** in the application.properties file. The default value is **1000**.

13. STAC API

STAC API provides a set of endpoints and operations for querying STAC catalog, collections, and items, making it easier to find and access relevant data.

13.1 Pre-requisites

13.1.1 Infrastructure Requirements

We need to install:

- Docker: 20.10.12 (or later)
- docker compose: 1.29.0 (or later)

13.1.2 Network Requirements

Application	Default port
PostgreSQL	5432
SolR	8983
Keycloak version 18 or later	8443

13.1.3 Software Requirements

Some tools must be installed in order to use the stac-api catalog:

- A running SolrCloud instance, (9.0.0 and after: https://lucene.apache.org/solr/downloads.html) with a collection (use the **toolbox**)
- A running PostgreSQL instance, (10.12 and after: https://www.postgresql.org/download/) and a database initialize with **toolbox**.
- A running Keycloak instance if you want to use authentication (18.0.0 or later: https://www.keycloak.org/downloads)
- A storage depending on your scenario: folder, openstack swift

The Solr schema and the database schema update/creation scripts are inside the **CDH-toolbox** module. The database and the solr should be updated before launching a new version of the stac-api.

13.2 Distribution

This software is available in two formats, a zip archive and a Docker image.

13.2.1 Zip Archive

Available at the following address, the archive must be downloaded and unzipped in order to launch the

STAC-api catalog:

https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-stac-api/3.4.0/cdh-stac-api-3.4.0.zip

The zip archive is presented like:

```
stac-api/
├── etc/
│   ├── application.properties.sample
│   ├── docker-compose.yml
│   ├── gss_sample.xml
│   └── log4j2.xml
│
├── lib/
├── logs/
├── start.sh
└── stop.sh
```

13.2.2 Docker Image

Available on Docker Hub -> `docker pull registry.hub.docker.com/gaeldockerhub/cdh-stac-api:3.4.0`

You can check that the Docker image has been downloaded with the command `docker image list | grep cdh-stac-api.`

The Docker image is based on an **openjdk:17-jdk-slim** image. The STAC-api application will be launched in a created /stac-api directory. Don't forget to expose the port of the STAC catalog when launching the Docker container.

To launch the Docker container, you must provide the **gss.xml** and the **application.properties** files as attached volumes. Examples for those files are available in the Zip archive.

Example:

```
docker run -d --name cdh-stac-api -v /path/gss.xml:/stac/etc/gss.xml -v
/path/application.properties:/stac/etc/application.properties -it
registry.hub.docker.com/gaeldockerhub/cdh-stac-api:3.4.0
```

The -d option is used to run the Docker container in detached mode. If you want your Docker container to point to the localhost of the hosting machine, add the launch option --network="host" (https://docs.docker.com/network/host/).

It means that the container shares the host machine network, the container will have the same IP and ports as the host. For example, if you launch the container and you want to access the STAC-api catalog on localhost:8080, you must use this option. You can map the log files of the application with a local folder by adding an attached volume:

```
-v /your/path/to/logs:/stac/logs.
```

To gracefully stop the container, add a timeout (in seconds) to the stop command:

```
docker stop -t 60 cdh-stac-api
```

13.3 STAC API Configuration

Go to the **stac** directory.

Configure **application.properties** to enable the STAC catalog navigation and Keycloak configuration. This file is self-described and is in *cdh-stac-api/etc/application.properties.sample*. Rename this file to application.properties to use it.

13.3.1 STAC API Keycloak Configuration

CDH-STAC-API requires Keycloak version 18 or later. To ensure the users access to CDH-STAC catalog a Keycloak setting must be enabled through the Keycloak console page.

- Authentication via a web API.
- Authentication via a web browser.

13.3.1.1 Keycloak Web-API Authentication Configuration

The Web-API authentication uses the OAuth 2.0 protocol to protected resource requests by verifying authorized access tokens. To enable this authentication, it is essential to add user/users to both the Keycloak client-access role and realm-access role.

To do so, the following steps must be taken:

- Access the Keycloak admin console.
- Select the relevant Keycloak realm.

1. **Keycloak Realm-access:**
   - Within your desired realm, navigate to the "Realm" section in the left sidebar.
   - To ensure that the user is listed in roles for realm-access, please follow the appropriate steps:
     - `Admin-console → Realm roles → <your-role> → Users in role → <desired-user>`

2. **Keycloak Client-access:**
   - Within your desired realm, navigate to the "Clients" section in the left sidebar.
   - Here, you'll see a list of clients associated with the current realm. Initially, you might see some default clients like "account" and "realm-management."
   - To ensure that the user is listed in roles for client-access, please follow the appropriate steps:
     - `Admin-console → Clients → <your-client> → roles → <your-role> → Users in role → <desired-user>`

3. **Keycloak Users:**
   - To begin, go to the "Users" section in the left sidebar of your desired realm.
   - You will find a list of users here.
   - It's important to note that there are two types of roles
     - The `realm-access role`.
     - The `client-access role`.

   - Role mapping for the realm-access role:
     - `Admin-console → Users → <desired-user> → Role Mapping → Assign role → Filter by realm roles → Filter by clients → <your-role> → Assign`

   - Role mapping for the client-access role:
     - `Admin-console → Users → <desired-user> → Role Mapping → Assign role → <your-role> → Assign`

Note:

- Please ensure to see the user/users in (Users in role) of realm-access and client-access.
- The settings in the Keycloak console may differ depending on the version utilized.
- The following website allows to see the users and roles:
  - Generate a token.
  - Decode the token.
  - Verification of the token containing the roles at the https://jwt.io
- Please refer to RD-4 for the complete Keycloak configuration.

User role mapping example in realm and client access below:

```
{
  "realm_access": {
    "roles": [
      "user-role",              // realm-access role
      "default-roles-admin-realm",
      "offline_access",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "user-client": {
      "roles": [
        "user-role"             // client-access role
      ]
    },
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "preferred_username": "cdh-user"
}
```

13.3.1.2 Keycloak Web-browser Authentication Configuration

CDH-STAC-API uses Oauth2 client with Single Sign-On (SSO) as an authentication mechanism to allows user access to the STAC catalog via a web browser. SSO on CDH-STAC-api works as follows:

- **Authentication:** When a user attempts to access the STAC catalog, the user is redirected to Keycloak's authentication server.
- **Login:** The user enters their credentials (username and password) on Keycloak's authentication login page.
- **Token Generation:** Upon successful authentication, Keycloak authentication server generates a security token that represents the user's identity and contains some relevant information about the user.
- **Token Validation:** The Catalog then receives the security token for further verification.

Enabling Single Sign-On and accessing the catalog via a web browser requires that the userInfo checkbox setting be selected in the following way:

- Log in to the Keycloak admin console.
- Choose the Keycloak realm.
- `Navigate to Client Scopes → roles → Mappers → client roles → Add to userinfo (checkbox)`
- `Proceed to Client Scopes → roles → Mappers → realm roles → Add to userinfo (checkbox)`

In Keycloak, userInfo is a pre-defined scope that allows clients to request access to fundamental user information from Keycloak's userinfo during the OAuth 2.0 token exchange process.

13.3.2 Keycloak Authentication

The table below describes the Keycloak roles needed by users to perform operations on the system.

User Role	Description	DHS Actor (admin / user)
end-user	End user with right to list products, list details of a product, download products	
admin	A user with all rights on Ingesters, datastores and metadatastores	

The **application.properties** can be used to enable authentication through a Keycloak server. To use OAuth2 authentication through command line:

- Request a token:

```
curl -d 'client_id=<keycloak_client>' -d 'username=***' -d 'password=***' -d
'grant_type=password' "https://<keycloak_url>/auth/realms/<keycloak_realm>/protocol/openid-connect/token"
```

You will obtain a JSON response containing the access_token

- Use the token in the request:

```
curl -X GET -H 'authorization: Bearer <access_token>' -H 'content-type: application/json'
"http://<server.url>:<server.port>/<context-path>/stac/search"
```

It is also important to configure which information will be used to identify users. So, you need to provide the correct value to parameter "**user-name-attribute** " in **application.properties**.

There are 2 possibilities:

- prefered_username: using the prefered user's name configured in Keycloak to identify the user in logs and for quotas management.
- user_id: using the user id defined in Keycloack to identify the user in logs and for quotas management.

IMPORTANT

```
# Which attribute to use to identify the user (displayed in logs and used for quotas)

# Possible values are: user_id, preferred_username. Default value is user_id.

user-name-attribute=preferred_username
```

13.3.3 Configuration via XML file

STAC collections are defined in this file. Each of them has mandatory attributes detailed in the following examples. A collection corresponds to a Solr Filter.

For example, if you want to have a collection dedicated to Sentinel-2 products, this filter would be "(platformShortName:SENTINEL-2)".

If you want a collection dedicated to a product type, like Sentinel-3 OLCI products, this filter would be "(instrumentShortName:OLCI)".

Note: even if you are using stores configured in database, the gss.xml is used to define STAC collections. If application.properties file is used with the stores created via Admin-API component, set also parameter useDbConfiguration = true and configure only collections part in the gss-stac-api.xml file.

13.3.3.1 HFSDataStore XML Configuration

Stac-gss-api.xml example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- This file is used to configure DataStores, MetadataStores, STAC Collections and
system parameters of the API -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->
<conf:configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:conf="fr:gael:gss:stac:configuration" xmlns:ds="fr:gael:gss:core:datastore"
xmlns:ms="fr:gael:gss:core:metadatastore" xmlns:dl="fr:gael:gss:odata:download"
xmlns:metrics="fr:gael:gss:core:metrics" xmlns:col="fr:gael:gss:stac:collection">

   <!-- [M] Access to the PostgreSQL database -->
   <!-- The attribute poolSize is optional (default 10) and is the max opened connection
to the database -->
   <conf:database jdbcUrl="jdbc:postgresql://localhost:5432/postgres" login="postgres"
password="password" poolSize="5" />

   <!-- [0] Configuration tweaks -->
   <!-- directDownloadLink: enable the direct download from cloud stores (swift) when using
$value. Default is true. -->
   <!-- quotaDisabled: disable quotas. Default is false. -->
   <conf:download directDownloadLink="false" quotaDisabled="true" />

   <!-- Stores are configured below. In this scenario, Swift containers are used to store
products and quicklooks. A Solr
      index is used to store metadata -->
   <conf:dataStores>
         <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfs">
         <!-- [0] Access permissions. Default is read-only -->
         <ds:permission>WRITE</ds:permission>
         <ds:permission>READ</ds:permission>
         <ds:permission>DELETE</ds:permission>
         <ds:properties>
            <!-- [0] if true, this store will be able to store attached files. Default is
false -->
            <ds:property>
```

```
               <ds:name>STORE_ATTACHED_FILES</ds:name>
               <ds:value>true</ds:value>
            </ds:property>
         </ds:properties>

         <ds:path>/catalogue/hfs</ds:path>
         <!-- [0] How many levels of directories there will be. Default is 2 -->
         <ds:depth>2</ds:depth>
         <!-- [0] How many characters of an UUID will be taken for a directory's name.
Default is 2 -->
         <ds:granularity>2</ds:granularity>
      </ds:dataStore>
   </conf:dataStores>

   <!-- [0] Metadata stores are defined below. SolrCloud instance in this scenario -->
   <conf:metadataStores>
      <!-- [0] Solr Metastore -->
      <ms:metadataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ms:SolrMetadataStoreConf" name="solr">
         <!-- [0] Access permissions. Default is READ only -->
         <ms:permission>READ</ms:permission>
         <ms:permission>WRITE</ms:permission>
         <ms:permission>DELETE</ms:permission>

         <!-- [M] comma separated solr nodes (1 to n nodes) -->
<ms:hosts>http://solr:8981/solr,http://solr:8982/solr,http://solr:8983/solr</ms:hosts>

         <!-- [0] Client type to connect solr -->
         <!-- LBHttp: load balanced http access (round-robin) to all specified nodes -->
         <!-- SolrCloud: default, discover solr nodes with zookeeper -->
         <ms:clientType>LBHttp</ms:clientType>

         <!-- [0] Solr username -->
         <!-- <ms:user>***</ms:user>  -->
         <!-- [0] Solr password -->
         <!-- <ms:password>***</ms:password>  -->

         <!-- [M] name of the solr collection -->
         <ms:collection>cdh</ms:collection>
      </ms:metadataStore>
   </conf:metadataStores>
```

```
   <!-- [0] List of exposed STAC Collections. In this example there is one SENTINEL-3
Collection. -->
   <!-- More info on collection specification here: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md -->
   <conf:stacCollections>
      <!-- [M] Collection with its id -->
      <col:stacCollection id="SENTINEL-3">
         <!-- [0] The STAC version the Collection implements. Default is 1.1.0 -->
         <col:stacVersion>1.1.0</col:stacVersion>
         <!-- [0] A short descriptive one-line title for the Collection -->
         <col:title>SENTINEL-3</col:title>
         <!-- [M] Detailed multi-line description to fully explain the Collection -->
         <col:description><![CDATA[The Copernicus Sentinel-3 mission consists of two
polar-orbiting satellites that are positioned in the same sun-synchronous orbit, with a
phase difference of 180°. It aims to monitor changes in land surface conditions. The
satellites have a wide swath width (290 km) and a high revisit time. Sentinel-2 is
equipped with an optical instrument payload that samples 13 spectral bands: four bands at
10 m, six bands at 20 m and three bands at 60 m spatial resolution.]]>
         </col:description>
         <!-- [0] List of comma separated keywords describing the Collection -->
         <col:keywords>copernicus, esa, sentinel, optical</col:keywords>
         <!-- [M] Collection's license(s), either a SPDX License identifier, various if
multiple licenses apply or proprietary
            for all other cases. -->
         <col:license>various</col:license>
         <!-- [0] A list of providers, which may include all organizations capturing or
processing the data or the hosting
            provider. Providers should be listed in chronological order with the most
recent provider being the last element of the list. -->
         <col:providers>
            <col:provider>
               <!-- [M] The name of the organization or the individual -->
               <col:name>Gael Systems</col:name>
               <!-- [0] Multi-line description to add further provider information such as
processing details for processors
                  and producers, hosting details for hosts or basic contact information. --
>
               <col:description></col:description>
               <!-- [0] Comma separated roles of the provider. Any of licensor, producer,
processor or host -->
```

```
               <col:roles>host</col:roles>
               <!-- [0] Homepage on which the provider describes the dataset and publishes
contact information -->
               <col:url>https://www.gael-systems.com/</col:url>
            </col:provider>
         </col:providers>
         <!-- [M] Spatial and temporal extents -->
         <col:extents>
            <!-- [0] Spatial extent. Default is an empty bbox -->
            <col:extent xsi:type="col:spatialExtentConf">
               <!-- [0] Bbox, definition: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md#spatial-extent-object -->
               <col:bbox>[[-180, -90, 180, 90]]</col:bbox>
            </col:extent>
            <!-- [0] Temporal extent. Default is open date range -->
            <col:extent xsi:type="col:temporalExtentConf">
               <!-- [0] Interval, do not use quotes for dates. Definition:
https://github.com/radiantearth/stac-spec/blob/master/collection-spec/collection-
spec.md#temporal-extent-object -->
               <col:interval>[[2015-07-01T00:00:00Z, null]]</col:interval>
            </col:extent>
         </col:extents>
         <!-- [M] Solr filter. Products matching it will be in this collection.-->
         <col:solrFilter>(platformShortName:SENTINEL-3)</col:solrFilter>
      </col:stacCollection>
   </conf:stacCollections>
</conf:configuration>
```

13.3.3.2 SwiftDataStore XML Configuration

Stac-gss-api.xml example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure DataStores, MetadataStores, STAC Collections and system
parameters of the API -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
   xmlns:conf="fr:gael:gss:stac:configuration" xmlns:ds="fr:gael:gss:core:datastore"
   xmlns:ms="fr:gael:gss:core:metadatastore" xmlns:dl="fr:gael:gss:odata:download"
   xmlns:metrics="fr:gael:gss:core:metrics" xmlns:col="fr:gael:gss:stac:collection">

   <!-- [M] Access to the PostgreSQL database -->
   <!-- The attribute poolSize is optional (default 10) and is the max opened connection to
the database -->
   <conf:database    jdbcUrl="jdbc:postgresql://postgres:5432/postgres"    login="postgres"
password="password" poolSize="5" />

   <!-- [0] Configuration tweaks -->
   <!-- directDownloadLink: enable the direct download from cloud stores (swift) when using
$value. Default is true. -->
   <!-- quotaDisabled: disable quotas. Default is false. -->
   <conf:download directDownloadLink="false" quotaDisabled="true" />

   <!-- Stores are configured below. In this scenario, Swift containers are used to store
products and quicklooks. A Solr index is used to store metadata -->
   <conf:dataStores>
      <!-- [0] Configuration tweaks for Swift behavior -->
      <!-- [0] segmentSizeMB: size in MB of segments. Default is 500MB -->
      <!-- [0] retries: number of retries done for every swift command. Default is 5 -->
      <!-- [0] retryDelayMs: delay before a command is retried. Default is 50 ms -->
      <!-- [0] maxConnections: maximum simultaneous connections opened to the swift storage.
Default is 20 -->
      <ds:swiftConfiguration     segmentSizeMB="100"     retries="5"     retryDelayMs="50"
maxConnections="20" />

      <!-- Swift credentials -->
      <ds:swiftCredentials name="credentials">
         <ds:tenant>***</ds:tenant>
         <ds:password>***</ds:password>
         <ds:user>***</ds:user>
         <ds:url>https://auth.cloud.ovh.net/v3</ds:url>
         <ds:region>***</ds:region>
      </ds:swiftCredentials>

      <!-- [0] First swift group that store products as packages -->
      <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ds:swiftDataStoreGroupConf" name="swiftPackage">
         <!-- [0] Access permissions. Default is read-only -->
```

```
         <ds:permission>WRITE</ds:permission>
         <ds:permission>READ</ds:permission>
         <ds:permission>DELETE</ds:permission>

         <!-- [0] Different properties managing the behavior of the store -->
         <ds:properties>
            <!-- [0] if true, products will be stored by their name. Default is false (stored
by uuid) -->
            <ds:property>
               <ds:name>STORE_BY_NAME</ds:name>
               <ds:value>true</ds:value>
            </ds:property>
            <!-- [0] if true, products are stored as multiparts. Default is false (stored
as package) -->
            <ds:property>
               <ds:name>SAVE_AS_MULTIPART</ds:name>
               <ds:value>false</ds:value>
            </ds:property>
         </ds:properties>
         <!-- [M] credentials to access the containers (defined before datastores) -->
         <ds:credentials>credentials</ds:credentials>
         <!-- [M] filter which products based on their name are accepted. ".*" means all
products -->
         <ds:filter>.*</ds:filter>
         <!-- [M] the pattern used to identify/create containers in this group -->
         <!-- patternMapper is a comma separated list of key:value pairs -->
         <!-- the keys are pattern on product names and the values are the name of the
containers -->
         <!-- pattern are analyzed in declared order, first one matching wins -->
         <ds:containerPattern xsi:type="ds:mapperContainerPatternConf">
            <ds:patternMapper>patternMapper="S1.*:CDH-Sentinel-1,S2.*:CDH-Sentinel-
2,S3.*:CDH-Sentinel-3,S5.*:CDH-Sentinel-5P,.*:CDH-Other
            </ds:patternMapper>
         </ds:containerPattern>
         <!-- [0] If defined, products will be stored with a prefix in containers -->
         <!-- the prefix is extracted from the product name, and can be combination of -->
         <!-- instrument, productType, sensing date (year, month, day) -->
         <ds:prefixLocation>instrument/productType/year/month/day</ds:prefixLocation>
      </ds:dataStore>

      <!-- [0] This store is a swift container that is used to store quicklooks -->
```

```
      <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:swiftDataStoreConf"
         name="swiftQL">
         <!-- [0] Access permissions. Default is read-only -->
         <ds:permission>WRITE</ds:permission>
         <ds:permission>READ</ds:permission>
         <ds:permission>DELETE</ds:permission>
         <ds:properties>
            <!-- [0] if true, this store will be able to store attached files. Default is
false -->
            <ds:property>
               <ds:name>STORE_ATTACHED_FILES</ds:name>
               <ds:value>true</ds:value>
            </ds:property>
            <!-- [0] if true, products will be stored by their name. Default is false
(stored by uuid) -->
            <ds:property>
               <ds:name>STORE_BY_NAME</ds:name>
               <ds:value>true</ds:value>
            </ds:property>
         </ds:properties>
         <!-- [M] credentials to access the container (defined before datastores) -->
         <ds:credentials>credentials</ds:credentials>
         <!-- [M] name of the container -->
         <ds:container>CDH-Quicklooks</ds:container>
         <!-- [0] If defined, products will be stored with a prefix in buckets -->
         <!-- the prefix is extracted from the product name, and can be combination of -->
         <!-- instrument, productType, sensing date (year, month, day) -->
         <ds:prefixLocation>instrument/productType/year/month/day</ds:prefixLocation>
      </ds:dataStore>
   </conf:dataStores>

   <!-- [0] Metadata stores are defined below. SolrCloud instance in this scenario -->
   <conf:metadataStores>
      <!-- [0] Solr Metastore -->
      <ms:metadataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ms:SolrMetadataStoreConf" name="solr">
         <!-- [0] Access permissions. Default is READ only -->
         <ms:permission>READ</ms:permission>
         <ms:permission>WRITE</ms:permission>
         <ms:permission>DELETE</ms:permission>
```

```
        <!-- [M] comma separated solr nodes (1 to n nodes) -->
        <ms:hosts>http://solr:8983/solr, http://solr:7574/solr</ms:hosts>

        <!-- [0] Client type to connect solr -->
        <!-- LBHttp: load balanced http access (round-robin) to all specified nodes -->
        <!-- SolrCloud: default, discover solr nodes with zookeeper -->
        <ms:clientType>LBHttp</ms:clientType>

        <!-- [0] Solr username -->
        <ms:user>***</ms:user>
        <!-- [0] Solr password -->
        <ms:password>***</ms:password>

        <!-- [M] name of the solr collection -->
        <ms:collection>cdh</ms:collection>
      </ms:metadataStore>
   </conf:metadataStores>

   <!-- [0] List of exposed STAC Collections. In this example there is one SENTINEL-2
Collection. -->
   <!-- More info on collection specification here: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md -->
   <conf:stacCollections>
      <!-- [M] Collection with its id -->
      <col:stacCollection id="SENTINEL-2">
         <!-- [0] The STAC version the Collection implements. Default is 1.1.0 -->
         <col:stacVersion>1.1.0</col:stacVersion>
         <!-- [0] A short descriptive one-line title for the Collection -->
         <col:title>SENTINEL-2</col:title>
         <!-- [M] Detailed multi-line description to fully explain the Collection -->
         <col:description><![CDATA[The Copernicus Sentinel-2 mission consists of two
polar-orbiting satellites that are positioned in the same sun-synchronous orbit, with a
phase difference of 180°. It aims to monitor changes in land surface conditions. The
satellites have a wide swath width (290 km) and a high revisit time. Sentinel-2 is
equipped with an optical instrument payload that samples 13 spectral bands: four bands at
10 m, six bands at 20 m and three bands at 60 m spatial resolution.]]>
         </col:description>
         <!-- [0] List of comma separated keywords describing the Collection -->
         <col:keywords>copernicus, esa, sentinel, optical</col:keywords>
```

```
         <!-- [M] Collection's license(s), either a SPDX License identifier, various if
multiple licenses apply or proprietary
            for all other cases. -->
         <col:license>various</col:license>
         <!-- [0] A list of providers, which may include all organizations capturing or
processing the data or the hosting
            provider. Providers should be listed in chronological order with the most
recent provider being the last element of the list. -->
         <col:providers>
            <col:provider>
               <!-- [M] The name of the organization or the individual -->
               <col:name>Gael Systems</col:name>
               <!-- [0] Multi-line description to add further provider information such as
processing details for processors
                  and producers, hosting details for hosts or basic contact information. -
->
               <col:description></col:description>
               <!-- [0] Comma separated roles of the provider. Any of licensor, producer,
processor or host -->
               <col:roles>host</col:roles>
               <!-- [0] Homepage on which the provider describes the dataset and publishes
contact information -->
               <col:url>https://www.gael-systems.com/</col:url>
            </col:provider>
         </col:providers>
         <!-- [M] Spatial and temporal extents -->
         <col:extents>
            <!-- [0] Spatial extent. Default is an empty bbox -->
            <col:extent xsi:type="col:spatialExtentConf">
               <!-- [0] Bbox, definition: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md#spatial-extent-object -->
               <col:bbox>[[-180, -90, 180, 90]]</col:bbox>
            </col:extent>
            <!-- [0] Temporal extent. Default is open date range -->
            <col:extent xsi:type="col:temporalExtentConf">
               <!-- [0] Interval, do not use quotes for dates. Definition:
https://github.com/radiantearth/stac-spec/blob/master/collection-spec/collection-
spec.md#temporal-extent-object -->
               <col:interval>[[2015-07-01T00:00:00Z, null]]</col:interval>
            </col:extent>
         </col:extents>
```

```
        <!-- [M] Solr filter. Products matching it will be in this collection. -->
        <col:solrFilter>(platformShortName:SENTINEL-2)</col:solrFilter>
      </col:stacCollection>
   </conf:stacCollections>
</conf:configuration>
```

13.3.3.3 SwiftDataStoreGroup XML Configuration

Stac-gss-api.xml example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure DataStores, MetadataStores, STAC Collections and system
parameters of the API -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:conf="fr:gael:gss:stac:configuration" xmlns:ds="fr:gael:gss:core:datastore"
   xmlns:ms="fr:gael:gss:core:metadatastore" xmlns:dl="fr:gael:gss:odata:download"
   xmlns:metrics="fr:gael:gss:core:metrics" xmlns:col="fr:gael:gss:stac:collection">

   <!-- [M] Access to the PostgreSQL database -->
   <!-- The attribute poolSize is optional (default 10) and is the max opened connection to
the database -->
   <conf:database    jdbcUrl="jdbc:postgresql://postgres:5432/postgres"    login="postgres"
password="password" poolSize="5" />

   <!-- [0] Configuration tweaks -->
   <!-- directDownloadLink: enable the direct download from cloud stores (swift) when using
$value. Default is true. -->
   <!-- quotaDisabled: disable quotas. Default is false. -->
   <conf:download directDownloadLink="false" quotaDisabled="true" />

   <!-- Stores are configured below. In this scenario, Swift containers are used to store
products and quicklooks. A Solr index is used to store metadata -->
   <conf:dataStores>
      <!-- [0] Configuration tweaks for Swift behavior -->
      <!-- [0] segmentSizeMB: size in MB of segments. Default is 500MB -->
      <!-- [0] retries: number of retries done for every swift command. Default is 5 -->
      <!-- [0] retryDelayMs: delay before a command is retried. Default is 50 ms -->
      <!-- [0] maxConnections: maximum simultaneous connections opened to the swift storage.
Default is 20 -->
      <ds:swiftConfiguration segmentSizeMB="100" retries="5" retryDelayMs="50"
maxConnections="20" />

      <!-- Swift credentials -->
      <ds:swiftCredentials name="credentials">
         <ds:tenant>***</ds:tenant>
         <ds:password>***</ds:password>
```

```
         <ds:user>***</ds:user>
         <ds:url>https://auth.cloud.ovh.net/v3</ds:url>
         <ds:region>***</ds:region>
      </ds:swiftCredentials>

      <!-- [0] First swift group that store products as packages -->
  <ds:dataStore xsi:type="ds:swiftDataStoreGroupConf" name="swiftPackage">
         <!-- [0] Access permissions. Default is read-only -->
         <ds:permission>WRITE</ds:permission>
         <ds:permission>READ</ds:permission>
         <ds:permission>DELETE</ds:permission>
         <!-- [0] Different properties managing the behavior of the store -->


         <ds:properties>
            <!-- [0] if true, products will be stored by their name. Default is false (stored
by uuid) -->
            <ds:property>
               <ds:name>STORE_BY_NAME</ds:name>
               <ds:value>true</ds:value>
            </ds:property>
            <!-- [0] if true, products will be stored as multiparts. Default is false (stored
as package) -->
            <ds:property>
               <ds:name>SAVE_AS_MULTIPART</ds:name>
               <ds:value>false</ds:value>
            </ds:property>
            <!-- [M] How much time in seconds a product will be kept in this store -->
            <ds:property>
               <ds:name>KEEP_PERIOD_SECONDS</ds:name>
               <ds:value>3000</ds:value>
            </ds:property>
         </ds:properties>


         <!-- [M] credentials to access the containers (defined before datastores) -->
         <ds:credentials>SwiftCredentials</ds:credentials>
        <!-- [M] This is a regex product filename filtering on kafka messages received. -->
```

```
         <!-- .* means that all messages will be processed by consumer without any filter.
-->
         <ds:filter>.*</ds:filter>
         <!-- [M] the pattern used to identify/create containers in this group -->
         <!-- patternMapper is a comma separated list of key:value pairs -->
         <!-- the keys are pattern on product names and the values are the name of the
containers -->
         <!-- pattern are analyzed in declared order, first one matching wins -->
         <ds:containerPattern xsi:type="ds:mapperContainerPatternConf">
            <ds:patternMapper>S1.*:S1-container,S2.*:S2-container,S3.*:S3-
container,S5.*:S5P-container</ds:patternMapper>
         </ds:containerPattern>


         <!-- [0] If defined, products will be stored with a prefix in containers -->
         <!-- the prefix is extracted from the product name, and can be combination of -->
         <!-- instrument, productType, sensing date (year, month, day) -->
         <ds:prefixLocation>instrument/productType/year/month/day</ds:prefixLocation>
      </ds:dataStore>
   </conf:dataStores>

   <!-- [0] Metadata stores are defined below. SolrCloud instance in this scenario -->
   <conf:metadataStores>
      <!-- [0] Solr Metastore -->
      <ms:metadataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ms:SolrMetadataStoreConf" name="solr">
         <!-- [0] Access permissions. Default is READ only -->
         <ms:permission>READ</ms:permission>
         <ms:permission>WRITE</ms:permission>
         <ms:permission>DELETE</ms:permission>

         <!-- [M] comma separated solr nodes (1 to n nodes) -->
         <ms:hosts>http://solr:8983/solr, http://solr:7574/solr</ms:hosts>

         <!-- [0] Client type to connect solr -->
         <!-- LBHttp: load balanced http access (round-robin) to all specified nodes -->
         <!-- SolrCloud: default, discover solr nodes with zookeeper -->
         <ms:clientType>LBHttp</ms:clientType>

         <!-- [0] Solr username -->
         <ms:user>***</ms:user>
         <!-- [0] Solr password -->
         <ms:password>***</ms:password>
```

```
         <!-- [M] name of the solr collection -->
         <ms:collection>cdh</ms:collection>
      </ms:metadataStore>
   </conf:metadataStores>

   <!-- [0] List of exposed STAC Collections. In this example there is one SENTINEL-2
Collection. -->
   <!-- More info on collection specification here: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md -->
   <conf:stacCollections>
      <!-- [M] Collection with its id -->
      <col:stacCollection id="SENTINEL-2">
         <!-- [0] The STAC version the Collection implements. Default is 1.1.0 -->
         <col:stacVersion>1.1.0</col:stacVersion>
         <!-- [0] A short descriptive one-line title for the Collection -->
         <col:title>SENTINEL-2</col:title>
         <!-- [M] Detailed multi-line description to fully explain the Collection -->
         <col:description><![CDATA[The Copernicus Sentinel-2 mission consists of two polar-
orbiting satellites that are positioned in the same sun-synchronous orbit, with a phase
difference of 180°. It aims to monitor changes in land surface conditions. The satellites
have a wide swath width (290 km) and a high revisit time. Sentinel-2 is equipped with an
optical instrument payload that samples 13 spectral bands: four bands at 10 m, six bands at
20 m and three bands at 60 m spatial resolution.]]>
         </col:description>
         <!-- [0] List of comma separated keywords describing the Collection -->
         <col:keywords>copernicus, esa, sentinel, optical</col:keywords>
         <!-- [M] Collection's license(s), either a SPDX License identifier, various if
multiple licenses apply or proprietary
            for all other cases. -->
         <col:license>various</col:license>
         <!-- [0] A list of providers, which may include all organizations capturing or
processing the data or the hosting
            provider. Providers should be listed in chronological order with the most recent
provider being the last element of the list. -->
         <col:providers>
            <col:provider>
               <!-- [M] The name of the organization or the individual -->
               <col:name>Gael Systems</col:name>
               <!-- [0] Multi-line description to add further provider information such as
processing details for processors
                  and producers, hosting details for hosts or basic contact information. --
>
               <col:description></col:description>
               <!-- [0] Comma separated roles of the provider. Any of licensor, producer,
processor or host -->
               <col:roles>host</col:roles>
               <!-- [0] Homepage on which the provider describes the dataset and publishes
contact information -->
               <col:url>https://www.gael-systems.com/</col:url>
            </col:provider>
         </col:providers>
         <!-- [M] Spatial and temporal extents -->
         <col:extents>
```

```
            <!-- [0] Spatial extent. Default is an empty bbox -->
            <col:extent xsi:type="col:spatialExtentConf">
               <!-- [0] Bbox, definition: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md#spatial-extent-object -->
               <col:bbox>[[-180, -90, 180, 90]]</col:bbox>
            </col:extent>
            <!-- [0] Temporal extent. Default is open date range -->
            <col:extent xsi:type="col:temporalExtentConf">
               <!-- [0] Interval, do not use quotes for dates. Definition:
https://github.com/radiantearth/stac-spec/blob/master/collection-spec/collection-
spec.md#temporal-extent-object -->
               <col:interval>[[2015-07-01T00:00:00Z, null]]</col:interval>
            </col:extent>
         </col:extents>
         <!-- [M] Solr filter. Products matching it will be in this collection. -->
         <col:solrFilter>(platformShortName:SENTINEL-2)</col:solrFilter>
      </col:stacCollection>
   </conf:stacCollections>

</conf:configuration>
```

13.3.3.4 TimeBasedDataStoreGroup XML Configuration

Stac-gss-api.xml example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure DataStores, MetadataStores, STAC Collections and
system parameters of the API -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:conf="fr:gael:gss:stac:configuration" xmlns:ds="fr:gael:gss:core:datastore"
   xmlns:ms="fr:gael:gss:core:metadatastore" xmlns:dl="fr:gael:gss:odata:download"
   xmlns:metrics="fr:gael:gss:core:metrics" xmlns:col="fr:gael:gss:stac:collection">

   <!-- [M] Access to the PostgreSQL database -->
   <!-- The attribute poolSize is optional (default 10) and is the max opened connection
to the database -->
   <conf:database jdbcUrl="jdbc:postgresql://postgres:5432/postgres" login="postgres"
password="password" poolSize="5" />

   <!-- [0] Configuration tweaks -->
   <!-- directDownloadLink: enable the direct download from cloud stores (swift) when
using $value. Default is true. -->
   <!-- quotaDisabled: disable quotas. Default is false. -->
   <conf:download directDownloadLink="false" quotaDisabled="true" />

   <!-- Stores are configured below. In this scenario, Time Based stores are used to store
products and quicklooks. A Solr index is used to store metadata -->
```

```
   <conf:dataStores>
      <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:timeBasedDataStoreGroupConf" name="TimeGroup">
         <!-- [0] Access permissions. Default is read-only -->
         <ds:permission>READ</ds:permission>
         <ds:permission>WRITE</ds:permission>
         <ds:permission>DELETE</ds:permission>

         <!-- Define the properties -->
         <ds:properties>
            <!-- EVICT_REFERENCE: if true, when an eviction occurs, the product will also
be evicted from MetaStores -->
            <ds:property>
               <ds:name>STORE_BY_NAME</ds:name>
               <ds:value>true</ds:value>
            </ds:property>

     <ds:property>
               <ds:name>SAVE_AS_MULTIPART</ds:name>
               <ds:value>false</ds:value>
            </ds:property>
     <ds:property>
               <ds:name>KEEP_PERIOD_SECONDS</ds:name>
               <ds:value>432000</ds:value>
            </ds:property>
         </ds:properties>

         <!-- [M] Regex used to filter products that can be added based on their name. To
accept all products: .* -->
         <ds:filter>.*</ds:filter>
         <!-- [M] Control the orders of stores in this group. Here each store has a
manually assigned priority -->
         <ds:policy>UserDefinedPriorityPolicy</ds:policy>

         <!-- [M] DataStores that make up this group. -->
         <ds:dataStores>
            <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfsQL_DAS_S1">
               <ds:permission>READ</ds:permission>
               <ds:permission>WRITE</ds:permission>
               <ds:permission>DELETE</ds:permission>

               <ds:properties>
                  <!-- EVICT_ATTACHED_FILES: If an eviction occurs, evict all attached
files (quicklooks) if set to true. Default is false -->
                  <ds:property>
                     <ds:name>STORE_ATTACHED_FILES</ds:name>
                     <ds:value>true</ds:value>
                  </ds:property>
                  <!-- [M] Assigned priority. The lower the value is, the highest is the
priority -->
                  <ds:property>
```

```
                     <ds:name>PRIORITY</ds:name>
                     <ds:value>0</ds:value>
                  </ds:property>
               </ds:properties>
               <ds:path>/ingest/folder1</ds:path>

               <!-- [0] How many levels of directories there will be. Default is 2 -->
               <ds:depth>0</ds:depth>
               <!-- [0] How many characters of an UUID will be taken for a directory's
name. Default is 2 -->
               <ds:granularity>2</ds:granularity>
            </ds:dataStore>

            <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfsQL_DAS_S2">
               <ds:permission>READ</ds:permission>
               <ds:permission>WRITE</ds:permission>
               <ds:permission>DELETE</ds:permission>
               <ds:properties>
                  <!-- EVICT_REFERENCE: if true, when an eviction occurs, the product will
also be evicted from MetaStores -->
                  <ds:property>
                     <ds:name>STORE_ATTACHED_FILES</ds:name>
                     <ds:value>true</ds:value>
                  </ds:property>

                  <!-- [M] Assigned priority. The lower the value is, the highest is the
priority -->
                  <ds:property>
                     <ds:name>PRIORITY</ds:name>
                     <ds:value>0</ds:value>
                  </ds:property>
               </ds:properties>
               <ds:path>/ingest/folder2</ds:path>
               <!-- [0] How many levels of directories there will be. Default is 2 -->
               <ds:depth>0</ds:depth>
               <!-- [0] How many characters of an UUID will be taken for a directory's
name. Default is 2 -->
               <ds:granularity>2</ds:granularity>
            </ds:dataStore>
            <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfsQL_DAS_S3">
               <ds:permission>READ</ds:permission>
               <ds:permission>WRITE</ds:permission>
               <ds:permission>DELETE</ds:permission>
               <ds:properties>
                  <!-- EVICT_REFERENCE: if true, when an eviction occurs, the product will
also be evicted from MetaStores -->
                  <ds:property>
                     <ds:name>STORE_ATTACHED_FILES</ds:name>
                     <ds:value>true</ds:value>
                  </ds:property>
```

```
                  <!-- [M] Assigned priority. The lower the value is, the highest is the
priority -->
                  <ds:property>
                     <ds:name>PRIORITY</ds:name>
                     <ds:value>0</ds:value>
                  </ds:property>
               </ds:properties>
               <ds:path>/ingest/folder3</ds:path>
               <!-- [0] How many levels of directories there will be. Default is 2 -->
               <ds:depth>0</ds:depth>
               <!-- [0] How many characters of an UUID will be taken for a directory's
name. Default is 2 -->
               <ds:granularity>2</ds:granularity>
            </ds:dataStore>
            <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf" name="hfsQL_DAS_S5P">
               <ds:permission>READ</ds:permission>
               <ds:permission>WRITE</ds:permission>
               <ds:permission>DELETE</ds:permission>
               <ds:properties>
                  <!-- EVICT_REFERENCE: if true, when an eviction occurs, the product will
also be evicted from MetaStores -->
                  <ds:property>
                     <ds:name>STORE_ATTACHED_FILES</ds:name>
                     <ds:value>true</ds:value>
                  </ds:property>

                  <!-- [M] Assigned priority. The lower the value is, the highest is the
priority -->
                  <ds:property>
                     <ds:name>PRIORITY</ds:name>
                     <ds:value>0</ds:value>
                  </ds:property>
               </ds:properties>
               <ds:path>/ingest/folder5P</ds:path>
               <!-- [0] How many levels of directories there will be. Default is 2 -->
               <ds:depth>0</ds:depth>
               <!-- [0] How many characters of an UUID will be taken for a directory's
name. Default is 2 -->
               <ds:granularity>2</ds:granularity>
            </ds:dataStore>
         </ds:dataStores>

      </ds:dataStore>
   </conf:dataStores>

   <!-- [0] Metadata stores are defined below. SolrCloud instance in this scenario -->
   <conf:metadataStores>
      <!-- [0] Solr Metastore -->
      <ms:metadataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ms:SolrMetadataStoreConf" name="solr">
         <!-- [0] Access permissions. Default is READ only -->
         <ms:permission>READ</ms:permission>
```

```
         <ms:permission>WRITE</ms:permission>
         <ms:permission>DELETE</ms:permission>

         <!-- [M] comma separated solr nodes (1 to n nodes) -->
         <ms:hosts>http://solr:8983/solr, http://solr:7574/solr</ms:hosts>

         <!-- [0] Client type to connect solr -->
         <!-- LBHttp: load balanced http access (round-robin) to all specified nodes -->
         <!-- SolrCloud: default, discover solr nodes with zookeeper -->
         <ms:clientType>LBHttp</ms:clientType>

         <!-- [0] Solr username -->
         <ms:user>***</ms:user>
         <!-- [0] Solr password -->
         <ms:password>***</ms:password>

         <!-- [M] name of the solr collection -->
         <ms:collection>cdh</ms:collection>
      </ms:metadataStore>
   </conf:metadataStores>

   <!-- [0] List of exposed STAC Collections. In this example there is one SENTINEL-2
Collection. -->
   <!-- More info on collection specification here: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md -->
   <conf:stacCollections>
      <!-- [M] Collection with its id -->
      <col:stacCollection id="SENTINEL-2">
         <!-- [0] The STAC version the Collection implements. Default is 1.1.0 -->
         <col:stacVersion>1.1.0</col:stacVersion>
         <!-- [0] A short descriptive one-line title for the Collection -->
         <col:title>SENTINEL-2</col:title>
         <!-- [M] Detailed multi-line description to fully explain the Collection -->
         <col:description><![CDATA[The Copernicus Sentinel-2 mission consists of two
polar-orbiting satellites that are positioned in the same sun-synchronous orbit, with a
phase difference of 180°. It aims to monitor changes in land surface conditions. The
satellites have a wide swath width (290 km) and a high revisit time. Sentinel-2 is
equipped with an optical instrument payload that samples 13 spectral bands: four bands at
10 m, six bands at 20 m and three bands at 60 m spatial resolution.]]>
         </col:description>
         <!-- [0] List of comma separated keywords describing the Collection -->
         <col:keywords>copernicus, esa, sentinel, optical</col:keywords>
         <!-- [M] Collection's license(s), either a SPDX License identifier, various if
multiple licenses apply or proprietary
            for all other cases. -->
         <col:license>various</col:license>
         <!-- [0] A list of providers, which may include all organizations capturing or
processing the data or the hosting
            provider. Providers should be listed in chronological order with the most
recent provider being the last element of the list. -->
         <col:providers>
            <col:provider>
```

```
               <!-- [M] The name of the organization or the individual -->
               <col:name>Gael Systems</col:name>
               <!-- [0] Multi-line description to add further provider information such as
processing details for processors
                  and producers, hosting details for hosts or basic contact information. -
->
               <col:description></col:description>
               <!-- [0] Comma separated roles of the provider. Any of licensor, producer,
processor or host -->
               <col:roles>host</col:roles>
               <!-- [0] Homepage on which the provider describes the dataset and publishes
contact information -->
               <col:url>https://www.gael-systems.com/</col:url>
            </col:provider>
         </col:providers>
         <!-- [M] Spatial and temporal extents -->
         <col:extents>
            <!-- [0] Spatial extent. Default is an empty bbox -->
            <col:extent xsi:type="col:spatialExtentConf">
               <!-- [0] Bbox, definition: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md#spatial-extent-object -->
               <col:bbox>[[-180, -90, 180, 90]]</col:bbox>
            </col:extent>
            <!-- [0] Temporal extent. Default is open date range -->
            <col:extent xsi:type="col:temporalExtentConf">
               <!-- [0] Interval, do not use quotes for dates. Definition:
https://github.com/radiantearth/stac-spec/blob/master/collection-spec/collection-
spec.md#temporal-extent-object -->
               <col:interval>[[2015-07-01T00:00:00Z, null]]</col:interval>
            </col:extent>
         </col:extents>
         <!-- [M] Solr filter. Products matching it will be in this collection. -->
         <col:solrFilter>(platformShortName:SENTINEL-2)</col:solrFilter>
      </col:stacCollection>
   </conf:stacCollections>

</conf:configuration>
```

13.3.3.5 S3DataStoreGroup and S3DataStore XML configuration

This example configuration uses a S3DataStoreGroup for storing products and a S3DataStore for storing quicklooks.

It also illustrates how to expose an alternate S3 link in a STAC item assets, for downloading a product or its quicklook. For this link to appear, the property **EXPOSE_IN_STAC** must be added on each DataStore you want to provide a link to, and must be set to **true**.

If you also want product nodes to have an alternate S3 link, the corresponding DataStore must also have the property **SAVE_AS_MULTIPART** set to **true**.

gss.xml example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure DataStores, MetadataStores, STAC Collections and
system parameters of the API -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:conf="fr:gael:gss:stac:configuration" xmlns:ds="fr:gael:gss:core:datastore"
   xmlns:ms="fr:gael:gss:core:metadatastore" xmlns:dl="fr:gael:gss:odata:download"
   xmlns:metrics="fr:gael:gss:core:metrics" xmlns:col="fr:gael:gss:stac:collection">

   <!-- [M] Access to the PostgreSQL database -->
   <!-- The attribute poolSize is optional (default 10) and is the max opened connection
to the database -->
   <conf:database jdbcUrl="jdbc:postgresql://postgres:5432/postgres" login="postgres"
password="password" poolSize="5" />

   <!-- [0] Configuration tweaks -->
   <!-- directDownloadLink: enable the direct download from cloud stores (swift) when
using $value. Default is true. -->
   <!-- quotaDisabled: disable quotas. Default is false. -->
   <conf:download directDownloadLink="false" quotaDisabled="true" />

   <!-- Stores are configured below. In this scenario, Time Based stores are used to store
products and quicklooks. A Solr index is used to store metadata -->
   <conf:dataStores>

      <!-- [0] Configuration tweaks for S3 behavior -->
      <!-- [0] maxConnections: maximum simultaneous connections opened to the S3 storage.
Default is 50 -->
      <!-- [0] urlValiditySeconds: validity time in seconds a direct download link will be
valid. Default is 15 -->
      <ds:s3Configuration maxConnections="50" urlValiditySeconds="15" />
 <ds:s3Credentials name="credentialsS3">
    <ds:accessKey>***</ds:accessKey>
    <ds:secretKey>***</ds:secretKey>
    <ds:region>gra</ds:region>
    <ds:endpoint>https://s3.gra.io.cloud.ovh.net</ds:endpoint>
 </ds:s3Credentials>

 <!-- [0] First s3 group that store products as packages -->
 <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ds:s3DataStoreGroupConf" name="s3Group">
    <!-- [0] Access permissions. Default is read-only -->
    <ds:permission>WRITE</ds:permission>
    <ds:permission>READ</ds:permission>
    <ds:permission>DELETE</ds:permission>

    <!-- [0] Different properties managing the behavior of the store -->
    <ds:properties>
```

```
       <!-- [0] if true, products will be stored by their name. Default is false (stored
by uuid) -->
       <ds:property>
          <ds:name>STORE_BY_NAME</ds:name>
          <ds:value>true</ds:value>
       </ds:property>
       <!-- [0] if true, products will be stored as multiparts. Default is false (stored
as package) -->
       <ds:property>
          <ds:name>SAVE_AS_MULTIPART</ds:name>
          <ds:value>false</ds:value>
       </ds:property>
       <!-- [0] if true, an alternate S3 link will be proposed in the assets of the
product in STAC -->
       <ds:property>
          <ds:name>EXPOSE_IN_STAC</ds:name>
          <ds:value>true</ds:value>
       </ds:property>
    </ds:properties>
    <!-- [M] credentials to access the containers (defined before datastores) -->
    <ds:credentials>credentialsS3</ds:credentials>
    <!-- [M] filter which products based on their name are accepted. ".*" means all
products -->
    <ds:filter>.*</ds:filter>
    <ds:containerPattern xsi:type="ds:mapperContainerPatternConf">
       <ds:patternMapper>S1.*:s3-sentinel-1,S2.*:s3-sentinel-2,S3.*:s3-sentinel-3,S5.*:s3-
sentinel-5P,.*:s3-other
       </ds:patternMapper>
    </ds:containerPattern>
 </ds:dataStore>

 <!-- [0] This store is a swift container that is used to store quicklooks -->
 <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:s3DataStoreConf"
    name="s3QL">
    <!-- [0] Access permissions. Default is read-only -->
    <ds:permission>WRITE</ds:permission>
    <ds:permission>READ</ds:permission>
    <ds:permission>DELETE</ds:permission>
    <ds:properties>
       <!-- [0] if true, this store will be able to store attached files. Default is false
-->
       <ds:property>
          <ds:name>STORE_ATTACHED_FILES</ds:name>
          <ds:value>true</ds:value>
       </ds:property>
       <!-- [0] if true, products will be stored by their name. Default is false (stored
by uuid) -->
       <ds:property>
          <ds:name>STORE_BY_NAME</ds:name>
          <ds:value>true</ds:value>
       </ds:property>
        <!-- [0] if true, an alternate S3 link will be proposed in the assets of the
```

```
product in STAC -->
       <ds:property>
          <ds:name>EXPOSE_IN_STAC</ds:name>
          <ds:value>true</ds:value>
       </ds:property>
    </ds:properties>
    <!-- [M] credentials to access the container (defined before datastores) -->
    <ds:credentials>credentialsS3</ds:credentials>
    <!-- [M] name of the container -->
    <ds:bucket>s3-quicklooks</ds:bucket>
    <!-- [0] If defined, products will be stored with a prefix in buckets -->
    <!-- the prefix is extracted from the product name, and can be combination of -->
    <!-- instrument, productType, sensing date (year, month, day) -->
    <ds:prefixLocation>instrument/productType/year/month/day</ds:prefixLocation>
 </ds:dataStore>

   </conf:dataStores>

   <!-- [0] Metadata stores are defined below. SolrCloud instance in this scenario -->
   <conf:metadataStores>
      <!-- [0] Solr Metastore -->
      <ms:metadataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ms:SolrMetadataStoreConf" name="solr">
         <!-- [0] Access permissions. Default is READ only -->
         <ms:permission>READ</ms:permission>
         <ms:permission>WRITE</ms:permission>
         <ms:permission>DELETE</ms:permission>

         <!-- [M] comma separated solr nodes (1 to n nodes) -->
         <ms:hosts>http://solr:8983/solr, http://solr:7574/solr</ms:hosts>

         <!-- [0] Client type to connect solr -->
         <!-- LBHttp: load balanced http access (round-robin) to all specified nodes -->
         <!-- SolrCloud: default, discover solr nodes with zookeeper -->
         <ms:clientType>LBHttp</ms:clientType>

         <!-- [0] Solr username -->
         <ms:user>***</ms:user>
         <!-- [0] Solr password -->
         <ms:password>***</ms:password>

         <!-- [M] name of the solr collection -->
         <ms:collection>cdh</ms:collection>
      </ms:metadataStore>
   </conf:metadataStores>

   <!-- [0] List of exposed STAC Collections. In this example there is one SENTINEL-2
Collection. -->
   <!-- More info on collection specification here: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md -->
   <conf:stacCollections>
      <!-- [M] Collection with its id -->
```

```
      <col:stacCollection id="SENTINEL-2">
         <!-- [0] The STAC version the Collection implements. Default is 1.1.0 -->
         <col:stacVersion>1.1.0</col:stacVersion>
         <!-- [0] A short descriptive one-line title for the Collection -->
         <col:title>SENTINEL-2</col:title>
         <!-- [M] Detailed multi-line description to fully explain the Collection -->
         <col:description><![CDATA[The Copernicus Sentinel-2 mission consists of two
polar-orbiting satellites that are positioned in the same sun-synchronous orbit, with a
phase difference of 180°. It aims to monitor changes in land surface conditions. The
satellites have a wide swath width (290 km) and a high revisit time. Sentinel-2 is
equipped with an optical instrument payload that samples 13 spectral bands: four bands at
10 m, six bands at 20 m and three bands at 60 m spatial resolution.]]>
         </col:description>
         <!-- [0] List of comma separated keywords describing the Collection -->
         <col:keywords>copernicus, esa, sentinel, optical</col:keywords>
         <!-- [M] Collection's license(s), either a SPDX License identifier, various if
multiple licenses apply or proprietary
            for all other cases. -->
         <col:license>various</col:license>
         <!-- [0] A list of providers, which may include all organizations capturing or
processing the data or the hosting
            provider. Providers should be listed in chronological order with the most
recent provider being the last element of the list. -->
         <col:providers>
            <col:provider>
               <!-- [M] The name of the organization or the individual -->
               <col:name>Gael Systems</col:name>
               <!-- [0] Multi-line description to add further provider information such as
processing details for processors
                  and producers, hosting details for hosts or basic contact information. -
->
               <col:description></col:description>
               <!-- [0] Comma separated roles of the provider. Any of licensor, producer,
processor or host -->
               <col:roles>host</col:roles>
               <!-- [0] Homepage on which the provider describes the dataset and publishes
contact information -->
               <col:url>https://www.gael-systems.com/</col:url>
            </col:provider>
         </col:providers>
         <!-- [M] Spatial and temporal extents -->
         <col:extents>
            <!-- [0] Spatial extent. Default is an empty bbox -->
            <col:extent xsi:type="col:spatialExtentConf">
               <!-- [0] Bbox, definition: https://github.com/radiantearth/stac-
spec/blob/master/collection-spec/collection-spec.md#spatial-extent-object -->
               <col:bbox>[[-180, -90, 180, 90]]</col:bbox>
            </col:extent>
            <!-- [0] Temporal extent. Default is open date range -->
            <col:extent xsi:type="col:temporalExtentConf">
               <!-- [0] Interval, do not use quotes for dates. Definition:
https://github.com/radiantearth/stac-spec/blob/master/collection-spec/collection-
spec.md#temporal-extent-object -->
```

```
                <col:interval>[[2015-07-01T00:00:00Z, null]]</col:interval>
            </col:extent>
        </col:extents>
        <!-- [M] Solr filter. Products matching it will be in this collection. -->
        <col:solrFilter>(platformShortName:SENTINEL-2)</col:solrFilter>
    </col:stacCollection>
  </conf:stacCollections>

</conf:configuration>
```

STAC collections are defined in the gss.xml as in the example above. As every product is stored in Solr, a STAC collection is a logical subset of those products, defined by a Solr filter. This filter can be a combination of metadata_name:metadata_value pairs with AND, OR operators. A metadata_value can start/end with the wildcard '*'. Make sure to put parenthesis around the filter to avoid any side-effect.

Examples:

A collection for all Sentinel-2 products:

```
<col:solrFilter>(platformShortName:SENTINEL-2)</col:solrFilter>
```

A collection for Sentinel-5 products of specific classes:

```
<col:solrFilter>(platformShortName:SENTINEL-5P AND productClass:OFFL OR productClass:NRTI
)</col:solrFilter>
```

A collection for all Zarr products:

```
<col:solrFilter>(name:*.zarr)</col:solrFilter>
```

Default quotas

If no quotas are assigned to a user, default ones will be used. These default values can be customized.

If using the gss.xml file:

```
<conf:download directDownloadLink="false" quotaDisabled="false">
 <dl:quota name="TOTAL_DOWNLOAD" value="300000000" durationMinutes="60" />
 <dl:quota name="PARALLEL_DOWNLOAD" value="2" />
 <dl:quota name="TOTAL_DOWNLOAD_LINK" value="10" durationMinutes="10" />
</conf:download>
```

If using the db configuration, add in application.properties:

```
##custom default quotas values if no quotas assigned to a user
#number of concurrent download, default = 2
download.quota.parallel.download.value = 2
```

```
#max volume of download in bytes/min, default is 3GB/hour
download.quota.total.download.value = 3000000000
download.quota.total.download.duration = 60
#max number of download links in numbers/min, default is 10/10 min
download.quota.total.download.link.value = 10
download.quota.total.download.link.duration = 1
```

Note that if authentication is disabled and quotas are enabled, a default user named JohnDoe will be used in all requests. This dummy user comes with specific quota values configured in database. You can edit them or remove them if you want to use default quotas as above. These quotas are defined in the table quota with user_id='JohnDoe'.

14. STAC API Catalog Access Endpoints

STAC API endpoints are the URLs that you can use to interact with a STAC catalog, collection, or item. The endpoints provide various operations for searching, filtering, and retrieving data. Here are the most common STAC API endpoints:

- **Root endpoint:** This is the base URL of the STAC API. It returns a JSON object with information about the API, such as the version and available endpoints.
- **Collection endpoint:** This endpoint allows users to retrieve information about a specific STAC collection.
- **Item endpoint:** This endpoint allows users to retrieve information about a specific STAC item.
- **Search endpoint:** This endpoint allows users to search for STAC items based on various criteria, such as bounding box, time interval, or other keywords. This endpoint is accessible with GET and POST requests.

For a detailed information about stac-api spec, endpoints and extensions, please consider the following specifications: https://stacspec.org/en/about/stac-spec/ and https://github.com/radiantearth/stac-api-spec

14.1 STAC API Collection Query Examples

The STAC-API collection endpoint allows users to query STAC collections using various parameters and filters to retrieve specific datasets.

For a detailed description of STAC API Collection Query Examples, please refer to the STAC-ICD document below:

- RD-5 - Collaborative Data Hub Software GSS STAC Catalog Access ICD.

14.2 STAC API Search Query Examples

The STAC-API search endpoint allows users to search and retrieve items based on different criteria.

For a detailed description of STAC API Search Query Examples, please refer to the STAC-ICD document below:

- RD-5 - Collaborative Data Hub Software GSS STAC Catalog Access ICD.

15. Log Aggregator

CDH components can send their logs to a Kafka topic. The logs can then be processed by consuming messages from this topic. No component is provided to read these messages.

To configure the log aggregator, the users must modify the log4j2.xml file. The log4j2 XML file is a configuration file that can be modified at will, to specify the logging levels and appenders (where logs will be sent).

15.1 Log Aggregator Configuration

The log aggregator configuration for all the components are identical, users can adapt the provided configuration, following is an example of the log appender in log4j2 XML file:

```
<Kafka name="kafka" topic="cdh_log">
   <!-- add referenced environment variable containing the kafka insance -->
   <Property name="bootstrap.servers">kafka-1:9092,kafka-2:9092</Property>
   <Property name="syncSend">false</Property>
   <PatternLayout pattern="${pattern}" />
</Kafka>
```

If you use Kafka authentication, add these additional properties with your configured username and password:

```
<Property name="security.protocol">SASL_PLAINTEXT</Property>
<Property name="sasl.mechanism">PLAIN</Property>
<Property name="sasl.jaas.config">org.apache.kafka.common.security.plain.PlainLoginModule
required username="kafka" password="pass123";</Property>
```

Following is an example of the CDH-Admin-api log4j2 XML file containing the log appender:

```
< <?xml version="1.0" encoding="UTF-8"?>

<Configuration>
   <Properties>
```

```
      <Property name="pattern">
[${sys:application_name}][${sys:build_version}][%d{DEFAULT}][%X{ingesterName}] %m
(%file:%line - %t)%n%throwable </Property>
   </Properties>
   <Appenders>
     <Kafka name="kafka" topic="cdh_log">
   <!-- add referenced environment variable containing the kafka insance -->
   <Property name="bootstrap.servers">kafka-1:9092,kafka-2:9092</Property>
   <Property name="syncSend">false</Property>
   <PatternLayout pattern="${pattern}" />
     </Kafka>
      <Console name="stdout" target="SYSTEM_OUT">
         <PatternLayout pattern="${pattern}" />
         <Filters>
            <ThresholdFilter level="DEBUG" />
            <ThresholdFilter level="WARN" onMatch="DENY" onMismatch="NEUTRAL" />
         </Filters>
      </Console>
      <Console name="stderr" target="SYSTEM_ERR">
         <PatternLayout pattern="${pattern}" />
         <Filters>
            <ThresholdFilter level="WARN" />
         </Filters>
      </Console>
      <RollingFile name="RollingFile" fileName="logs/cdh-admin.log"
         filePattern="logs/cdh-admin-%d{yyyy-MM-dd}.log">
         <PatternLayout>
            <Pattern>${pattern}</Pattern>
         </PatternLayout>
         <Policies>
            <TimeBasedTriggeringPolicy interval="1" modulate="true" />
         </Policies>
         <Filters>
            <ThresholdFilter level="DEBUG" />
         </Filters>
      </RollingFile>
   </Appenders>ev
   <Loggers>
      <logger name="fr.gael" level="info" />
      <Root level="info">
         <AppenderRef ref="stderr" />
         <AppenderRef ref="stdout" />
         <AppenderRef ref="kafka" />
         <AppenderRef ref="RollingFile" />
      </Root>
   </Loggers>
</Configuration>
```

15.2 Topic Customization

As there are many logs, it can be a good idea to create or modify the topic where logs are stored. In Kafka a topic can have a max retention time (in ms) and a max size (in bytes). Messages older than **retention.ms** will be purged, and if the topic exceeds **retention.bytes,** oldest messages will be removed to maintain this maximum size.

15.2.1 Topic Creation

On the machine hosting one kafka broker, execute the following command to open a shell terminal inside the Docker container:

```
docker exec -it <container_name> /bin/bash
```

Create the topic with custom configuration:

```
/bin/kafka-topics --bootstrap-server localhost:9092 --create --topic <topic_name> --partitions 10 --replication-factor 1 --config retention.ms=<retention_ms> --config retention.bytes=<retention_bytes>
```

15.2.2 Topic Alteration

On the machine hosting one kafka broker, execute the following command to open a shell terminal inside the Docker container:

```
docker exec -it <container_name> /bin/bash
```

Alter the topic with custom configuration:

```
/bin/kafka-configs --bootstrap-server localhost:9092 --entity-type topics --entity-name <topic_name> --alter --add-config retention.ms=<retention_in_ms> --add-config retention.bytes=<retention_bytes>
```

16. CDH Version update

To use a new version of the CDH distribution, you should run the toolbox update tool. This will update the database schema and Solr schema if there are changes to apply. Otherwise, it will do nothing. All updates are incremental; you can update from any version to any more recent version.

There are multiple ways to run the toolbox update tool:
- If you are using the Zip distribution of the cdh-toolbox, refer to sections 4.2.1 and 4.2.2.
- If you are using cdh-compose, refer to section 5.4.1.
- You can also directly use the cdh-toolbox docker image, with the command:

```
docker run -it --network="host" gaeldockerhub/cdh-toolbox:3.4.0 -e
JDBC_URL=jdbc:postgresql://localhost:5432/postgres -e DB_LOGIN:postgres -e
DB_PASSWORD=password –e SOLR_URL=http://localhost:8983/solr -e SOLR_CORE=cdh
```

Use *–network="host"* if you are running the database and Solr on the local machine.
Environment variables are:

Variable Name	Variable value	Default value
JDBC_URL	JDBC Database URL	jdbc:postgresql://localhost:5432/postgres
DB_LOGIN	Database user	postgres
DB_PASSWORD	Database password	password
SOLR_URL	Solr URL	http://localhost:8983/solr
SOLR_CORE	Solr collection name	cdh
SOLR_USER	Solr username	
SOLR_PASSWORD	Solr password	

17. Customize JVM args

You can add or change JVM arguments for each component with the environment variable **JAVA_OPTS**.

For example if you want to increase the JVM heap space for cdh-catalogue, and you launch it with docker compose:

```
services:
  catalogue:
    image: "gaeldockerhub/cdh-catalogue:3.4.0"
    ports:
     - 8081:8081
    volumes:
     - ./gss-catalogue.xml:/catalogue/etc/gss.xml
```

```
 - ./application.properties:/catalogue/etc/application.properties
 - ./log4j2.xml:/catalogue/etc/log4j2.xml
 - ~/test/data/:/catalogue/hfs
networks:
 - backend
environment:
 - JAVA_OPTS=-Xms256m -Xmx4g
```