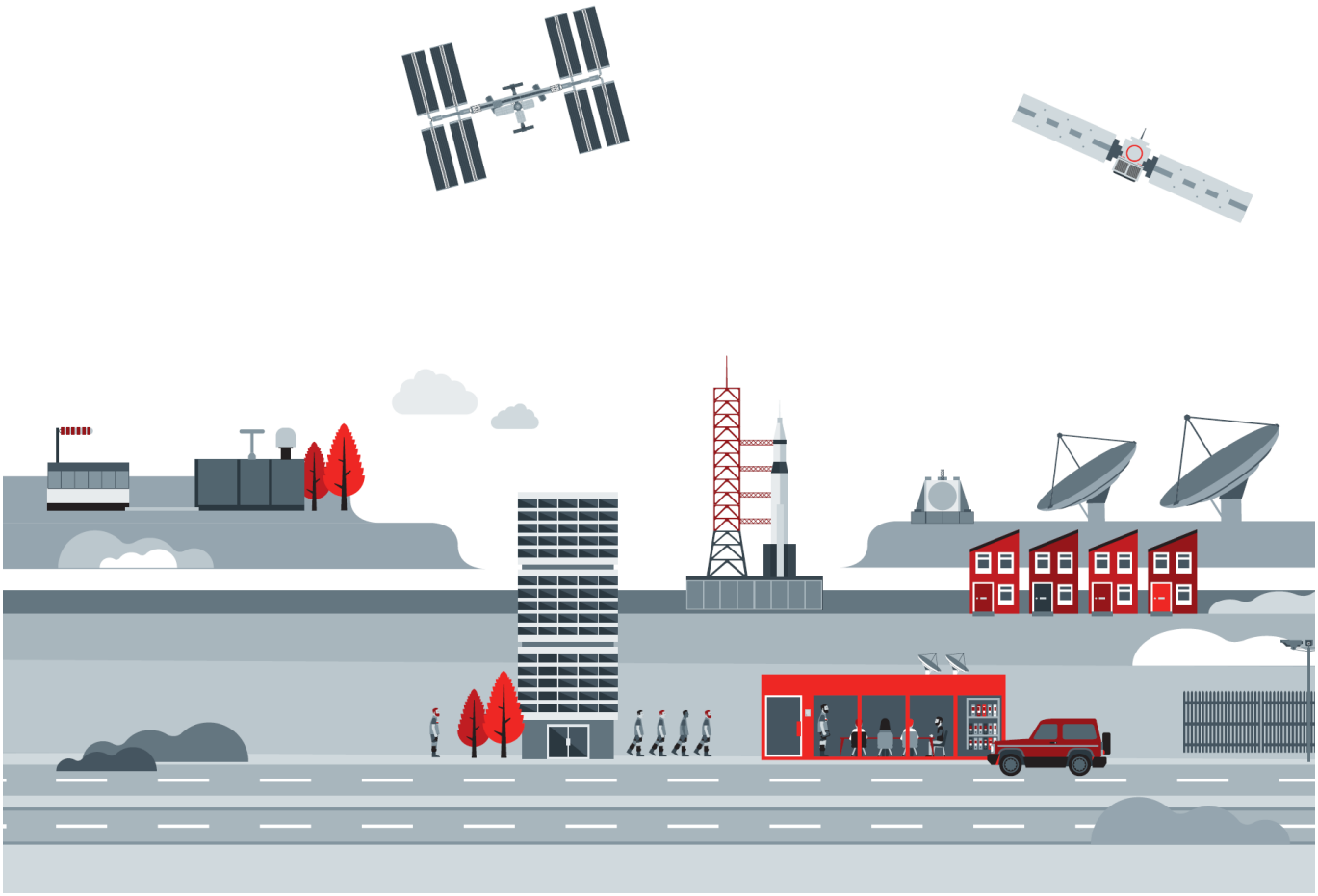
GAEL SYSTEMS
Earth Observation Data Solutions

Serco Business

serco

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

GSS Administration Manual



Reference: GAEL-P311-GSS-CDH-Administration Manual
Date: 24/05/2024
Version: 1.6.5

Role/Title	Name	Signature	Date
Author	GAEL Team		24/05/2024
Verified	Federica Volpi		24/05/2024
Approved	Elisabetta Giuliani		24/05/2024

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 □ Capitalized letters of headings □ Reformat	

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.

In particular, 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	DHS Milestone
V1.6.5	DHS#6 (1.4.3)

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 , v 1.7.2
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, v1.3
RD-4	COPE-SERCO-TN-21-1229 - Keycloak Installation and Configuration Manual	COPE-SERCO-TN-21-1229 - Keycloak Installation and Configuration Manual, v2.4

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 Secure
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:

- 1 Producer and 1 Consumer for Ingestion
- 1 Catalogue
- 1 Admin API
- 1 Notification Consumer

3. Other tools for GSS

We need other tools to install and manage GSS:

- docker: 20.10.12 (or later)
- docker-compose: 1.29.0 (or later)

4. Toolbox

The toolbox is a set of scripts which allows to initialize 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)
- docker-compose: 1.29.0 (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 8.0 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.gaelsystems.com/repository/thirdparty/fr/gael/gss/cdh-toolbox/1.4.3/cdh-toolbox-1.4.3.zip

The zip archive structured is :

```
cdh-toolbox
├── db
│   └── database_updater.sh
├── etc
│   └── log4j2.xml
├── lib
└── logs└── 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.gaelsystems.com/repository/thirdparty/fr/gael/gss/cdh-toolbox/1.4.3/cdh-toolbox-1.4.3.zip
```

Unzip the file with the command:

```
unzip cdh-toolbox-1.4.3.zip
```

4.2.1 Database Updater

Enter in the "cdh-toolbox-1.4.3" 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** , the **cdh-catalogue** and **cdh-admin-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-1.4.3"* 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 script, edit the file *solr_configuration.sh.* Set the configuration file with the command:

```
set /path_to_toolbox-1.4.3/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
```

4.3 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 the 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")
containers	varchar (1024)			Container name defined for Swift storage of products in error. It is null for folder storage
credentials_name	varchar (128)			Credential's name which is used in case of swift storage. It is null for folder storage.
error_location	varchar (4096)			Path to store products in error. It is null for swift storage.
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

generate_trace_task represents the generation of trace task used by consumers

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 (,)
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_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 (,)
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)

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

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
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

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

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"

job_deletion_products represent the deletion job.

Column	Type	Constraint	Indexed	Description
id	uuid	PK	yes	autogenerated
message	varchar (512)			A comment to describe the deletion
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 :** - PAUSED - CANCELED - CREATED - RUNNING
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 have to be updated to fit operational scenarios.

5.1 Pre-requesites

We recommend to install:

- docker: 20.10.12 (or later)
- docker-compose: 1.29.0 (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/1.4.3/cdh-compose-1.4.3.zip The archive must be downloaded and unzipped in order to launch the components.

```
/cdh-compose-1.4.3
├── admin/
├── backend/
├── catalogue/
├── notification/
└── ingest/
```

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

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_1.4.3/S1:/ingest/folder1
    - /mnt/dhsnas/IVV_PLATFORM/GSS/DEST_FOLDER_GSS_1.4.3/error:/ingest/error
    - /mnt/dhsnas/IVV_PLATFORM/GSS/DEST_FOLDER_GSS_1.4.3/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)
- 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 with nohup

```
nohup docker-compose up &
```

You can check the logs with:

```
docker-compose logs -f
```

If the database and solr schema have to be created/updated, run the dedicated script:

```
./init.sh
```

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
```

5.5 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 Scihub 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-1.4.3.zip file. These files can be used for ingester's configuration. They fit to main of operational scenarios known. The files are stored into **"operational-samples"** directory

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 (with OData source)

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 make 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 with nohup.

```
nohup docker-compose up &
```

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 120
```

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 120
```

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-1.4.3.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. We need to add these properties since release 1.3.0

```
### Database configuration.
# WARNING: This part is mandatory even if you use XML configuration
spring.datasource.url=jdbc:postgresql://<server.url>:<server.port>/<database.name>
spring.datasource.username=<user.name>
spring.datasource.password=<user.password>
spring.datasource.driver-class-name=org.postgresql.Driver
# Max pool connexion size
spring.datasource.hikari.maximum-pool-size=3
#db.socketTimeout=1000
#db.networkTimeout=5000

# To encrypt secret information in database (password for example)
secret.key.password=<encryption.value>
## End of mandatory part
```

5.6.1.2 Admin API Configuration

Go to the **catalogue** directory.

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

```
## OAuth-2.0 Authentication configuration
```

```
# To enable oauth2 authentication, if auth = true and uncomment all
spring.security.oauth2... properties
auth = false

#spring.security.oauth2.resourceserver.jwt.issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>
#spring.security.oauth2.resourceserver.jwt.jwk-issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>/protocol/openid-connect/certs

### IMPORTANT ### Please ensure that the 'keycloak.client', 'keycloak.role' and
'cors.origins' are not commented out !
# if auth=true, please provide the 'keycloak Client ID' and 'keycloak Client Role.'
keycloak.client = <client-id>
keycloak.role = <client-role>

## To enable Cross Origin Requests, for example to allow the Admin UI to call this API
cors.origins = http://<localhost>:<8081>

# 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

## Database configuration.
# WARNING: This part is mandatory even if you use XML configuration
spring.datasource.url=jdbc:postgresql://<server.url>:<server.port>/<database.name>
spring.datasource.username=<user.name>
spring.datasource.password=<user.password>
spring.datasource.driver-class-name=org.postgresql.Driver
# Max pool connexion size
spring.datasource.hikari.maximum-pool-size=3
#db.socketTimeout=1000
#db.networkTimeout=5000
# To encrypt secret information in database (password for example)
secret.key.password=<encryption.value>
## End of mandatory part

## CDH Configuration
## Only used when stores are configured into DB
## Datastore's names
cdh.useDbConfiguration=true
cdh.datastores=datastore-name1;datastore-name2
## Metadatastore's names
cdh.metadatastores=metadatastore.name1;metadatsore.name2

## Swift Configuration
#swiftConfiguration.segmentSizeMB=100
#swiftConfiguration.retries=5
#swiftConfiguration.retryDelayMs=50
#swiftConfiguration.maxConnections=20

## ENABLE / DISABLE EVICTION BY TIME
process.evictionByTime=false
```

```
# directDownloadLink : enable the direct download from # cloud stores (swift) when using
$value. Default is true. --
# quotaDisabled : disable quotas. Default is false.
download.directDownloadLink=false
download.quotaDisabled=false

# The following attributes used to define the properties to perform deletion
# Configuration to have in the first launch
deletion.kafka.hosts=<kafka-1>:<port>;<kafka-2>:<port>
deletion.kafka.topic=deletion
deletion.datastores=<datastore-name1;datastore-name2>
deletion.metadatastores=<metadata store[s] name>
```

The stores are previously created in Admin API.

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 with nohup

```
nohup docker-compose up &
```

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 to the database set in **backend** part. Authentication is disabled.

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 with nohup

```
nohup docker-compose up &
```

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

IMPORTANT
In a distributed environment, please add the "Port" section to the "docker-compose.yml" file as follows:
ports: - <Port_value>:<Port_value>

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 with nohup

```
nohup docker-compose up &
```

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
```

6. Admin API

GSS Admin API is named CDH-Admin. It is a REST API exposing different components to configure GSS components (ingests, user's quotas, datastores, metadatastores). They are 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)
- docker-compose: 1.29.0 (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/1.4.3/cdh-admin-api-1.4.3.zip
The archive must be downloaded and unzipped in order to launch the AdminAPI.

The zip archive is presented as follows :

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

The **application.properties** file of Admin API is used for the Keycloak server configuration and the embedded http server. The content of this file is described further in the sample file: *application.properties.sample*. 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:${project.version}`

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:${project.version}`

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:/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

Go to **admin** directory.

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

```
## OAuth-2.0 Authentication configuration

# To enable oauth2 authentication, set auth = true and uncomment all
spring.security.oauth2... properties
auth = false

#spring.security.oauth2.resourceserver.jwt.issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>
```

```
#spring.security.oauth2.resourceserver.jwt.jwk-issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>/protocol/openid-connect/certs

### IMPORTANT ### Please ensure that the 'keycloak.client', 'keycloak.role' and
'cors.origins' are not commented out !
# if auth=true, please provide the 'keycloak Client ID' and 'keycloak Client Role.'
keycloak.client = <client-id>
keycloak.role = <client-role>

## To enable Cross Origin Requests, for example to allow the Admin UI to call this API
cors.origins = http://<localhost>:<8082>

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

## Server configuration

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

# Configure context path (optional)
# server.servlet.contextPath = /gss-admin-api

############ PRODUCT DELETION
# To perform perform deletion of products, the configuration of stores need to be in
database.

# Used to define the properties to perform deletion

# Configuration to have in the first launch
deletion.kafka.hosts=<kafka.host>:<kafka.port>
deletion.kafka.topic=<kafka topic name>
deletion.datastores=datastore-name1;datastore-name2;datastore-name3
deletion.metadatastores=metadatastore-name1;metadatastore-name2

# The two below parameter are used to not have custom errors pages
server.error.whitelabel.enabled = false
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.web.servlet.error.Erro
rMvcAutoConfiguration

# Configure the management of the HTTP Headers between the web server and the application.
# Possible values:
# - NATIVE = use the web server support for forwaded headers. It allows us to
# to handle HTTPS queries inside the application and do not modify what the server
received.
# - FRAMEWORK : Spring will handle forwarded headers. It is possible that some headers
will be
# updated according to the framework.
server.forward-headers-strategy = NATIVE

# Used to encrypt secured information (Datastore Admin API and Ingester Admin API)
```

```
# This password must be strong (8 characters min with upper case, digit and special
characters
# like *,#)
secret.key.password = <encryption.value>

### Database configuration.
# WARNING: This part is mandatory even if you use XML configuration
spring.datasource.url=jdbc:postgresql://<server.url>:<server.port>/<database.name>
spring.datasource.username=<user.name>
spring.datasource.password=<user.password>
spring.datasource.driver-class-name=org.postgresql.Driver
# Max pool connexion size
spring.datasource.hikari.maximum-pool-size=3
#db.socketTimeout=1000
#db.networkTimeout=5000
```

6.3.2 Admin-API Keycloak Configuration

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

- Authentication via a web API.

6.3.2.1 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": [
            "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

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 (**application.properties.sample**). 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.

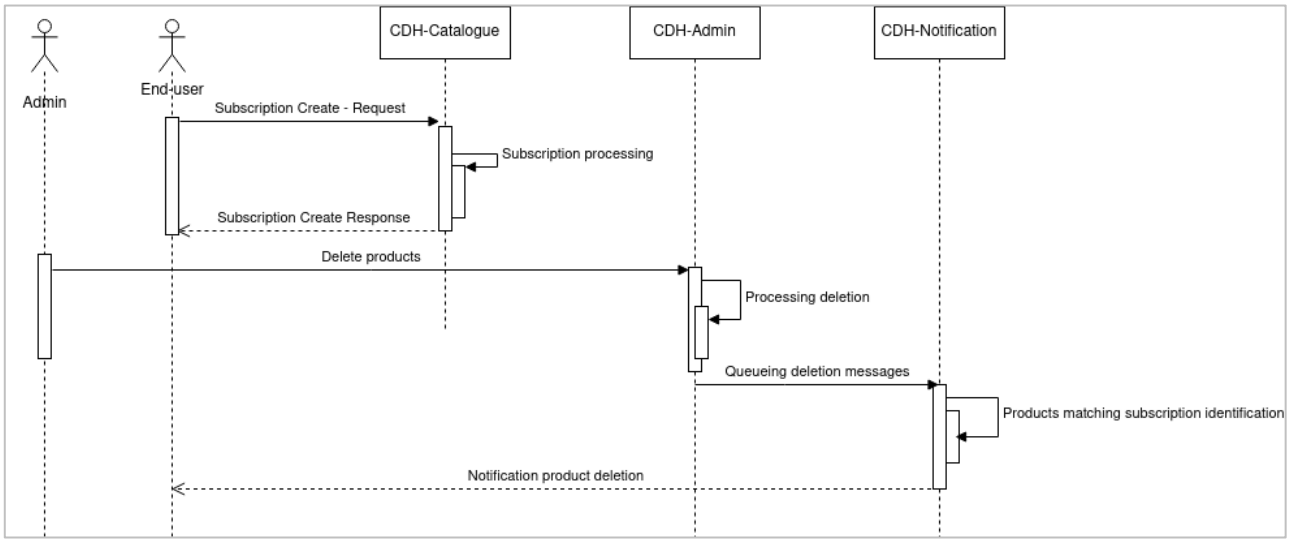
Important

`# 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 like below :



- 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.

8. Ingest

Ingest any products by following a consumer/producer paradigm using Kafka. Products will be stored inside configurable DataStores (HFS, Swift) 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)
- docker-compose: 1.29.0 (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/1.4.3/cdh-ingest-1.4.3.zip

The zip archive structure is :

```
cdh-ingest/
    ├── etc/
    │   ├── odata-csc/
    │   │   ├── gss-consumer.xml
    │   │   └── gss-producer.xml
    │   │
    │   ├── odata-dhus/
    │   │   ├── gss-consumer.xml
    │   │   └── gss-producer.xml
    │   │
    │   ├── folder/
    │   │   ├── gss-consumer.xml
    │   │   └── gss-producer.xml
    │   │
    │   ├── obj-storage/
    │   │   ├── gss-consumer.xml
    │   │   └── gss-producer.xml
    │   │
    │   ├── database-conf/
    │   │   └── database-configuration-for-ingestion.properties
    │   │
    │   ├── operational-samples/
    │   │   ├── 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-swift.xml
    │   │   │   └── gss_producer.xml
    │   │   │
    │   │   └── source-folder/
    │   │       ├── gss_consumer-dest-folder.xml
    │   │       ├── gss_consumer-dest-swift.xml
    │   │       └── gss_producer.xml
    │   │
    │   ├── JSON-samples/
    │   │   ├── SWIFT-credentials/
    │   │   │   └── swift_credentials.json
    │   │   │
    │   │   ├── Producer/
    │   │   │   ├── producer_cdse_source.json
    │   │   │   ├── producer_dhus_source.json
    │   │   │   ├── producer_folder_source.json
    │   │   │   └── producer_GSS_source.json
```

```
        ├── Consumer/
        │   ├── CDSE/
        │   │   ├── consumer_cdse_source_dest_folder_error_folder.json
        │   │   └── consumer_cdse_source_dest_swift_error_swift.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
        │   │
        │   └── 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
        │   │
        │   └── SWIFT/
        │       ├── datastore_quicklook.json
        │       └── timebased_datastore.json
        │
        ├── Metadatastore/
        │   └── metadatastore.json
        │
        │
        └── Quota/
            └── quota_user.json

    └── log4j2.xml
├── lib/
├── logs/
├── start.sh
└── stop.sh
```

- odata-dhus/
  - *gss-consumer.xml* : consumer example that retrieve products from a DHuS OData v4 endpoint
  - *gss-producer.xml* : producer example that scans a DHuS OData v4 endpoint periodically
- odata-csc/
  - *gss-consumer.xml* : consumer example that retrieve products from a CSC OData v4 endpoint
  - *gss-producer.xml* : producer example that scans a CSC OData v4 endpoint periodically
- obj-storage/

- *gss-consumer.xml* : consumer example that retrieve products from a swift object storage
- *gss-producer.xml* : producer example that scans swift object storage periodically

- folder/
  - *gss-consumer.xml* : consumer example that retrieve products from folder
  - *gss-producer.xml* : producer example that scans a folder

- database-conf/
  - *database-configuration-for-ingestion.properties* : example of configuration fole to add to read configuration of ingesters and datastores from the database.

- operational-samples/
  - *source-csc/*
    - *gss_consumer-dest-folder.xml*: consumer example that ingests from a CSC OData v4 endpoint into a folder
    - *gss_consumer-dest-swift.xml*: consumer example that ingests from a CSC OData v4 endpoint into a swift object storage
    - *gss_producer.xml*: producer example that ingests from a CSC OData v4 endpoint
  - *source-dhus/*
    - *gss_consumer-dest-folder.xml*: consumer example that ingests from a DHuS OData v4 endpoint into a folder
    - *gss_consumer-dest-swift.xml*: consumer example that ingests from a DHuS OData v4 endpoint into a swift object storage
    - *gss_producer.xml*: producer example that ingests from a DHuS OData v4 endpoint
  - *source-folder/*
    - *gss_consumer-dest-folder.xml*: consumer example that ingests from a folder into a folder
    - *gss_consumer-dest-swift.xml*: consumer example that ingests from a folder into a swift object storage
    - *gss_producer.xml*: producer example that ingests from a folder

- JSON-samples/
  - `SWIFT-credentials/`
    - *swift_credentials.json :* credentials example that enables ingest access to an object-storage.
  - *Producer/*
    - *producer_cdse_source.json :* producer example that ingests from a cdse source
    - *producer_dhus_source.json :* producer example that ingests from a DHuS OData v4 endpoint
    - *producer_folder_source.json :* producer example that ingests from a folder
    - *producer_GSS_source.json :* producer example that ingests from a CSC OData v4 endpoint
  - `Consumer/`

- DHuS/
  - *consumer_dhus_source_dest_folder_error_folder.json :* consumer example that ingests from a DHuS OData v4 endpoint into a folder
  - *consumer_dhus_source_dest_swift_error_swift.json :* consumer example that ingests from a DHuS OData v4 endpoint into a swift object storage
- Folder/
  - *consumer_folder_source_dest_folder_error_folder.json :* consumer example that ingests from a folder into a folder
  - *consumer_folder_source_dest_swift_error_swift.json :* consumer example that ingests from a folder into a swift object storage
- CSC/
  - *consumer_GSS_source_dest_folder_error_folder.json :* consumer example that ingests from a CSC OData v4 endpoint into a folder
  - *consumer_GSS_source_dest_swift_error_swift.json :* consumer example that ingests from a CSC OData v4 endpoint into a swift object storage

- `Datastore/`
  - HFS/
    - *datastore_quicklook.json :* HFS datastore example to store quicklooks
    - *timebased_datastore.json :* Timebased HFS datastire example to store products
  - SWIFT/
    - *datastore_quicklook.json :* Swift datastore example to store quicklooks
    - *timebased_datastore.json :* Timebased Swift datastire example to store products
- `Metadatastore/`
  - *metadatastore.json :* Metadata store example to store metadatas
- `Quota/`
  - *quota_user.json :* quota example to define user quotas

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 **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 the ingester and use *stop.sh* to gracefully stop it.

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

8.2.2 Docker Image

Available on Docker Hub -> `docker pull registry.hub.docker.com/gaeldockerhub/cdh-ingest:{project.version}` . 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 have to 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, theses 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 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:/catalogue/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 also be **multi-threaded** with the **parallelIngests** tag.

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 or 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>
```

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 -->
      <!-- instrument, productType, sensing date (year, month, day) -->
      <ds:prefixLocation>instrument/productType/year/month/day</ds:prefixLocation>
</ds:dataStore>
```

8.3.3.3 SwiftDataStoreGroup

You can organise datastores in group, the datastore group.

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's name. 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 -->
    <!-- instrument, productType, sensing date (year, month, day) -->
    <ds:prefixLocation>instrument/productType/year/month/day</ds:prefixLocation>

</ds:dataStore>
```

8.3.3.4 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>
  <!-- [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] 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>
    <!-- [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>
  </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 SwiftDataStores.

- 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

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 swift container 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 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>
```

8.3.5 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.

- **generateTrace**: This task is used to generate a trace record (json) for each ingested product and save it to a directory.

```
<ingester:task xsi:type="ingester:generateTrace" pattern=".*" tryLimit="1"
    stopOnFailure="true" privateKeyPath="/path/to/secret-key.txt"
    passphrase="*****" destinationFolder="/path/to/traces"
    serviceContext="gs-cdh" serviceType="DISTRIBUTION" serviceProvider="cdh01" />
```

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 about filters / patterns
The ingester has : - A **filter for producer** to define the product which name matches the pattern it will check on the source. To process all the products, use ".*". (For all type of producers) - An **OData filter** only for OData source (on Producer) to apply on OData source (CSC or DHuS) - A **geographical filter** only for OData source (on Producer) to apply on OData source (CSC or DHuS) after OData filter - A **pattern** defined **for each task of the consumer**. The products matching the pattern are processed by the given task.

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. 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)
- docker-compose: 1.29.0 (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 in order 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

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 in order to launch the catalogue :

https://repository.gael-systems.com/repository/thirdparty/fr/gael/gss/cdh-catalogue/1.4.3/cdh-catalogue-1.4.3.zip

The zip archive is presented as follows :

```
cdh-catalogue/
├── etc/
│   ├── operational-samples/
│   │   ├── application.properties.sample
│   │   ├── gss_dest-folder.xml
│   │   └── gss_dest-swift.xml
│   │
│   ├── JSON-Samples/
│   │   └── Subscription
│   │       └── subscription.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.

The **application.properties** is used for the Keycloak server configuration and the embedded http server. The content of this file is described further in the sample file. 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:${project.version}`

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:${project.version}`

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 60 cdh-catalogue`

9.3 Catalogue Configuration

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 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"
}
```

9.3.1.2 Keycloak Web-browser Authentication Configuration

CDH-Catalogue uses Oauth2 client with Single Sign-On (SSO) as an authentication mechanism to allows 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	
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 this 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 §6.3.1

9.3.3.1 HFSDataStore

See 6.3.1.1

9.3.3.2 SwiftDataStore

See 6.3.1.2

9.3.3.3 SwiftDataStoreGroup

See 6.3.1.3

9.3.3.4 TimeBasedDataStoreGroup

See 6.3.1.4 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*)

9.3.4 DataStore Options

See §6.3.2 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 to 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.

DirectDownloadLink is used to enable/disable a possible download from a swift 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] 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="true" quotaDisabled="false" />
```

10. Catalogue End-points

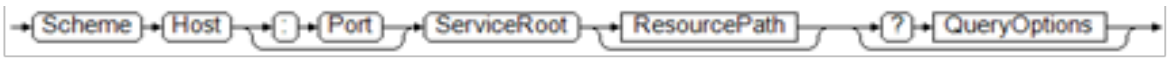
Here are some URLs for accessing each catalogue functionalities.12.8

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.

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 they exist and match 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 they exist.

10.3.7 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 they exist.

```
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))')

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.

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 d 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.

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)

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",
    "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-1.4.3.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
```

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 into the database.

11.1 Pre-requisites

11.1.1 Infrastructure Requirements

We recommend to install :

- docker: 20.10.12 (or later)
- docker-compose: 1.29.0 (or later)

11.1.2 Network Requirements

Application	Default port
PostgreSQL	5432

11.1.3 Software Requirements

Some tools must be installed in order 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/1.4.3/cdh-notification-1.4.3.zip The archive must be downloaded and unzipped in order 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:${project.version}`

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:${project.version}`

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>

# MANDATORY - Semi-colon separated list of topics
kafka.topics = deletion

# 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>
# OPTIONAL - Prefix for kafka notifications topics names
#kafka.topics.prefix = GSSN01-
```

RECOMMENDATION
You can find operational configuration sample files in the cdh-notification-1.4.3.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 end-points

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.

12.1 Quota schema

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-1.4.3.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-1.4.3.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 Datastores API

We can configure the 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	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
credentialsName	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

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

12.4.1 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)

12.4.2 HFS Datastore API

12.4.2.1 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-1.4.3.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.

12.4.2.2 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

12.4.2.3 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
```

12.4.2.4 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}

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 logs.

12.4.2.5 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

12.4.3 Swift Datastore API

12.4.3.1 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-1.4.3.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.

12.4.3.2 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

12.4.3.3 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": "intrument"
}
```

Response :

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

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

12.4.3.4 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>

Body :

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

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


Response :


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


Error :

HTTP 404 - If datastore does not exist.

HTTP 500 - Internal error. See the logs.

12.4.3.5 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
```

12.4.4 SwiftGroup Datastore API

12.4.4.1 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)

12.4.4.2 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.

12.4.4.3 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": "0"
            },
            {
                "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": "0"
            },
            {
                "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.

12.4.4.4 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>

Body :

```
{
    "permission": [
        "READ",
        "WRITE",
        "DELETE"
    ],
    "properties": {
        "property": [
            {
                "name": "STORE_BY_NAME",
                "value": "true"
            },
            {
                "name": "SAVE_AS_MULTIPART",
                "value": "0"
            },
            {
                "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": "0"
            },
            {
                "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.

12.4.4.5 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 exist.

HTTP 500 - Internal error. See the logs.

12.4.5 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-1.4.3.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.

12.4.5.1 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.

12.4.5.2 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.

12.4.5.3 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": "timeBasedDataStoreGroupConf",
    "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": "timeBasedDataStoreGroupConf",
    "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.

12.4.5.4 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>

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": "0"
                    },
                    {
                        "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": "0"
                    },
                    {
                        "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": "0"
                    },
                    {
                        "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": "0"
                    },
                    {
                        "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.

12.4.5.5 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.

12.5 Metadatastores API

12.5.1 Solr Metadatastore API

The field strategy could have the values:

- REQUEST_BY_ID
- REQUEST_BY_FILTER
- COUNT_PRODUCTS

RECOMMENDATION
You can find JSON configuration sample files in the cdh-ingest-1.4.3.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.

12.5.1.1 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
```

12.5.1.2 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
```

12.5.1.3 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
```

12.5.1.4 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>

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.

12.5.1.5 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
```

12.6 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

12.6.1 Reprocess

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

12.6.2 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	
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

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

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

Note:

- The assumed format property is mandatory to ingest with the CDSE type.
- The assumedFormat property is mandatory to ingest with both dhus and cdse type.

12.6.3 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`

12.6.4 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`

12.6.5 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-1.4.3.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",
    "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",
    "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",
    "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)

Folder source :

```
{
    "name": "producer1",
    "hosts": "host1,host2",
    "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",
    "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",
    "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",
    "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 – basic auth :

```
curl -v -X POST 'https://<server.url>:<port>/producers' -d '{
    "name": "producerOdata",
    "hosts": "host1, host2",
    "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"
        }
```

```
    }
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" |
jq
```

cURL – oauth2 :

```
curl -v -X POST 'https://<server.url>:<port>/producers' -d '{
    "name": "producerOdataOAuth2",
    "hosts": "host1, host2",
    "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"
        }
    }
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" |
jq
```

12.6.6 Update a Producer (PATCH)

Description : Update a producer by name.

NOTE: 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>

Body : Folder source :

```
{
    "name": "producer1",
    "hosts": "host1,host2",
    "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": "hhhh",
    "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",
    "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",
    "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.

12.6.7 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`

12.7 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

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
fr.gael.gss.ingest.ingester.GenerateTrace	Generation of trace task

NOTE : An error Manager can be either a Swift container or a folder. A credential must exist in case of Swift.

Swift container :

```
"errorManager": {
```

```
    "container": "containerError",
    "type": "swift",
    "credentials": "credential1"
}
```

Folder :

```
"errorManager": {
    "errorLocation": "/path/to/store/products/in/error", // internal path in docker compose file
    "type": "folder"
}
```

JSON Properties for consumers

Property	Type	Mandatory	Default value
hosts	String	true	
topics	String	true	
pollIntervalMs	int		1200000
filter	String		
parallelIngests	int		4
groupId	String	true	
topicPattern	String		
reprocess	boolean		false
sourceDelete	boolean		false
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		

Generate trace task properties

Property	Type	Mandatory	Default value
pattern	String		
stopOnFailure	boolean		
tryLimit	int		
active	boolean		
privateKeyPath	String	true	
passphrase	String	true	
serviceContext	String	true	
serviceType	String	true	
serviceProvider	String	true	
traceDestination.folder	String		
traceDestination.type	String	true	Value: folder or server
traceDestination.password	String		
traceDestination.serverUrl	String		
traceDestination.clientId	String		
traceDestination.tokenEndpoint	String		

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	

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

WARNING: In the current implementation, all the default values will be applied if the property is not present. Moreover, during the metadata synchronization from the CDSE, the credentials in the consumer shall be set in order to retrieve the product manifest.

12.7.1 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, ODATA
- **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`

12.7.2 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`

12.7.3 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-1.4.3.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 and GSS.

Body : Folder source

```
{
    "name": "consumerFolder1",
    "parallelIngests": 4,
    "hosts": "hots1,host2",
    "groupId": "group-consumer",
    "topics": "topic-ingestion-S2,topic-ingestion-S3",
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "errorManager": {
        "errorLocation": "/path/to/error",
        "type": "folder"
    },
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService",
            "traceDestination": {
                "type": "folder",
                "folder": "/path/to/store/traces/"
            }
        }
    ],
    "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",
    "groupId": "my_group_consumer",
    "topics": "topic1,topic2",
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
```

```
    "ingestThreads": 3,
    "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"
        },
```

```
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService",
            "traceDestination": {
                "type": "server",
                "user": "myuser",
                "serverUrl": "http://trace.server.url",
                "clientId": "myClient",
                "tokenEndpoint": "https://token.end.point"
            }
        }
    ],
    "tmpPath": "/path/to/tmp/for/tasks"
}
```

Response : The created consumer (HTTP 200)

Body : OData source with basic authentication

```
{
    "name": "consumerOdata",
    "parallelIngests": 4,
    "hosts": "host1,host2",
    "groupId": "my_consumer_group",
    "topics": "topic1",
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "errorManager": {
        "errorLocation": "/path/to/store/products/in/error", // internal path in docker compose file
    "type": "folder"
    },
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
```

```
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService",
            "traceDestination": {
                "type": "folder",
                "folder": "/path/to/store/traces/"
            }
        }
    ],
    "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",
    "groupId": "my_consumer_group",
    "topics": "topic1,topic3",
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
```

```
            "serviceType": "typeService",
            "serviceProvider": "provideOfService",
            "traceDestination": {
                "type": "server",
                "user": "myuser",
                "serverUrl": "http://trace.server.url",
                "clientId": "myClient",
                "tokenEndpoint": "https://token.end.point"
            }
        }
    ],
    "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,
    "hosts": "hots1,host2",
    "groupId": "group-consumer",
    "topics": "topic-ingestion-S2,topic-ingestion-S3",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "errorManager": {
        "errorLocation": "/path/to/error",  // internal path in docker compose file
    "type": "folder"
    },
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService",
            "traceDestination": {
                "type": "folder",
                "folder": "/path/to/store/traces/" // internal path in docker compose file
            }
        }
    ],
    "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",
    "groupId": "my_group_consumer",
    "topics": "topic1,topic2",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "traceDestination": {
                "folder": null,
                "user": "myuser",
                "password": null,
                "serverUrl": "http://trace.server.url",
                "clientId": "client_id",
                "tokenEndpoint": "https://token.end.point",
                "type": "server"
            },
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService"
        }
    ],
    "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)

OData source with basic authentication :

```
{
    "name": "consumerOdata",
    "parallelIngests": 4,
    "hosts": "host1,host2",
    "groupId": "my_consumer_group",
    "topics": "topic1",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "traceDestination": {
                "folder": "/path/to/store/traces/",
                "user": null,
                "password": null,
                "serverUrl": null,
                "clientId": "client_id",
                "tokenEndpoint": null,
                "type": "folder"
            },
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService"
        }
    ],
    "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)

OData source with OAuth2 authentication :

```
{
    "name": "consumerOdataOAuth2",
    "parallelIngests": 4,
    "hosts": "host1,host2",
    "groupId": "my_consumer_group",
    "topics": "topic1,topic3",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "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": "cdse",
        "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "traceDestination": {
                "folder": null,
                "user": "myuser",
                "password": null,
                "serverUrl": "http://trace.server.url",
                "clientId": "client_id",
                "tokenEndpoint": "https://token.end.point",
                "type": "server"
            },
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService"
        }
```

```
    ],
    "tmpPath": "/path/to/tmp/for/tasks",  // internal path in docker compose file
  "errorManager": null
}
```

Response : A Created consumer (HTTP 200)

Error :

HTTP 400 - If a field is missing or the object is not correct.

HTTP 409 – Consumer with the same name already exists.

HTTP 500 - Internal error. See the logs.

cURL basic auth

```
curl -v -X POST 'https://<server.url>:<port>/consumers' -d '{
  "name": "consumerOdata21",
  "parallelIngests": 4,
  "hosts": "host1,host2",
  "groupId": "my_consumer_group",
  "topics": "topic1",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": false,
  "ingestThreads": 3,
  "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"
    },
```

```
    {
      "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
      "pattern": "string",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
      "traceDestination": {
        "folder": "/path/to/store/traces/",
        "user": null,
        "password": null,
        "serverUrl": null,
        "clientId": "client_id",
        "tokenEndpoint": null,
        "type": "folder"
      },
      "privateKeyPath": "/path/to/keypath",
      "passphrase": "passphrase",
      "serviceContext": "serviceContext-context",
      "serviceType": "typeService",
      "serviceProvider": "provideOfService"
    }
  ],
  "tmpPath": "/path/to/tmp/for/tasks",
  "errorManager": {
    "errorLocation": "/path/to/store/products/in/error",
    "container": null,
    "type": "folder",
    "credentials": null
  }
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" | jq
```

cURL oauth2

```
curl -v -X POST 'https://<server.url>:<port>/consumers' -d '{
  "name": "consumerOdataOAuth2",
  "parallelIngests": 4,
  "hosts": "host1,host2",
  "groupId": "my_consumer_group",
  "topics": "topic1,topic3",
  "topicPattern": null,
  "reprocess": false,
  "pollIntervalMs": 1200000,
  "sourceDelete": false,
  "ingestThreads": 3,
  "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": "cdse",
    "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"
    },
    {
      "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
      "pattern": "string",
      "stopOnFailure": false,
      "tryLimit": 0,
      "active": true,
      "traceDestination": {
        "folder": null,
        "user": "myuser",
        "password": null,
        "serverUrl": "http://trace.server.url",
        "clientId": "client_id",
```

```
        "tokenEndpoint": "https://token.end.point",
        "type": "server"
      },
      "privateKeyPath": "/path/to/keypath",
      "passphrase": "passphrase",
      "serviceContext": "serviceContext-context",
      "serviceType": "typeService",
      "serviceProvider": "provideOfService"
    }
  ],
  "tmpPath": "/path/to/tmp/for/tasks",
  "errorManager": null
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" |
jq
```

12.7.4 Update a Consumer (PATCH)

Description : Update a consumer by name.

HTTP Method : PATCH

URL : http://<server.url>:<server.port>/<context.path>/consumers/<consumer.name>

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",
    "groupId": "group-consumer",
    "topics": "topic-ingestion-S2,topic-ingestion-S3",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "errorManager": {
        "errorLocation": "/path/to/error",  // internal path in docker compose file
    "type": "folder"
    },
    "ingestThreads": 3,
```

```
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
```

```
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService",
            "traceDestination": {
                "type": "folder",
                "folder": "/path/to/store/traces/" // internal path in docker compose file
            }
        }
    ],
    "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",
    "groupId": "my_group_consumer",
    "topics": "topic1,topic2",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "traceDestination": {
                "folder": null,
                "user": "myuser",
                "password": null,
                "serverUrl": "http://trace.server.url",
                "clientId": "client_id",
                "tokenEndpoint": "https://token.end.point",
                "type": "server"
            },
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService"
        }
```

```
    ],
    "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",
    "groupId": "my_consumer_group",
    "topics": "topic1",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "traceDestination": {
                "folder": "/path/to/store/traces/",
                "user": null,
                "password": null,
                "serverUrl": null,
                "clientId": "client_id",
                "tokenEndpoint": null,
                "type": "folder"
            },
            "privateKeyPath": "/path/to/keypath",
```

```
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService"
        }
    ],
    "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",
    "groupId": "my_consumer_group",
    "topics": "topic1,topic3",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 1200000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "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"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.GenerateTrace",
            "pattern": "string",
            "stopOnFailure": false,
            "tryLimit": 0,
            "active": true,
            "traceDestination": {
                "folder": null,
                "user": "myuser",
                "password": null,
                "serverUrl": "http://trace.server.url",
```

```
                "clientId": "client_id",
                "tokenEndpoint": "https://token.end.point",
                "type": "server"
            },
            "privateKeyPath": "/path/to/keypath",
            "passphrase": "passphrase",
            "serviceContext": "serviceContext-context",
            "serviceType": "typeService",
            "serviceProvider": "provideOfService"
        }
    ],
    "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.

12.7.5 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
```

12.8 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-1.4.3.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.

12.8.1 Schema

Property	Type	Mandatory	Description / Default value
jobId	uuid	yes	autogenerated
message	String		A comment on deletion
odataFilter	String		Filter for products in stores
reason	**Enum** CORRUPTED_PRODUCT WRONG_CHECKSUM DUPLICATED_PRODUCT OBSOLETE_PRODUCT	yes	Define the deletion reason
status	**Enum** PAUSED CANCELED CREATED	yes	Default status : CREATED
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 launch for the job

12.8.2 Create a deletion job (POST)

Description : Create a deletion job.

HTTP Method : POST

URL : http://<server.url>:<server.port>/jobs

Body :

```
{
    "message": "My testing",
    "reason": "CORRUPTED_PRODUCT",
    "status": "RUNNING",
    "odataFilter": "$filter=contains(Name,'R142')",
    "nbThreads": 2
}
```


Response:


```
{
    "message": "My testing",
    "reason": "CORRUPTED_PRODUCT",
    "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "errors": 0,
    "nbProducts": 3,
    "creationDate": "2023-11-12T19:00:48.235Z",
    "updatedDate": null,
    "status": "RUNNING",
    "odataFilter": "$filter=contains(Name,'R142')",
    "nbThreads": 2
}
```


cURL

```
curl -v -X POST ''https://<server.url>:<port>/jobs' -d '{
    "message": "Command line test",
    "reason": "CORRUPTED_PRODUCT",
    "status": "RUNNING",
    "odataFilter": "$filter=startswith(Name,'\''S3'\'')",
    "nbThreads": 2
}' -H 'Content-Type: application/json' -H "Authorization: Bearer ${AT}" |
jq
```

12.8.3 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:

```
{
    "message": "My testing",
    "reason": "CORRUPTED_PRODUCT",
    "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
```

12.8.4 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 :


```
[
    {
        "message": "job 1",
        "reason": "CORRUPTED_PRODUCT",
        "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
        "errors": 0,
        "nbProducts": 3,
        "creationDate": "2023-11-12T19:00:48.235Z",
        "updatedDate": null,
        "status": "RUNNING",
        "odataFilter": "$filter=contains(Name,'R142')",
        "nbThreads": 2
    },
    {
        "message": "job 2",
        "reason": "CORRUPTED_PRODUCT",
        "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
```

```
        "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`

12.8.5 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 :


```
{
    "message": "My testing",
    "reason": "CORRUPTED_PRODUCT",
    "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "errors": 0,
    "nbProducts": 0,
    "creationDate": "2023-11-12T19:00:48.235Z",
    "updatedDate": null,
    "status": "RUNNING",
    "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`

12.8.6 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`

12.8.7 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 :

```
{
    "message": "My testing",
    "reason": "CORRUPTED_PRODUCT",
    "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "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
```

12.8.8 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 :

```
{
    "message": "My testing",
    "reason": "CORRUPTED_PRODUCT",
    "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "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
```

12.8.9 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 :


```
{
    "message": "My testing",
    "reason": "CORRUPTED_PRODUCT",
    "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "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
```

12.8.10 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 :


```
{
    "message": "My testing",
    "reason": "CORRUPTED_PRODUCT",
    "jobId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
```

```
    "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
```

12.8.11 ANNEXES – Structure

```
.
├── Annex 1.    Examples of XML Configuration Files
│   ├── Annex 1.1.  XML - Examples of CDH-Ingest - Component
│   │   ├── Annex 1.1.1.    Examples of Docker Compose for CDH-Ingest - HFS Datastores -
DHuS source
│   │   │   ├── Annex 1.1.1.1.  Example of Docker Compose - INGESTION-DHUS-HFS-S1
│   │   │   ├── Annex 1.1.1.2.  Example of Docker Compose - INGESTION-DHUS-HFS-S2
│   │   │   ├── Annex 1.1.1.3.  Example of Docker Compose - INGESTION-DHUS-HFS-S3
│   │   │   ├── Annex 1.1.1.4.  Example of Docker Compose - INGESTION-DHUS-HFS-S5
│   │   │
│   │   ├── Annex 1.1.2.    Examples of Docker Compose for CDH-Ingest - SWIFT Datastores -
DHuS source
│   │   │
│   │   │   ├── Annex 1.1.2.1.  Example of Docker Compose - INGESTION-DHUS-SWIFT-S1
│   │   │   ├── Annex 1.1.2.2.  Example of Docker Compose - INGESTION-DHUS-SWIFT-S2
│   │   │   ├── Annex 1.1.2.3.  Example of Docker Compose - INGESTION-DHUS-SWIFT-S3
│   │   │   ├── Annex 1.1.2.4.  Example of Docker Compose - INGESTION-DHuS-SWIFT-S5
│   │   │
│   │   ├── Annex 1.1.3.    Example of application.properties - HFS Datastores - DHuS source
│   │   │   ├── Annex 1.1.3.1.  Example of application.properties - INGESTION_DHUS-HFS_S1
│   │   │   ├── Annex 1.1.3.2.  Example of application.properties - INGESTION_DHUS-HFS_S2
│   │   │   ├── Annex 1.1.3.3.  Example of application.properties - INGESTION_DHUS-HFS_S3
│   │   │   ├── Annex 1.1.3.4.  Example of application.properties - INGESTION_DHUS-HFS_S5
│   │   │
│   │   ├── Annex 1.1.4.    Examples of application.properties - SWIFT Datastores - DHuS
source
│   │   │   ├── Annex 1.1.4.1.  Example of application.properties - INGESTION_DHUS-SWIFT_S1
│   │   │   ├── Annex 1.1.4.2.  Example of application.properties - INGESTION_DHUS-SWIFT_S2
│   │   │   ├── Annex 1.1.4.3.  Example of application.properties - INGESTION_DHUS-SWIFT_S3
│   │   │   ├── Annex 1.1.4.4.  Example of application.properties - INGESTION_DHUS-SWIFT_S5
│   │   │
│   │   ├── Annex 1.1.5.    XML - Examples of CDH-Ingest - Producers
│   │   │   ├── Annex 1.1.5.1.  Example of gss.xml producer - Ingest from a Folder source
│   │   │   ├── Annex 1.1.5.2.  Example of gss.xml producer - Ingest from a DHuS source
│   │   │   ├── Annex 1.1.5.3.  Example of gss.xml producer - Ingest from a CSC (GSS) source
│   │   │   ├── Annex 1.1.5.4.  Example of gss.xml producer - Ingest from an Object-storage
source
│   │   │
│   │   └── Annex 1.1.6.    XML - Examples of CDH-Ingest - Consumers
│   │       ├── Annex 1.1.6.1.  Example of gss.xml consumer - Retrieve from a Folder
│   │       ├── Annex 1.1.6.2.  Example of gss.xml consumer - Retrieve from a DHuS
│   │       ├── Annex 1.1.6.3.  Example of gss.xml consumer - Retrieve from a CSC (GSS)
│   │       ├── Annex 1.1.6.4.  Example of gss.xml consumer - Retrieve from an Object-
Storage (Swift)
│   │
```

Annex 1. Examples of XML Configuration Files

Annex 1 contains all the configuration files and XML operational samples concerning the CDH-Ingestion via XML.

Some sample of configuration files are suggested below. Mandatory parameters will be commented with [M] and optional ones with a [O].

Annex 1.1. XML - Examples of CDH-Ingest – Component

Annex 1.1.1. Examples of Docker Compose for CDH-Ingest – HFS Datastores – DHuS source

Annex 1.1.1.1. Example of Docker Compose – INGESTION-DHUS-HFS-S1

```
version: "3"
services:
  INGESTION-DHUS-HFS-S1:
    image: "gaeldockerhub/cdh-ingest:${TAG}"

    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties

    volumes:
       - ./INGESTION_DHUS-HFS_S1.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
       - <path-to-destination-folder>/error:/ingest/error
       - <path-to-destination-folder>/tmp:/ingest/tmp
       - <path-to-destination-folder>/<dest-folder-S1-name>:/ingest/folder-S1
```

Annex 1.1.1.2. Example of Docker Compose – INGESTION-DHUS-HFS-S2

```
version: "3"
services:
  INGESTION-DHUS-HFS-S2:
    image: "gaeldockerhub/cdh-ingest:${TAG}"

    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties

    volumes:
       - ./INGESTION_DHUS-HFS_S2.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
       - <path-to-destination-folder>/error:/ingest/error
       - <path-to-destination-folder>/tmp:/ingest/tmp
       - <path-to-destination-folder>/<dest-folder-S2-name>:/ingest/folder-S2
```

Annex 1.1.1.3. Example of Docker Compose – INGESTION-DHUS-HFS-S3

```
version: "3"
services:
  INGESTION-DHUS-HFS-S3:
    image: "gaeldockerhub/cdh-ingest:${TAG}"

    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties

    volumes:
       - ./INGESTION_DHUS-HFS_S3.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
       - <path-to-destination-folder>/error:/ingest/error
       - <path-to-destination-folder>/tmp:/ingest/tmp
       - <path-to-destination-folder>/<dest-folder-S3-name>:/ingest/folder-S3
```

Annex 1.1.1.4. Example of Docker Compose – INGESTION-DHUS-HFS-S5

```
version: "3"
services:
  INGESTION-DHUS-HFS-S5:
    image: "gaeldockerhub/cdh-ingest:${TAG}"

    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties

    volumes:
       - ./INGESTION_DHUS-HFS_S5.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
       - <path-to-destination-folder>/error:/ingest/error
       - <path-to-destination-folder>/tmp:/ingest/tmp
       - <path-to-destination-folder>/<dest-folder-S5-name>:/ingest/folder-S5
```

Annex 1.1.2. Examples of Docker Compose for CDH-Ingest – SWIFT Datastores – DHuS source

Annex 1.1.2.1. Example of Docker Compose – INGESTION-DHUS-SWIFT-S1

```
version: "3"
services:
  INGESTION-DHUS-SWIFT-S1:
    image: "gaeldockerhub/cdh-ingest:${TAG}"

    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties

    volumes:
       - ./INGESTION_DHUS-SWIFT_S1.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
```

```
      - <path-to-destination-folder>/error:/ingest/error
      - <path-to-destination-folder>/tmp:/ingest/tmp
```

Annex 1.1.2.2. Example of Docker Compose – INGESTION-DHUS-SWIFT-S2

```
version: "3"
services:
  INGESTION-DHUS-SWIFT-S2:
    image: "gaeldockerhub/cdh-ingest:${TAG}"
    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties
    volumes:
       - ./INGESTION_DHUS-SWIFT_S2.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
       - <path-to-destination-folder>/error:/ingest/error
       - <path-to-destination-folder>/tmp:/ingest/tmp
```

Annex 1.1.2.3. Example of Docker Compose – INGESTION-DHUS-SWIFT-S3

```
version: "3"
services:
  INGESTION-DHUS-SWIFT-S3:
    image: "gaeldockerhub/cdh-ingest:${TAG}"
    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties
    volumes:
       - ./INGESTION_DHUS-SWIFT_S3.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
       - <path-to-destination-folder>/error:/ingest/error
       - <path-to-destination-folder>/tmp:/ingest/tmp
```

Annex 1.1.2.4. Example of Docker Compose – INGESTION-DHuS-SWIFT-S5

```
version: "3"
services:
  INGESTION-DHUS-SWIFT-S5:
    image: "gaeldockerhub/cdh-ingest:${TAG}"

    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties

    volumes:
       - ./INGESTION_DHUS-SWIFT_S5.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
       - <path-to-destination-folder>/error:/ingest/error
       - <path-to-destination-folder>/tmp:/ingest/tmp
```

Annex 1.1.3. Example of application.properties – HFS Datastores – DHuS source

Annex 1.1.3.1. Example of application.properties – INGESTION_DHUS-HFS_S1

```
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=

# CDH Configuration
## Datastore's names
cdh.datastores=<datastoreHfs-S1-name1>;<datastoreHfs-S1-name2>;<datastoreHfs-S1-name3>
## Metadatastore's names
cdh.metadatastores=<metadastore-name>
## Producer's names
```

```
cdh.producers=<producerName-DHUS-HFS-S1>
## Consumer's names
cdh.consumers=<consumerName-DHUS-HFS-S1>

# Generic Swift Configuration
#swiftConfiguration.segmentSizeMB=5000
#swiftConfiguration.retries=5
#swiftConfiguration.retryDelayMs=50
#swiftConfiguration.maxConnections=20

# Secret for encrypt
secret.key.password=<encryption>
```

Annex 1.1.3.2. Example of application.properties – INGESTION_DHUS-HFS_S2

```
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=

# CDH Configuration
## Datastore's names
cdh.datastores=<datastoreHfs-S2-name1>;<datastoreHfs-S2-name2>;<datastoreHfs-S2-name3>
## Metadatastore's names
cdh.metadatastores=<metadastore-name-S2>
## Producer's names
cdh.producers=<producerName-DHUS-HFS-S2>
## Consumer's names
cdh.consumers=<consumerName-DHUS-HFS-S2>

# Generic Swift Configuration
#swiftConfiguration.segmentSizeMB=5000
#swiftConfiguration.retries=5
#swiftConfiguration.retryDelayMs=50
#swiftConfiguration.maxConnections=20
```

```
# Secret for encrypt
secret.key.password=<encryption>
```

Annex 1.1.3.3. Example of application.properties – INGESTION_DHUS-HFS_S3

```
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=

# CDH Configuration
## Datastore's names
cdh.datastores=<datastoreHfs-S3-name1>;<datastoreHfs-S3-name2>;<datastoreHfs-S3-name3>
## Metadatastore's names
cdh.metadatastores=<metadastore-name-S3>
## Producer's names
cdh.producers=<producerName-DHUS-HFS-S3>
## Consumer's names
cdh.consumers=<consumerName-DHUS-HFS-S3>

# Generic Swift Configuration
#swiftConfiguration.segmentSizeMB=5000
#swiftConfiguration.retries=5
#swiftConfiguration.retryDelayMs=50
#swiftConfiguration.maxConnections=20

# Secret for encrypt
secret.key.password=<encryption>
```

Annex 1.1.3.4. Example of application.properties – INGESTION_DHUS-HFS_S5

```
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
```

```
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=

# CDH Configuration
## Datastore's names
cdh.datastores=<datastoreHfs-S5-name1>;<datastoreHfs-S5-name2>;<datastoreHfs-S5-name3>
## Metadatastore's names
cdh.metadatastores=<metadastore-name-S5>
## Producer's names
cdh.producers=<producerName-DHUS-HFS-S5>
## Consumer's names
cdh.consumers=<consumerName-DHUS-HFS-S5>

# Generic Swift Configuration
#swiftConfiguration.segmentSizeMB=5000
#swiftConfiguration.retries=5
#swiftConfiguration.retryDelayMs=50
#swiftConfiguration.maxConnections=20

# Secret for encrypt
secret.key.password=<encryption>
```

Annex 1.1.4. Examples of application.properties – SWIFT Datastores – DHuS source

Annex 1.1.4.1. Example of application.properties – INGESTION_DHUS-SWIFT_S1

```
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=
```

```
# CDH Configuration
## Datastore's names
cdh.datastores=<datastoreSwift-S1-name1>;<datastoreSwift-S1-name2>;<datastoreSwift-S1-
name3>
## Metadatastore's names
cdh.metadatastores=<metadastore-name-S1>
## Producer's names
cdh.producers=<producerName-DHUS-SWIFT-S1>
## Consumer's names
cdh.consumers=<consumerName-DHUS-SWIFT-S1>

# Generic Swift Configuration
swiftConfiguration.segmentSizeMB=5000
swiftConfiguration.retries=5
swiftConfiguration.retryDelayMs=50
swiftConfiguration.maxConnections=20

# Secret for encrypt
secret.key.password=<encryption>
```

Annex 1.1.4.2. Example of application.properties – INGESTION_DHUS-SWIFT_S2

```
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=

# CDH Configuration
## Datastore's names
cdh.datastores=<datastoreSwift-S2-name1>;<datastoreSwift-S2-name2>;<datastoreSwift-S2-
name3>
## Metadatastore's names
cdh.metadatastores=<metadastore-name-S2>
## Producer's names
cdh.producers=<producerName-DHUS-SWIFT-S2>
## Consumer's names
```

```
cdh.consumers=<consumerName-DHUS-SWIFT-S2>

# Generic Swift Configuration
swiftConfiguration.segmentSizeMB=5000
swiftConfiguration.retries=5
swiftConfiguration.retryDelayMs=50
swiftConfiguration.maxConnections=20

# Secret for encrypt
secret.key.password=<encryption>
```

Annex 1.1.4.3. Example of application.properties – INGESTION_DHUS-SWIFT_S3

```
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=

# CDH Configuration
## Datastore's names
cdh.datastores=<datastoreSwift-S3-name1>;<datastoreSwift-S3-name2>;<datastoreSwift-S3-
name3>
## Metadatastore's names
cdh.metadatastores=<metadastore-name-S3>
## Producer's names
cdh.producers=<producerName-DHUS-SWIFT-S3>
## Consumer's names
cdh.consumers=<consumerName-DHUS-SWIFT-S3>

# Generic Swift Configuration
swiftConfiguration.segmentSizeMB=5000
swiftConfiguration.retries=5
swiftConfiguration.retryDelayMs=50
swiftConfiguration.maxConnections=20

# Secret for encrypt
```

```
secret.key.password=<encryption>
```

Annex 1.1.4.4. Example of application.properties – INGESTION_DHUS-SWIFT_S5

```
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=

# CDH Configuration
## Datastore's names
cdh.datastores=<datastoreSwift-S5-name1>;<datastoreSwift-S5-name2>;<datastoreSwift-S5-
name3>
## Metadatastore's names
cdh.metadatastores=<metadastore-name-S5>
## Producer's names
cdh.producers=<producerName-DHUS-SWIFT-S5>
## Consumer's names
cdh.consumers=<consumerName-DHUS-SWIFT-S5>

# Generic Swift Configuration
swiftConfiguration.segmentSizeMB=5000
swiftConfiguration.retries=5
swiftConfiguration.retryDelayMs=50
swiftConfiguration.maxConnections=20

# Secret for encrypt
secret.key.password=<encryption>
```

Annex 1.1.5. XML - Examples of CDH-Ingest – Producers

Annex 1.1.5.1. Example of gss.xml producer – Ingest from a Folder source

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure an ingester in producer mode scanning a folder. -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:conf="fr:gael:gss:ingest:configuration"
   xmlns:ingester="fr:gael:gss:ingest:ingester" xmlns:ds="fr:gael:gss:core:datastore"
   xmlns:ms="fr:gael:gss:core:metadatastore">

   <!-- [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="test"
password="test"
      poolSize="2" />

   <!-- [M] Define the ingesters. In this scenario, there is only one producer that will
scan a DHuS -->
   <conf:ingesters>
      <ingester:ingester xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ingester:kafkaIngesterProducerConf" name="folder-producer">
         <!-- [M] comma separated kafka nodes (1 to n nodes) -->
         <ingester:hosts>localhost:9092</ingester:hosts>
         <!-- [M] name of the kafka topic where produced messages will be sent -->
         <ingester:topic>folder-ingestion</ingester:topic>
         <!-- [M] Frequency of the source scan in seconds. Default is 60. -->
         <ingester:pushInterval>10</ingester:pushInterval>
         <!-- [0] Only produce messages for products matching a regular expression -->
         <ingester:filter>.*</ingester:filter>

         <!-- [M] Source folder configuration -->
         <ingester:source xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="ingester:producerFolderConf">
            <!-- [M] Abolute path to the source folder -->
            <ingester:path>/path/to/source</ingester:path>
         </ingester:source>
```

```
      </ingester:ingester>
   </conf:ingesters>
</conf:configuration>
```

Annex 1.1.5.2. Example of gss.xml producer – Ingest from a DHuS source

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure an ingester in producer mode scanning products from a
DHuS OData v4 endpoint. -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:conf="fr:gael:gss:ingest:configuration"
   xmlns:ingester="fr:gael:gss:ingest:ingester" xmlns:ds="fr:gael:gss:core:datastore"
   xmlns:ms="fr:gael:gss:core:metadatastore">

   <!-- [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="test"
password="test"
      poolSize="2" />

   <!-- [M] Define the ingesters. In this scenario, there is only one producer that will
scan a DHuS -->
   <conf:ingesters>
      <ingester:ingester xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ingester:kafkaIngesterProducerConf" name="dhus-producer">
         <!-- [M] comma separated kafka nodes (1 to n nodes) -->
         <ingester:hosts>localhost:9092</ingester:hosts>
         <!-- [M] name of the kafka topic where produced messages will be sent -->
         <ingester:topic>dhus-ingestion</ingester:topic>
         <!-- [M] Frequency of the source scan in seconds. Default is 60. -->
         <ingester:pushInterval>10</ingester:pushInterval>
         <!-- [0] Only produce messages for products matching a regular expression -->
         <ingester:filter>.*</ingester:filter>

         <!-- [M] OData source configuration (elements can appear in any order) -->
         <ingester:source xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
            xsi:type="ingester:producerOdataConf">
            <!-- [M] Url of the OData endpoint -->

<ingester:serviceRootUrl>https://scihub.copernicus.eu/dhus/odata/v2</ingester:serviceRootU
rl>
            <!-- [0] Use basic auth to connect to the OData endpoint -->
            <ingester:auth type="basic">
               <!-- [M] User name -->
               <ingester:user>username</ingester:user>
               <!-- [M] User password -->
               <ingester:password>***</ingester:password>
            </ingester:auth>
            <!-- [0] How many products to take at each scan, default is 10 -->
            <ingester:top>100</ingester:top>
            <!-- [0] product published after this date will be selected -->
            <ingester:lastPublicationDate>2021-06-
01T00:00:00Z</ingester:lastPublicationDate>
            <!-- [0] OData filter to apply to select products -->
            <ingester:filter>Online eq true</ingester:filter>
            <!-- [M] Type of OData endpoint (can be dhus or csc) -->
            <ingester:type>dhus</ingester:type>
            <!-- [0] If you know the format of the source products, you can set it here.
It will avoid making requests to the DhuS -->
            <!-- instance to discover products format and make the producer run faster  --
>
            <ingester:assumedFormat>.zip</ingester:assumedFormat>
            <!-- [0] if true, use the lastPublicationDate saved in db if not null (search
by ingester name). Default is true -->
            <ingester:useDateFromDb>true</ingester:useDateFromDb>
            <!-- [0] if true, try to get the quicklook from the source. Default is false -
->
            <ingester:fetchQuicklook>false</ingester:fetchQuicklook>
            <!-- [0] Polygon in WKT format to get only products intersecting it-->
            <ingester:geoPostFilter>POLYGON((2.5902972717712602
48.83790703748822,2.59068082766039 48.83790703748822,2.59068082766039
48.83779052047628,2.5902972717712602 48.83779052047628,2.5902972717712602
48.83790703748822))</ingester:geoPostFilter>
         </ingester:source>
      </ingester:ingester>
   </conf:ingesters>
</conf:configuration>
```

Annex 1.1.5.3. Example of gss.xml producer – Ingest from a CSC (GSS) source

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure an ingester in producer mode scanning products from a
CSC OData v4 endpoint, -->
<!-- like DAS, PRIP or even a CDH instance. -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:conf="fr:gael:gss:ingest:configuration"
   xmlns:ingester="fr:gael:gss:ingest:ingester" xmlns:ds="fr:gael:gss:core:datastore"
   xmlns:ms="fr:gael:gss:core:metadatastore">

   <!-- [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="test"
password="test"
      poolSize="2" />

   <!-- [M] Define the ingesters. In this scenario, there is only one producer that will
scan a CSC source -->
   <conf:ingesters>
      <ingester:ingester xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ingester:kafkaIngesterProducerConf" name="csc-producer">
         <!-- [M] comma separated kafka nodes (1 to n nodes) -->
         <ingester:hosts>localhost:9092</ingester:hosts>
         <!-- [M] name of the kafka topic where produced messages will be sent -->
         <ingester:topic>csc-ingestion</ingester:topic>
         <!-- [M] Frequency of the source scan in seconds. Default is 60. -->
         <ingester:pushInterval>10</ingester:pushInterval>
         <!-- [0] Only produce messages for products matching a regular expression -->
         <ingester:filter>.*</ingester:filter>

         <!-- [M] OData source configuration (elements can appear in any order) -->
         <ingester:source xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="ingester:producerOdataConf">
            <!-- [M] Url of the OData endpoint -->
            <ingester:serviceRootUrl>https://localhost/odata/v1</ingester:serviceRootUrl>
            <!-- [0] Use OAuth2 to connect to the OData endpoint -->
            <ingester:auth type="oauth2">
```

```
                <!-- [M] User name -->
                <ingester:user>username</ingester:user>
                <!-- [M] User password -->
                <ingester:password>***</ingester:password>
                <!-- [M] Client ID -->
                <ingester:clientId>***</ingester:clientId>
                <!-- [M] Endpoint to get token -->
                <ingester:tokenEndpoint>https://address/auth/realms/realm/protocol/openid-
connect/token</ingester:tokenEndpoint>
            </ingester:auth>
            <!-- [0] How many products to take at each scan, default is 10 -->
            <ingester:top>100</ingester:top>
            <!-- [0] product published after this date will be selected -->
            <ingester:lastPublicationDate>2021-06-
01T00:00:00Z</ingester:lastPublicationDate>
            <!-- [0] OData filter to apply to select products -->
            <ingester:filter>Online eq true</ingester:filter>
            <!-- [M] Type of OData endpoint (can be dhus or csc) -->
            <ingester:type>csc</ingester:type>
            <!-- [0] if true, use the lastPublicationDate saved in db if not null (search
by ingester name). Default is true -->
            <ingester:useDateFromDb>true</ingester:useDateFromDb>
            <!-- [0] if true, get the metadata from the source. Default is false -->
            <ingester:fetchAttributes>false</ingester:fetchAttributes>
            <!-- [0] if true, try to get the quicklook from the source. Default is false -
->
            <ingester:fetchQuicklook>false</ingester:fetchQuicklook>
        </ingester:source>
      </ingester:ingester>
   </conf:ingesters>
</conf:configuration>
```

Annex 1.1.5.4. Example of gss.xml producer – Ingest from an Object-storage source

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure an ingester in producer mode scanning products from a
swift container. -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->
```

```
<conf:configuration xmlns:conf="fr:gael:gss:ingest:configuration"
xmlns:ingester="fr:gael:gss:ingest:ingester" xmlns:ds="fr:gael:gss:core:datastore"
xmlns:ms="fr:gael:gss:core:metadatastore">
<!-- [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="test"
password="test"
  poolSize="2" />

<!-- [M] Define the ingesters. In this scenario, there is only one producer that will scan
a Swift container  -->
<conf:ingesters>
  <ingester:ingester xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:type="ingester:kafkaIngesterProducerConf" name="obj-storage-producer">
     <!-- [M] comma separated kafka nodes (1 to n nodes) -->
     <ingester:hosts>localhost:9092</ingester:hosts>

     <!-- [M] name of the kafka topic where produced messages will be sent -->
     <ingester:topic>obj-storage-ingestion</ingester:topic>

     <!-- [M] Frequency of the source scan in seconds. Default is 60. -->
     <ingester:pushInterval>10</ingester:pushInterval>

     <!-- [0] Only produce messages for products matching a regular expression -->
     <ingester:filter>.*</ingester:filter>

     <!-- [M] Configuration of source containers -->
     <ingester:source xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="ingester:producerSwiftConf">
        <ingester:credentials name="credentialsSource">
           <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>
        </ingester:credentials>
        <ingester:containers>cdh-swift-ref</ingester:containers>
     </ingester:source>
  </ingester:ingester>
</conf:ingesters>
</conf:configuration>
```

Annex 1.1.6. XML - Examples of CDH-Ingest – Consumers

Annex 1.1.6.1. Example of gss.xml consumer – Retrieve from a Folder

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure an ingester in consumer mode for products stored in a
folder -->
<!-- Products will be ingested in a HFS DataStore and indexed in Solr -->
<!-- Mandatory parameters are commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:conf="fr:gael:gss:ingest:configuration"
xmlns:ingester="fr:gael:gss:ingest:ingester" xmlns:ds="fr:gael:gss:core:datastore"
xmlns:ms="fr:gael:gss:core:metadatastore">

<!-- [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="test"
password="test"
  poolSize="2" />

<!-- Stores are configured below. In this scenario, one HFS DataStore is 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>
    <!-- [M] Absolute path to the root path of the store -->
    <ds:path>/path/to/root</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://localhost:8983/solr, http://localhost: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>solr</ms:user>

    <!-- [0] Solr password -->
    <ms:password>SolrRocks</ms:password>

    <!-- [M] name of the solr collection -->
    <ms:collection>cdh</ms:collection>
  </ms:metadataStore>
</conf:metadataStores>

<!-- [M] Define the ingesters. In this scenario, the consumers will retrieve products from
a folder -->
```

```
<conf:ingesters>
  <ingester:ingester xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:type="ingester:kafkaIngesterConsumerConf" name="dhus-consumer">
     <!-- [0] number of parallel ingestions. Default is 4. Should match the parallel
download source quota -->
     <ingester:parallelIngests>4</ingester:parallelIngests>
     <!-- [M] comma separated kafka nodes (1 to n nodes) -->
     <ingester:hosts>localhost:9092</ingester:hosts>
     <!-- [M] if you want multiple consumers to read messages from the same topics, each
one of them must have the same
        groupId -->
     <ingester:groupId>folder-consumer-group</ingester:groupId>
     <!-- [M] comma separated kafka topics where consumed messages will be fetched -->
     <ingester:topics>folder-ingestion</ingester:topics>
     <!-- [0] whether or not to delete products from the source after a successful
ingestion -->
     <ingester:sourceDelete>true</ingester:sourceDelete>

     <!-- [M] Ingestion tasks. At least 1 task has to be configured -->
     <!-- The tasks can be declared in any order, they have an assigned not configurable
priority -->
     <!-- Each task as at least 3 parameters -->
     <!-- active: boolean, activate or not this task. Default is true -->
     <!-- pattern: filter which products will be processed. Default is ".*" i.e. all
products -->
     <!-- tryLimit: how many attempts will be done in case of error. Default is 1 -->
     <!-- stopOnFailure: if an error occurs during the task, indicates whether or not to
stop the ingestion -->
     <!-- and undo all previous successful tasks. Default is true -->

     <!-- tmpPath attribute : some tasks need to make a local copy of the product, it will
be done in this folder. Default
        is /tmp -->
     <ingester:tasks tmpPath="/tmp">
        <!-- [0] ingestInDataStores task: used to ingest the products in the datastores --
>
        <!-- [0] targetStores: comma separated list of stores where products will be
ingested -->
        <!-- if not specified, products will be ingested in all configured datastores -->
        <ingester:task xsi:type="ingester:ingestInDataStores" pattern=".*" tryLimit="1"
           stopOnFailure="true" targetStores="hfs" />
```

```
        <!-- [0] extractMetadata task: used to extract/retrieve metadata of a product -->
        <ingester:task xsi:type="ingester:extractMetadata" pattern=".*" tryLimit="1"
           stopOnFailure="true" />

        <!-- [0] ingestInMetadataStores task: used to ingest the products in the
MetadataStores -->
        <!-- [0] targetStores: comma separated list of stores where products will be
ingested -->
        <!-- if not specified, products will be ingested in all configured MetadataStores
-->
        <ingester:task xsi:type="ingester:ingestInMetadataStores" pattern=".*"
tryLimit="1"
           stopOnFailure="true" targetStores="solr" />

        <!-- [0] createQuicklook task: used to create and save quicklooks -->
        <!-- [M] targetStores: comma separated list of stores where quicklooks will be
saved -->
        <!-- Those store must have the property STORE_ATTACHED_FILES set to true -->
        <ingester:task xsi:type="ingester:createQuicklook" pattern=".*" tryLimit="1"
           stopOnFailure="false" targetStores="hfs" />

        <!-- [0] generateTrace task: generate a trace record (json) and save it in a
server -->
        <!-- [M] privateKeyPath: path to the file containing your private key -->
        <!-- [M] passphrase: your passphrase -->
        <!-- [M] serviceContext: name of your service on the traceability server -->
        <!-- [M] serviceType: type of service, should be DISTRIBUTION -->
        <!-- [M] serviceProvider: your identifer to indicate the source of the trace -->
        <!-- <ingester:task xsi:type="ingester:generateTrace" pattern=".*" tryLimit="1"
           stopOnFailure="true" privateKeyPath="/path/to/secret-key.txt"
passphrase="*****"
           serviceContext="gs-cdh" serviceType="DISTRIBUTION" serviceProvider="cdh01" > --
>
           <!-- [M] Where to send the trace, type can be : server, folder -->
           <!-- <ingester:traceDestination type="folder">
<ingester:folder>/path/to/folder</ingester:folder> </ingester:traceDestination> -->
           <!-- <ingester:traceDestination type="server">
              <ingester:user>***</ingester:user>
              <ingester:password>***</ingester:password>
              <ingester:serverUrl>https://demo.trace.gael-systems.com/trace-
api/Traces</ingester:serverUrl>
```

```
            <ingester:tokenEndpoint>https://demo.trace.gael-
systems.com/auth/realms/ts/protocol/openid-connect/token</ingester:tokenEndpoint>
            <ingester:clientId>trace-api</ingester:clientId>
          </ingester:traceDestination>
       </ingester:task>-->
    </ingester:tasks>

    <!-- [M] Source folder configuration -->
    <ingester:source xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:type="ingester:consumerFolderConf">
       <!-- [M] Abolute path to the source folder -->
       <ingester:path>/path/to/source</ingester:path>
    </ingester:source>
  </ingester:ingester>
</conf:ingesters>
</conf:configuration>
```

Annex 1.1.6.2. Example of gss.xml consumer – Retrieve from a DHuS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure an ingester in consumer mode for products stored in a
DHuS OData v4 endpoint -->
<!-- Products will be ingested in a swift containers and indexed in Solr -->
<!-- Mandatory parameters are commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:conf="fr:gael:gss:ingest:configuration"
   xmlns:ingester="fr:gael:gss:ingest:ingester" xmlns:ds="fr:gael:gss:core:datastore"
   xmlns:ms="fr:gael:gss:core:metadatastore">

   <!-- [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="test"
password="test"
      poolSize="2" />

   <!-- Stores are configured below. In this scenario, three swift stores are used. Two
are logical groups of the same containers
```

```
      : one that store products as packages (zip, tar.gz ...) and one that stores them as
multiparts. The third swift store is
      used to store quicklooks. Products will be stored by their names. 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">
         <!-- [M] Tenant name if using keystone V2 or Project name is using keystone V3 --
>
         <ds:tenant>***</ds:tenant>
         <!-- [M] User password -->
         <ds:password>***</ds:password>
         <!-- [M] User name -->
         <ds:user>***</ds:user>
         <!-- [M] Connection URL -->
         <ds:url>https://auth.cloud.ovh.net/v3</ds:url>
         <!-- [M] Region -->
         <ds:region>***</ds:region>
         <!-- [M] Domain -->
         <ds:domain>***</ds:domain>
      </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 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 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] Second swift group that store products in their original format
(directories) -->
      <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ds:swiftDataStoreGroupConf" name="swiftOriginal">
         <!-- [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>true</ds:value>
           </ds:property>
           <!-- [0] how many are used per product to upload parts. Default is 4 -->
           <ds:property>
              <ds:name>MULTIPART_THREADS</ds:name>
              <ds:value>4</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>quicklooks</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 -->
         <!-- 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://localhost:8983/solr, http://localhost: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>solr</ms:user>

        <!-- [0] Solr password -->
        <ms:password>SolrRocks</ms:password>

        <!-- [M] name of the solr collection -->
        <ms:collection>cdh</ms:collection>
      </ms:metadataStore>
   </conf:metadataStores>

   <!-- [M] Define the ingesters. In this scenario, the consumers will take products from
a DHuS OData v4 endpoint -->
   <conf:ingesters>
      <ingester:ingester xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="ingester:kafkaIngesterConsumerConf" name="dhus-consumer">
         <!-- [0] number of parallel ingestions. Default is 4. Should match the parallel
download source quota -->
         <ingester:parallelIngests>4</ingester:parallelIngests>
         <!-- [M] comma separated kafka nodes (1 to n nodes) -->
         <ingester:hosts>localhost:9092</ingester:hosts>
         <!-- [M] if you want multiple consumers to read messages from the same topics,
each one of them must have the same
            groupId -->
         <ingester:groupId>dhus-consumer-group</ingester:groupId>
         <!-- [M] comma separated kafka topics where consumed messages will be fetched -->
         <ingester:topics>dhus-ingestion</ingester:topics>

         <!-- [M] Ingestion tasks. At least 1 task has to be configured -->
         <!-- The tasks can be declared in any order, they have an assigned not
configurable priority -->
         <!-- Each task as at least 3 parameters -->
         <!-- active: boolean, activate or not this task. Default is true -->
```

```
        <!-- pattern: filter which products will be processed. Default is ".*" i.e. all
products -->
        <!-- tryLimit: how many attempts will be done in case of error. Default is 1 -->
        <!-- stopOnFailure: if an error occurs during the task, indicates whether or not
to stop the ingestion -->
        <!-- and undo all previous successful tasks. Default is true -->

        <!-- tmpPath attribute : some tasks need to make a local copy of the product, it
will be done in this folder. Default
           is /tmp -->
        <ingester:tasks tmpPath="/tmp">
           <!-- [0] ingestInDataStores task: used to ingest the products in the
datastores -->
           <!-- [0] targetStores: comma separated list of stores where products will be
ingested -->
           <!-- if not specified, products will be ingested in all configured datastores
-->
           <ingester:task xsi:type="ingester:ingestInDataStores" pattern=".*"
tryLimit="1"
              stopOnFailure="true" targetStores="swiftPackage, swiftOriginal" />

           <!-- [0] extractMetadata task: used to extract/retrieve metadata of a product
-->
           <ingester:task xsi:type="ingester:extractMetadata" pattern=".*" tryLimit="1"
              stopOnFailure="true" />

           <!-- [0] ingestInMetadataStores task: used to ingest the products in the
MetadataStores -->
           <!-- [0] targetStores: comma separated list of stores where products will be
ingested -->
           <!-- if not specified, products will be ingested in all configured
MetadataStores -->
           <ingester:task xsi:type="ingester:ingestInMetadataStores" pattern=".*"
tryLimit="1"
              stopOnFailure="true" targetStores="solr" />

           <!-- [0] createQuicklook task: used to create and save quicklooks -->
           <!-- [M] targetStores: comma separated list of stores where quicklooks will be
saved -->
           <!-- Those store must have the property STORE_ATTACHED_FILES set to true -->
           <ingester:task xsi:type="ingester:createQuicklook" pattern=".*" tryLimit="1"
              stopOnFailure="false" targetStores="swiftQL" />
```

```
            <!-- [0] generateTrace task: generate a trace record (json) and save it in a
server -->
            <!-- [M] privateKeyPath: path to the file containing your private key -->
            <!-- [M] passphrase: your passphrase -->
            <!-- [M] serviceContext: name of your service on the traceability server -->
            <!-- [M] serviceType: type of service, should be DISTRIBUTION -->
            <!-- [M] serviceProvider: your identifer to indicate the source of the trace -
->
            <!-- <ingester:task xsi:type="ingester:generateTrace" pattern=".*"
tryLimit="1"
               stopOnFailure="true" privateKeyPath="/path/to/secret-key.txt"
passphrase="*****"
               serviceContext="gs-cdh" serviceType="DISTRIBUTION" serviceProvider="cdh01"
> -->
               <!-- [M] Where to send the trace, type can be : server, folder -->
               <!-- <ingester:traceDestination type="folder">
<ingester:folder>/path/to/folder</ingester:folder> </ingester:traceDestination> -->
               <!-- <ingester:traceDestination type="server">
                  <ingester:user>***</ingester:user>
                  <ingester:password>***</ingester:password>
                  <ingester:serverUrl>https://demo.trace.gael-systems.com/trace-
api/Traces</ingester:serverUrl>
                  <ingester:tokenEndpoint>https://demo.trace.gael-
systems.com/auth/realms/ts/protocol/openid-connect/token</ingester:tokenEndpoint>
                  <ingester:clientId>trace-api</ingester:clientId>
                </ingester:traceDestination>
            </ingester:task>-->
         </ingester:tasks>

         <!-- [M] OData source configuration (elements can appear in any order) -->
         <ingester:source xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="ingester:consumerOdataConf">
            <!-- [M] Url of the OData endpoint -->

<ingester:serviceRootUrl>https://scihub.copernicus.eu/dhus/odata/v2</ingester:serviceRootU
rl>
            <!-- [0] Use basic auth to connect to the OData endpoint -->
            <ingester:auth type="basic">
               <!-- [M] User name -->
               <ingester:user>username</ingester:user>
               <!-- [M] User password -->
```

```
            <ingester:password>***</ingester:password>
         </ingester:auth>
         <!-- [M] Type of OData endpoint (can be dhus or csc) -->
         <ingester:type>dhus</ingester:type>
         <!-- [0] retriesOn429: if a HTTP 429 error is received, retry the download.
Default is 0 -->
         <ingester:retriesOn429>3</ingester:retriesOn429>
         <!-- [0] retryWaitOn429ms: time to wait in ms between retries. Default is 1000
-->
         <ingester:retryWaitOn429ms>5000</ingester:retryWaitOn429ms>
      </ingester:source>
   </ingester:ingester>
</conf:ingesters>
</conf:configuration>
```

Annex 1.1.6.3. Example of gss.xml consumer – Retrieve from a CSC (GSS)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- This file is used to configure an ingester in consumer mode for products stored in a
CSC OData v4 endpoint, -->
<!-- like DAS, PRIP or even a CDH instance. -->
<!-- Products will be ingested in a swift containers and indexed in Solr -->
<!-- Mandatory parameters are commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:conf="fr:gael:gss:ingest:configuration"
xmlns:ingester="fr:gael:gss:ingest:ingester" xmlns:ds="fr:gael:gss:core:datastore"
xmlns:ms="fr:gael:gss:core:metadatastore">

<!-- [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="test"
password="test"
  poolSize="2" />

<!-- Stores are configured below. In this scenario, three swift stores are used. Two are
logical groups of the same containers
```

```
  : one that store products as packages (zip, tar.gz ...) and one that stores them as
multiparts. The third swift store is
  used to store quicklooks. Products will be stored by their names. 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">
     <!-- [M] Tenant name if using keystone V2 or Project name is using keystone V3 -->
     <ds:tenant>***</ds:tenant>
     <!-- [M] User password -->
     <ds:password>***</ds:password>
     <!-- [M] User name -->
     <ds:user>***</ds:user>
     <!-- [M] Connection URL -->
     <ds:url>https://auth.cloud.ovh.net/v3</ds:url>
     <!-- [M] Region -->
     <ds:region>***</ds:region>
     <!-- [M] Domain -->
     <ds:domain>***</ds:domain>
  </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 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 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] Second swift group that store products in their original format (directories) -
->
  <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:type="ds:swiftDataStoreGroupConf" name="swiftOriginal">
     <!-- [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>true</ds:value>
        </ds:property>
        <!-- [0] how many are used per product to upload parts. Default is 4 -->
        <ds:property>
           <ds:name>MULTIPART_THREADS</ds:name>
           <ds:value>4</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>quicklooks</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 -->
     <!-- 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://localhost:8983/solr, http://localhost: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>solr</ms:user>

    <!-- [0] Solr password -->
    <ms:password>SolrRocks</ms:password>

    <!-- [M] name of the solr collection -->
    <ms:collection>cdh</ms:collection>
  </ms:metadataStore>
</conf:metadataStores>

<!-- [M] Define the ingesters. In this scenario, the consumers will take products from a
CSC OData v4 endpoint -->
<conf:ingesters>
  <ingester:ingester xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ingester:kafkaIngesterConsumerConf" name="csc-consumer">
    <!-- [0] number of parallel ingestions. Default is 4. Should match the parallel
download source quota -->
    <ingester:parallelIngests>4</ingester:parallelIngests>
    <!-- [M] comma separated kafka nodes (1 to n nodes) -->
    <ingester:hosts>localhost:9092</ingester:hosts>
    <!-- [M] if you want multiple consumers to read messages from the same topics, each
one of them must have the same
       groupId -->
    <ingester:groupId>csc-consumer-group</ingester:groupId>
    <!-- [M] comma separated kafka topics where consumed messages will be fetched -->
    <ingester:topics>csc-ingestion</ingester:topics>

    <!-- [M] Ingestion tasks. At least 1 task has to be configured -->
    <!-- The tasks can be declared in any order, they have an assigned not configurable
priority -->
    <!-- Each task as at least 4 parameters with default values -->
    <!-- active: boolean, activate or not this task. Default is true -->
```

```
     <!-- pattern: filter which products will be processed. Default is ".*" i.e. all
products -->
     <!-- tryLimit: how many attempts will be done in case of error. Default is 1 -->
     <!-- stopOnFailure: if an error occurs during the task, indicates whether or not to
stop the ingestion -->
     <!-- and undo all previous successful tasks. Default is true -->

     <!-- tmpPath attribute : some tasks need to make a local copy of the product, it will
be done in this folder. Default
        is /tmp -->
     <ingester:tasks tmpPath="/tmp">
        <!-- [0] ingestInDataStores task: used to ingest the products in the datastores --
>
        <!-- [0] targetStores: comma separated list of stores where products will be
ingested -->
        <!-- if not specified, products will be ingested in all configured datastores -->
        <ingester:task xsi:type="ingester:ingestInDataStores" pattern=".*" tryLimit="1"
           stopOnFailure="true" targetStores="swiftPackage, swiftOriginal" />

        <!-- [0] extractMetadata task: used to extract/retrieve metadata of a product -->
        <ingester:task xsi:type="ingester:extractMetadata" pattern=".*" tryLimit="1"
           stopOnFailure="true" />

        <!-- [0] ingestInMetadataStores task: used to ingest the products in the
MetadataStores -->
        <!-- [0] targetStores: comma separated list of stores where products will be
ingested -->
        <!-- if not specified, products will be ingested in all configured MetadataStores
-->
        <ingester:task xsi:type="ingester:ingestInMetadataStores" pattern=".*"
tryLimit="1"
           stopOnFailure="true" targetStores="solr" />

        <!-- [0] createQuicklook task: used to create and save quicklooks -->
        <!-- [M] targetStores: comma separated list of stores where quicklooks will be
saved -->
        <!-- Those store must have the property STORE_ATTACHED_FILES set to true -->
        <!-- [0] onlyUseProvidedQL: if true and producer has been configured to provide
the QL, -->
        <!-- only get the provided QL. If it does not exist, does not try to create a QL.
-->
        <ingester:task xsi:type="ingester:createQuicklook" pattern=".*" tryLimit="1"
```

```
          stopOnFailure="false" targetStores="swiftQL" onlyUseProvidedQL="false" />

       <!-- [0] generateTrace task: generate a trace record (json) and save it in a
server -->
       <!-- [M] privateKeyPath: path to the file containing your private key -->
       <!-- [M] passphrase: your passphrase -->
       <!-- [M] serviceContext: name of your service on the traceability server -->
       <!-- [M] serviceType: type of service, should be DISTRIBUTION -->
       <!-- [M] serviceProvider: your identifer to indicate the source of the trace -->
       <!-- <ingester:task xsi:type="ingester:generateTrace" pattern=".*" tryLimit="1"
          stopOnFailure="true" privateKeyPath="/path/to/secret-key.txt"
passphrase="*****"
          serviceContext="gs-cdh" serviceType="DISTRIBUTION" serviceProvider="cdh01" > --
>
          <!-- [M] Where to send the trace, type can be : server, folder -->
          <!-- <ingester:traceDestination type="folder">
<ingester:folder>/path/to/folder</ingester:folder> </ingester:traceDestination> -->
          <!-- <ingester:traceDestination type="server">
             <ingester:user>***</ingester:user>
             <ingester:password>***</ingester:password>
             <ingester:serverUrl>https://demo.trace.gael-systems.com/trace-
api/Traces</ingester:serverUrl>
             <ingester:tokenEndpoint>https://demo.trace.gael-
systems.com/auth/realms/ts/protocol/openid-connect/token</ingester:tokenEndpoint>
             <ingester:clientId>trace-api</ingester:clientId>
           </ingester:traceDestination>
       </ingester:task>-->
    </ingester:tasks>

    <!-- [M] OData source configuration (elements can appear in any order) -->
    <ingester:source xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:type="ingester:consumerOdataConf">
       <!-- [M] Url of the OData endpoint -->

<ingester:serviceRootUrl>https://catalogue.dataspace.copernicus.eu/odata/v1/</ingester:ser
viceRootUrl>
       <!-- [0] Use OAuth2 to connect to the OData endpoint -->
       <ingester:auth type="oauth2">
          <!-- [M] User name -->
          <ingester:user>***</ingester:user>
          <!-- [M] User password -->
          <ingester:password>***</ingester:password>
```

```
            <!-- [M] Client ID -->
            <ingester:clientId>cdse-public</ingester:clientId>
            <!-- [M] Endpoint to get token -->

<ingester:tokenEndpoint>https://identity.dataspace.copernicus.eu/auth/realms/CDSE/protocol
/openid-connect/token</ingester:tokenEndpoint>
        </ingester:auth>
        <!-- [M] Type of OData endpoint (can be dhus or csc) -->
        <ingester:type>csc</ingester:type>
        <!-- [0] If the source does not set file format in product name -->
        <ingester:assumedFormat>.zip</ingester:assumedFormat>
        <!-- [0] retriesOn429: if a HTTP 429 error is received, retry the download.
Default is 0 -->
        <ingester:retriesOn429>3</ingester:retriesOn429>
        <!-- [0] retryWaitOn429ms: time to wait in ms between retries. Default is 1000 -->
        <ingester:retryWaitOn429ms>5000</ingester:retryWaitOn429ms>
     </ingester:source>
  </ingester:ingester>
</conf:ingesters>
</conf:configuration>
```

Annex 1.1.6.4. Example of gss.xml consumer – Retrieve from an Object-Storage (Swift)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- This file is used to configure an ingester in consumer mode for products stored in a
Swift container -->

<!-- Products will be ingested in a swift containers and indexed in Solr -->
<!-- Mandatory parameters are commented with a [M], optional ones with a [0] -->

<conf:configuration xmlns:conf="fr:gael:gss:ingest:configuration"
xmlns:ingester="fr:gael:gss:ingest:ingester" xmlns:ds="fr:gael:gss:core:datastore"
xmlns:ms="fr:gael:gss:core:metadatastore">

<!-- [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="test"
password="test"
```

```
  poolSize="2" />

<!-- Stores are configured below. In this scenario, three swift stores are used. Two are
logical groups of the same containers
  : one that store products as packages (zip, tar.gz ...) and one that stores them as
multiparts. The third swift store is
  used to store quicklooks. Products will be stored by their names. 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">
     <!-- [M] Tenant name if using keystone V2 or Project name is using keystone V3 -->
     <ds:tenant>***</ds:tenant>
     <!-- [M] User password -->
     <ds:password>***</ds:password>
     <!-- [M] User name -->
     <ds:user>***</ds:user>
     <!-- [M] Connection URL -->
     <ds:url>https://auth.cloud.ovh.net/v3</ds:url>
     <!-- [M] Region -->
     <ds:region>***</ds:region>
     <!-- [M] Domain -->
     <ds:domain>***</ds:domain>
  </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 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 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>
           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] Second swift group that store products in their original format (directories) -
->
  <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:type="ds:swiftDataStoreGroupConf" name="swiftOriginal">
     <!-- [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>true</ds:value>
        </ds:property>

        <!-- [0] how many are used per product to upload parts. Default is 4 -->
        <ds:property>
           <ds:name>MULTIPART_THREADS</ds:name>
           <ds:value>4</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>
           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>quicklooks</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 -->
     <!-- 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://localhost:8983/solr, http://localhost: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>solr</ms:user>

     <!-- [0] Solr password -->
     <ms:password>SolrRocks</ms:password>
```

```
     <!-- [M] name of the solr collection -->
     <ms:collection>cdh</ms:collection>
  </ms:metadataStore>
</conf:metadataStores>

<!-- [M] Define the ingesters. In this scenario, the consumers will take products from a
swift container -->
<conf:ingesters>
  <ingester:ingester xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:type="ingester:kafkaIngesterConsumerConf" name="obj-storage-consumer">

     <!-- [0] number of parallel ingestions. Default is 4. Should match the parallel
download source quota -->
     <ingester:parallelIngests>4</ingester:parallelIngests>
     <!-- [M] comma separated kafka nodes (1 to n nodes) -->
     <ingester:hosts>localhost:9092</ingester:hosts>
     <!-- [M] if you want multiple consumers to read messages from the same topics, each
one of them must have the same
        groupId -->
     <ingester:groupId>obj-storage-consumer-group</ingester:groupId>

     <!-- [M] comma separated kafka topics where consumed messages will be fetched -->
     <ingester:topics>obj-storage-ingestion</ingester:topics>

     <!-- Indicates if ingested products have to be reprocessed -->
     <ingester:reprocess>false</ingester:reprocess>
     <!-- [0] whether or not to delete products from the source after a successful
ingestion -->
     <ingester:sourceDelete>false</ingester:sourceDelete>

     <!-- [M] Ingestion tasks. At least 1 task has to be configured -->
     <!-- The tasks can be declared in any order, they have an assigned not configurable
priority -->
     <!-- Each task as at least 3 parameters -->
     <!-- active: boolean, activate or not this task. Default is true -->
     <!-- pattern: filter which products will be processed. Default is ".*" i.e. all
products -->
     <!-- tryLimit: how many attempts will be done in case of error. Default is 1 -->
     <!-- stopOnFailure: if an error occurs during the task, indicates whether or not to
stop the ingestion -->
     <!-- and undo all previous successful tasks. Default is true -->
```

```
    <!-- tmpPath attribute : some tasks need to make a local copy of the product, it will
be done in this folder. Default
        is /tmp -->
    <ingester:tasks tmpPath="/tmp">
        <!-- [0] ingestInDataStores task: used to ingest the products in the datastores --
>
        <!-- [0] targetStores: comma separated list of stores where products will be
ingested -->
        <!-- if not specified, products will be ingested in all configured datastores -->
        <ingester:task xsi:type="ingester:ingestInDataStores" pattern=".*" tryLimit="1"
            stopOnFailure="true" targetStores="swiftPackage, swiftOriginal" />

        <!-- [0] extractMetadata task: used to extract/retrieve metadata of a product -->
        <ingester:task xsi:type="ingester:extractMetadata" pattern=".*" tryLimit="1"
            stopOnFailure="true" />

        <!-- [0] ingestInMetadataStores task: used to ingest the products in the
MetadataStores -->
        <!-- [0] targetStores: comma separated list of stores where products will be
ingested -->
        <!-- if not specified, products will be ingested in all configured MetadataStores
-->
        <ingester:task xsi:type="ingester:ingestInMetadataStores" pattern=".*"
tryLimit="1"
            stopOnFailure="true" targetStores="solr" />

        <!-- [0] createQuicklook task: used to create and save quicklooks -->
        <!-- [M] targetStores: comma separated list of stores where quicklooks will be
saved -->
        <!-- Those store must have the property STORE_ATTACHED_FILES set to true -->
        <ingester:task xsi:type="ingester:createQuicklook" pattern=".*" tryLimit="1"
            stopOnFailure="false" targetStores="swiftQL" />

        <!-- [0] generateTrace task: generate a trace record (json) and save it in a
server -->
        <!-- [M] privateKeyPath: path to the file containing your private key -->
        <!-- [M] passphrase: your passphrase -->
        <!-- [M] serviceContext: name of your service on the traceability server -->
        <!-- [M] serviceType: type of service, should be DISTRIBUTION -->
        <!-- [M] serviceProvider: your identifer to indicate the source of the trace -->
        <!-- <ingester:task xsi:type="ingester:generateTrace" pattern=".*" tryLimit="1"
```

```
          stopOnFailure="true" privateKeyPath="/path/to/secret-key.txt"
passphrase="*****"
          serviceContext="gs-cdh" serviceType="DISTRIBUTION" serviceProvider="cdh01" > --
>
          <!-- [M] Where to send the trace, type can be : server, folder -->
          <!-- <ingester:traceDestination type="folder">
<ingester:folder>/path/to/folder</ingester:folder> </ingester:traceDestination> -->
          <!-- <ingester:traceDestination type="server">
             <ingester:user>***</ingester:user>
             <ingester:password>***</ingester:password>
             <ingester:serverUrl>https://demo.trace.gael-systems.com/trace-
api/Traces</ingester:serverUrl>
             <ingester:tokenEndpoint>https://demo.trace.gael-
systems.com/auth/realms/ts/protocol/openid-connect/token</ingester:tokenEndpoint>
             <ingester:clientId>trace-api</ingester:clientId>
           </ingester:traceDestination>
        </ingester:task>-->
     </ingester:tasks>

     <!-- [M] Configuration of source containers -->
     <ingester:source xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="ingester:consumerSwiftConf">
        <ingester:credentials name="credentialsSource">
           <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>
        </ingester:credentials>
        <ingester:containers>cdh-swift-ref</ingester:containers>
     </ingester:source>
  </ingester:ingester>
</conf:ingesters>
</conf:configuration>
```

Annex 1.2. XML - Examples of CDH-Catalogue – Component

Annex 1.2.1. Example of application.properties Configuration for CDH-Catalogue

```
## OAuth-2.0 Authentication configuration

# To enable oauth2 authentication, set auth = true and uncomment all
spring.security.oauth2... properties
auth = false

#spring.security.oauth2.resourceserver.jwt.issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>
#spring.security.oauth2.resourceserver.jwt.jwk-issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>/protocol/openid-connect/certs

spring.security.oauth2.client.registration.keycloak.client-id=<client-id>
spring.security.oauth2.client.registration.keycloak.client-secret=<client-secret>
spring.security.oauth2.client.registration.keycloak.authorization-grant-
type=authorization_code
spring.security.oauth2.client.registration.keycloak.scope=openid,profile,roles
spring.security.oauth2.client.provider.keycloak.issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>
spring.security.oauth2.client.provider.keycloak.user-name-attribute=preferred_username

### IMPORTANT ### Please ensure that the 'keycloak.client', 'keycloak.role' and
'cors.origins' are not commented out !
# if auth=true, please provide the 'keycloak Client ID' and 'keycloak Client Role.'
keycloak.client = <client-id>
keycloak.role = <client-role>

## To enable Cross Origin Requests, for example to allow the Admin UI to call this API
cors.origins = http://<localhost>:<8081>

# 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

## Server configuration

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

```
server.port = <server.port>

# Configure context path
server.servlet.contextPath = <server.servlet.contextPath>

# The two below parameter are used to not have custom errors pages
server.error.whitelabel.enabled = false
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.web.servlet.error.Erro
rMvcAutoConfiguration

# If you want to use role configured in Keycloak client, set true
keycloak.use-resource-role-mappings = true

server.forward-headers-strategy = NATIVE

### Database configuration.
# WARNING: This part is mandatory even if you use XML configuration
spring.datasource.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
spring.datasource.username=<user.name>
spring.datasource.password=<user.password>
spring.datasource.driver-class-name=org.postgresql.Driver
# Max pool connexion size
spring.datasource.hikari.maximum-pool-size=100
#db.socketTimeout=1000
#db.networkTimeout=5000

deletion.kafka.hosts=<kafka.host>:<kafka.port>
deletion.kafka.topic=<kafka topic name>
deletion.datastores=datastore-name1;datastore-name2;datastore-name3
deletion.metadatastores=metadatastore-name1;metadatastore-name2

# To encrypt secret information in database (password for example)
secret.key.password=<encryption>
# End of mandatory part


# CDH Configuration
# Only used when stores are configured into DB
# Datastore's names
cdh.useDbConfiguration=true
cdh.datastores=datastore-name1;datastore-name2;datastore-name3
cdh.metadatastores=metadatastore-name1;metadatastore-name2
```

```
## Swift Configuration
#swiftConfiguration.segmentSizeMB=5000
#swiftConfiguration.retries=5
#swiftConfiguration.retryDelayMs=50
#swiftConfiguration.maxConnections=20

# ENABLE / DISABLE EVICTION BY TIME
process.evictionByTime=false

# directDownloadLink : enable the direct download from # cloud stores (swift) when using
$value. Default is true. --
# quotaDisabled : disable quotas. Default is false.
download.directDownloadLink=true
download.quotaDisabled=false
```

Annex 1.2.2. Example of gss.xml for CDH-Catalogue – HFS Datastore with Folder source

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- This file is used to configure DataStores, MetadataStores and other different system
parameters of the catalogue -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->
<conf:configuration xmlns:conf="fr:gael:gss:odata:configuration"
xmlns:ds="fr:gael:gss:core:datastore" xmlns:ms="fr:gael:gss:core:metadatastore"
xmlns:dl="fr:gael:gss:odata:download">
<!-- [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://IP_ADDRESS:PORT/database" login="username"
  password="password" poolSize="10" />
<!-- [0] Activate some background processes -->
<!-- evictionByTime activate the data transfer for TimeBasedDataStoreGroups. Default is
false -->
<conf:process evictionByTime="true" />
<!-- [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="false" />
```

```
<conf:dataStores>
  <ds:dataStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ds:hfsDataStoreConf"
     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>
     </ds:properties>
     <!-- [M] Abolute path folder where quicklooks are store. -->
     <!-- In case of using Docker, this path is referred to an internal conatiner path -->
     <!-- which will be mapped to an external absolute path folder of VM host into dokcer
run command -->
     <!-- as follows where quicklooks are stored: -v
/path/to/external/folder:/catalogue/folder -->
     <!-- In case of using .zip package, this path is referred to an absolute path folder
of VM host where -->
     <!-- quicklooks are stored. -->
     <ds:path>/catalogue/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>
  <!-- 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>EVICT_REFERENCE</ds:name>
           <ds:value>true</ds:value>
        </ds:property>
        <ds:property>
           <ds:name>EVICT_ATTACHED_FILES</ds:name>
           <ds:value>true</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="hfs">
           <ds:permission>READ</ds:permission>
           <ds:permission>WRITE</ds:permission>
           <ds:permission>DELETE</ds:permission>
           <!-- [M] Control the orders of stores in this group. Here each store has a
manually assigned priority -->
           <ds:properties>

              <!-- KEEP_PERIOD_SECONDS: How much time in seconds a product will be kept in
this store -->
              <!-- .................. : It should have the same value set into gss.xml of
Consumer Ingest -->
              <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] filter which products based on their name are accepted. ".*" means all
products -->
            <ds:filter>.*</ds:filter>
            <!-- [M] Abolute path folder where products are store. -->
            <!-- In case of using Docker, this path is referred to an internal conatiner
path -->
            <!-- which will be mapped to an external absolute path folder of VM host into
dokcer run command -->
            <!-- as follows where products are stored: -v
/path/to/external/folder:/catalogue/folder -->
            <!-- In case of using .zip package, this path is referred to an absolute path
folder of VM host where -->
            <!-- products are stored. -->
            <ds:path>/catalogue/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>
     </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://IP_ADDRESS:PORT/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>SolrCloud</ms:clientType>
     <!-- [0] Solr username -->
     <ms:user>username</ms:user>
     <!-- [0] Solr password -->
```

```
     <ms:password>password</ms:password>
     <!-- [M] name of the solr collection -->
     <ms:collection>gss</ms:collection>
     <!-- [M] defaultTop : how many results will be returned for every entities that can
be listed. Default 1000 -->
     <!-- [0] maxSkip : maximum value for skip. Default is 1000 -->
     <ms:defaultTop>100</ms:defaultTop>
     <ms:maxSkip>10000</ms:maxSkip>
  </ms:metadataStore>
</conf:metadataStores>
</conf:configuration>
```

Annex 1.2.3. Example of gss.xml for CDH-Catalogue – SWIFT Datastore with Swift source

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- This file is used to configure DataStores, MetadataStores and other different system
parameters of the catalogue -->
<!-- Mandatory parameters will be commented with a [M], optional ones with a [0] -->
<conf:configuration xmlns:conf="fr:gael:gss:odata:configuration"
   xmlns:ds="fr:gael:gss:core:datastore" xmlns:ms="fr:gael:gss:core:metadatastore"
xmlns:dl="fr:gael:gss:odata:download">
   <!-- [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://IP_ADDRESS:PORT/database" login="username"
      password="password" poolSize="10" />
   <!-- [0] Activate some background processes -->
   <!-- evictionByTime activate the data transfer for TimeBasedDataStoreGroups. Default is
false -->
   <conf:process evictionByTime="true" />
   <!-- [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="false" />
   <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="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 -->
         <!-- instrument, productType, sensing date (year, month, day) -->
         <ds:prefixLocation>instrument/productType/year/month/day</ds:prefixLocation>
      </ds:dataStore>
      <!-- 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>
         <ds:properties>
            <!-- EVICT_REFERENCE : if true, when an eviction occurs, the product will also
be evictedfromMetaStores -->
            <ds:property>
               <ds:name>EVICT_REFERENCE</ds:name>
               <ds:value>true</ds:value>
            </ds:property>
            <ds:property>
               <ds:name>EVICT_ATTACHED_FILES</ds:name>
               <ds:value>true</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>
            <!-- [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>

                  <!-- KEEP_PERIOD_SECONDS: How much time in seconds a product will be
kept in this store -->
                  <!-- .................. : It should have the same value set into gss.xml
of Consumer Ingest -->
                  <ds:property>
                     <ds:name>KEEP_PERIOD_SECONDS</ds:name>
                     <ds:value>3000</ds:value>
                  </ds:property>
               </ds:properties>
               <!-- [M] filter which products based on their name are accepted. ".*" means
all products -->
               <ds:filter>.*</ds:filter>
               <!-- [M] credentials to access the containers (defined before datastores) -
->
               <ds:credentials>SwiftCredentials</ds:credentials>
               <!-- [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 -->
               <!-- We suggest not to use underscore character -->
               <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>
         </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://IP_ADDRESS:PORT/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>SolrCloud</ms:clientType>
         <!-- [0] Solr username -->
         <ms:user>username</ms:user>
         <!-- [0] Solr password -->
         <ms:password>password</ms:password>
         <!-- [M] name of the solr collection -->
         <ms:collection>gss</ms:collection>
         <!-- [M] defaultTop : how many results will be returned for every entities that
can be listed. Default 1000 -->
         <!-- [0] maxSkip : maximum value for skip. Default is 1000 -->
         <ms:defaultTop>100</ms:defaultTop>
         <ms:maxSkip>10000</ms:maxSkip>
      </ms:metadataStore>
   </conf:metadataStores>
</conf:configuration>
```

Annex 2. Examples of Admin-API Configuration

Annex 2 contains all the configuration files and JSON examples concerning the CDH-Ingestion via Admin-API.

Annex 2.1. Examples of CDH-Admin-API – Component

Annex 2.1.1. Example of application.properties Configuration for CDH-Admin-API

```
## OAuth-2.0 Authentication configuration

# To enable oauth2 authentication, set auth = true and uncomment all
spring.security.oauth2... properties
auth = false

# Replace the 8443 with the keycloak.confidential-port
#spring.security.oauth2.resourceserver.jwt.issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>
#spring.security.oauth2.resourceserver.jwt.jwk-issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>/protocol/openid-connect/certs

### IMPORTANT ### Please ensure that the 'keycloak.client', 'keycloak.role' and
'cors.origins' are not commented out !
# if auth=true, please provide the 'keycloak Client ID' and 'keycloak Client Role.'
keycloak.client = <client-id>
keycloak.role = <client-role>

## To enable Cross Origin Requests, for example to allow the Admin UI to call this API
cors.origins = http://<localhost>:<8082>


# 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

## Server configuration

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

# Configure context path
server.servlet.contextPath=/gss-admin-api

# The two below parameter are used to not have custom errors pages
server.error.whitelabel.enabled = false
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.web.servlet.error.Erro
rMvcAutoConfiguration
```

```
# Configure the management of the HTTP Headers between the web server and the application.
# Possible values:
# - NATIVE = use the web server support for forwading headers. It allows us to
# to handle HTTPS queries inside the application and do not modify what the server
received.
# - FRAMEWORK : Spring will handle forwarded headers. It is possible that some headers
will be
# updated according to the framework.
server.forward-headers-strategy = NATIVE

# Database configuration
spring.datasource.url = jdbc:postgresql://<server.ip>:<server.port>/<database.name>
spring.datasource.username = <user.name>
spring.datasource.password = <user.password>
spring.datasource.driver-class-name = org.postgresql.Driver
# Max pool connexion size
spring.datasource.hikari.maximum-pool-size = 10
```

Annex 2.1.2. Example of Docker-Compose File for CDH-Admin-API

```
services:
  catalogue:
    image: "gaeldockerhub/cdh-admin-api:${TAG}"
    ports:
     - 8082:8082

    volumes:
     - ./application.properties:/admin-api/etc/application.properties

    networks:
     - backend

networks:
  backend:
    name: backend
    external: true
```

Annex 2.1.3. JSON Examples of CDH-Admin-API

Annex 2.1.3.1. JSON Example of CDH-Admin-API – Deletion Configuration

```
{
```

```
    "message": "<deletion_cause>",
    "reason": "<reason_name>",
    "status": "RUNNING",
    "odataFilter": "$filter=startswith(Name,'S1')",
    "nbThreads": 2
}
```

Annex 2.2. Examples of CDH-Catalogue – Component

Annex 2.2.1. Example of application.properties Configuration for CDH-Catalogue

```
## OAuth-2.0 Authentication configuration

# To enable oauth2 authentication, set auth = true and uncomment all
spring.security.oauth2... properties
auth = false

## To enable Cross Origin Requests, for example to allow the Admin UI to call this API
cors.origins = http://<localhost>:<8081>

#spring.security.oauth2.resourceserver.jwt.issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>
#spring.security.oauth2.resourceserver.jwt.jwk-issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>/protocol/openid-connect/certs

spring.security.oauth2.client.registration.keycloak.client-id=<client-id>
spring.security.oauth2.client.registration.keycloak.client-secret=<client-secret>
spring.security.oauth2.client.registration.keycloak.authorization-grant-
type=authorization_code
spring.security.oauth2.client.registration.keycloak.scope=openid,profile,roles
spring.security.oauth2.client.provider.keycloak.issuer-
uri=https://<localhost>:<8443>/<realms>/<realm-name>
spring.security.oauth2.client.provider.keycloak.user-name-attribute=preferred_username

# 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

## Server configuration

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

# Configure context path
server.servlet.contextPath = <server.servlet.contextPath>

# The two below parameter are used to not have custom errors pages
```

```
server.error.whitelabel.enabled = false
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.web.servlet.error.Erro
rMvcAutoConfiguration

# Configure the management of the HTTP Headers between the web server and the application.
# Possible values:
# - NATIVE = use the web server support for forwaded headers. It allows us to
# to handle HTTPS queries inside the application and do not modify what the server
received.
# - FRAMEWORK : Spring will handle forwarded headers. It is possible that some headers
will be
# updated according to the framework.
server.forward-headers-strategy = NATIVE

### Database configuration.
# WARNING: This part is mandatory even if you use XML configuration
spring.datasource.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
spring.datasource.username=<user.name>
spring.datasource.password=<user.password>
spring.datasource.driver-class-name=org.postgresql.Driver

deletion.kafka.hosts=<kafka.host>:<kafka.port>
deletion.kafka.topic=<kafka topic name>
deletion.datastores=datastore-name1;datastore-name2;datastore-name3
deletion.metadatastores=metadatastore-name1;metadatastore-name2
# Max pool connexion size
spring.datasource.hikari.maximum-pool-size=100
#db.socketTimeout=1000
#db.networkTimeout=5000

# To encrypt secret information in database (password for example)
secret.key.password=<encryption.value>
## End of mandatory part

# CDH Configuration
## Only used when stores are configured into DB
## Datastore's names
cdh.useDbConfiguration=true
cdh.datastores=datastore-name1;datastore-name2;datastore-name3
cdh.metadatastores=metadatastore-name1;metadatastore-name2

## Swift Configuration
#swiftConfiguration.segmentSizeMB=5000
#swiftConfiguration.retries=5
#swiftConfiguration.retryDelayMs=50
#swiftConfiguration.maxConnections=20

## ENABLE / DISABLE EVICTION BY TIME
process.evictionByTime=false

# directDownloadLink : enable the direct download from # cloud stores (swift) when using
$value. Default is true. --
# quotaDisabled : disable quotas. Default is false.
```

```
download.directDownloadLink=true
download.quotaDisabled=false
```

Annex 2.2.2. Example of Docker-Compose File for CDH-Catalogue

```
version: "3"
services:
  catalogue:
    image: "gaeldockerhub/cdh-catalogue:${TAG}"

    ports:
     - <out-port>:<in-port>

    volumes:
     - ./application.properties:/catalogue/etc/application.properties
     - <path-to-datastore-folder>/<folder-S1-name>:/ingest/folder-S1
     - <path-to-datastore-folder>/<folder-S2-name>:/ingest/folder-S2
     - <path-to-datastore-folder>/<folder-S3-name>:/ingest/folder-S3
     - <path-to-datastore-folder>/<folder-S5-name>:/ingest/folder-S5
     - ./logs:/catalogue/logs
```

Annex 2.2.3. JSON Examples of CDH-Catalogue

Annex 2.2.3.1. JSON Example of CDH-Catalogue – Subscription Configuration

```
{
    "Status": "running",
    "SubscriptionEvent": "deleted",
    "FilterParam": "Products?$filter=startswith(Name,'S1')",
    "NotificationEndpoint": "<ENDPOINT_WEB_ADDRESS>"
}
```

Annex 2.3. Examples of CDH-Ingest – Component

Annex 2.3.1. Example of application.properties Configuration for CDH-Ingest

```
# 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=<encryption-value>
```

Annex 2.3.2. Example of Docker-Compose File for CDH-Ingest

```
version: "3"
services:
  INGESTION-DHUS-SWIFT-S1:
    image: "gaeldockerhub/cdh-ingest:${TAG}"
    environment:
       CONF_FILE: /ingest/etc/database-conf-application.properties
    volumes:
       - ./INGESTION_DHUS-SWIFT_S1.properties:/ingest/etc/database-conf-
application.properties
       - /data/GSS/cdh-compose-<version-tag>/ingest/logs/:/ingest/logs
       - <path-to-destination-folder>/error:/ingest/error
       - <path-to-destination-folder>/tmp:/ingest/tmp
```

Annex 2.3.3. JSON Examples of CDH-Ingest

Annex 2.3.4. JSON Example of CDH-Ingest – Swift Credentials

```
{
    "tenant": "<tenant_value>",
    "password": "<password>",
    "user": "<username>",
    "url": "<obj_storage_url>",
    "region": "<region_name>",
    "domain": "<domain_name>",
    "name": "<swift_credentials_name>"
}
```

Annex 2.3.5. JSON Examples of CDH-Ingest – Producers

Annex 2.3.5.1. JSON Example of CDH-Ingest – Producer Folder Configuration

```
{
    "name": "<producer_name>",
    "hosts": "<IP>:<PORT>",
    "topic": "<topic_name>",
    "pushInterval": 10,
    "filter": ".*",
    "processError": {
        "active": true,
        "retries": 0
    },
    "reprocess": false,
    "dataSource": "folder",
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ProducerFolderConf",
        "path": "/path/to/source_folder"
    }
}
```

Annex 2.3.5.2. JSON Example of CDH-Ingest – Producer DHuS Configuration

```
{
    "name": "<producer_name>",
    "hosts": "<IP>:<PORT>",
    "topic": "<topic_name>",
    "pushInterval": 10,
    "filter": ".*",
    "processError": {
        "active": true,
        "retries": 0
    },
    "reprocess": false,
    "dataSource": "DHuS",
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
        "serviceRootUrl": "https://colhub.copernicus.eu/dhus/odata/v2",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": null,
            "tokenEndpoint": null,
            "type": "basic"
        },
        "top": 10,
        "lastPublicationDate": "2024-03-21T02:00:00.000Z",
        "filter": "startswith(Name,'S1') and CreationDate lt 2024-03-21T10:00:00.000Z",
        "type": "dhus",
        "assumedFormat": ".zip",
        "fetchAttributes": false,
        "fetchQuicklook": true,
        "useDateFromDb": false,
        "geoPostFilter": "POLYGON((4.4824218750000036 47.78363463526375,6.152343749999999
39.605688178320804,19.511718750000004 40.813809230569575,17.490234375000007
49.1242192485914,4.4824218750000036 47.78363463526375,4.4824218750000036
47.78363463526375))"
    }
}
```

Annex 2.3.5.3. JSON Example of CDH-Ingest – Producer GSS Configuration

```
{
    "name": "<producer_name>",
    "hosts": "<IP>:<PORT>",
    "topic": "<topic_name>",
    "pushInterval": 10,
    "filter": ".*",
    "processError": {
        "active": true,
        "retries": 0
    },
    "reprocess": false,
    "dataSource": "<Datasource_name>",
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
        "serviceRootUrl": "<GSS-Catalogue-URL>",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": "<client-id>",
            "tokenEndpoint": "<token-endpoint-url>",
            "type": "<auth-type>"
        },
        "top": 10,
        "lastPublicationDate": "2024-03-21T02:00:00.000Z",
        "filter": "startswith(Name,'S1') and CreationDate lt 2024-03-21T10:00:00.000Z",
        "type": "csc",
        "assumedFormat": null,
        "fetchAttributes": false,
        "fetchQuicklook": true,
        "useDateFromDb": false,
        "geoPostFilter": "POLYGON((4.4824218750000036 47.78363463526375,6.152343749999999
39.605688178320804,19.511718750000004 40.813809230569575,17.490234375000007
49.1242192485914,4.4824218750000036 47.78363463526375,4.4824218750000036
47.78363463526375))"
    }
}
```

Annex 2.3.5.4. JSON Example of CDH-Ingest – Producer CDSE Configuration

```
{
    "name": "<producer_name>",
    "hosts": "<IP>:<PORT>",
    "topic": "<topic_name>",
    "pushInterval": 40,
    "filter": ".*",
    "processError": {
        "active": true,
        "retries": 0
    },
    "reprocess": false,
    "dataSource": "CDSE",
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ProducerOdataConf",
        "serviceRootUrl": "https://catalogue.dataspace.copernicus.eu/odata/v1",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": "cdse-public",
            "tokenEndpoint":
"https://identity.dataspace.copernicus.eu/auth/realms/CDSE/protocol/openid-connect/token",
            "type": "oauth2"
        },
        "top": 10,
        "lastPublicationDate": "2024-03-21T02:00:00.000Z",
        "filter": "startswith(Name,'S1') and CreationDate lt 2024-03-21T10:00:00.000Z",
        "type": "cdse",
        "assumedFormat": ".zip",
        "fetchAttributes": false,
        "fetchQuicklook": true,
        "useDateFromDb": false,
        "geoPostFilter": "POLYGON((4.4824218750000036 47.78363463526375,6.152343749999999
39.605688178320804,19.511718750000004 40.813809230569575,17.490234375000007
49.1242192485914,4.4824218750000036 47.78363463526375,4.4824218750000036
47.78363463526375))"
    }
}
```

Annex 2.3.6. JSON Examples of CDH-Ingest – Consumers

Annex 2.3.6.1. JSON Example of CDH-Ingest – Consumer Folder with Error Manager Folder Destination

```
{
    "name": "<consumer-name>",
    "parallelIngests": 4,
    "hosts": "<IP>:<PORT>",
    "groupId": "<topic-group-name>",
    "topics": "<topic-name>",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 20000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ConsumerFolderConf",
        "path": "/path/to/source_folder"
    },
    "taskList": [
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<timebased-datastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<metadatastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "forceOnline": false
        },
```

```
        {
            "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "onlyUseProvidedQL": true,
            "height": 45,
            "width": 45,
            "targetStores": "<quicklook-datastore-name>"
        }
    ],
    "tmpPath": "/path/to/tmp/folder",
    "errorManager": {
        "errorLocation": "/path/to/error/folder",
        "container": null,
        "type": "folder",
        "credentials": null
    }
}
```

Annex 2.3.6.2. JSON Example of CDH-Ingest – Consumer Folder with Error Manager Swift Destination

```
{
    "name": "<consumer-name>",
    "parallelIngests": 4,
    "hosts": "<IP>:<PORT>",
    "groupId": "<topic-group-name>",
    "topics": "<topic-name>",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 20000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ConsumerFolderConf",
        "path": "/path/to/source_folder"
    },
    "taskList": [
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
            "pattern": ".*",
            "stopOnFailure": true,
```

```
            "tryLimit": 5,
            "active": true,
            "targetStores": "<swift_timebased_datastore_name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<metadatastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "forceOnline": false
        },
        {
            "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "onlyUseProvidedQL": true,
            "height": 45,
            "width": 45,
            "targetStores": "<quicklook_swift_datastore_name>"
        }
    ],
    "tmpPath": "/path/to/tmp/folder",
    "errorManager": {
        "errorLocation": null,
        "container": "<container_error_name>",
        "type": "swift",
        "credentials": "<swift_credentials>"
    }
}
```

Annex 2.3.6.3. JSON Example of CDH-Ingest – Consumer DHuS with Error Manager Folder Destination

```
{
    "name": "<consumer-name>",
    "parallelIngests": 10,
    "hosts": "<IP>:<PORT>",
    "groupId": "<topic-group-name>",
    "topics": "<topic_name>",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 40000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
        "serviceRootUrl": "https://colhub.copernicus.eu/dhus/odata/v2",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": null,
            "tokenEndpoint": null,
            "type": "basic"
        },
        "type": "dhus",
        "retriesOn429": 10,
        "retryWaitOn429Ms": 5000
    },
    "taskList": [
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<timebased-datastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<metadatastore-name>"
        },
        {
```

```
            "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "forceOnline": false
        },
        {
            "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
            "pattern": ".*",
            "stopOnFailure": false,
            "tryLimit": 5,
            "active": true,
            "onlyUseProvidedQL": true,
            "height": 45,
            "width": 45,
            "targetStores": "<quicklook-datastore-name>"
        }
    ],
    "tmpPath": "/path/to/tmp/folder",
    "errorManager": {
        "errorLocation": "/path/to/error/folder",
        "container": null,
        "type": "folder",
        "credentials": null
    }
}
```

Annex 2.3.6.4. JSON Example of CDH-Ingest – Consumer DHuS with Error Manager Swift Destination

```
{
    "name": "<consumer-name>",
    "parallelIngests": 10,
    "hosts": "<IP>:<PORT>",
    "groupId": "<topic-group-name>",
    "topics": "<topic_name>",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 30000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
```

```
        "serviceRootUrl": "https://colhub.copernicus.eu/dhus/odata/v2",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": null,
            "tokenEndpoint": null,
            "type": "basic"
        },
        "type": "dhus",
        "retriesOn429": 10,
        "retryWaitOn429Ms": 5000
    },
    "taskList": [
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<swift_timebased_datastore_name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<metadatastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "forceOnline": false
        },
        {
            "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
            "pattern": ".*",
            "stopOnFailure": false,
            "tryLimit": 5,
            "active": true,
            "onlyUseProvidedQL": true,
            "height": 45,
            "width": 45,
```

```
            "targetStores": "<quicklook_swift_datastore_name>"
        }
    ],
    "tmpPath": "/path/to/tmp/folder",
    "errorManager": {
        "errorLocation": null,
        "container": "<container_error_name>",
        "type": "swift",
        "credentials": "<swift_credentials>"
    }
}
```

Annex 2.3.6.5. JSON Example of CDH-Ingest – Consumer GSS with Error Manager Folder Destination

```
{
    "name": "<consumer-name>",
    "parallelIngests": 10,
    "hosts": "<IP>:<PORT>",
    "groupId": "<topic-group-name>",
    "topics": "<topic-name>",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 40000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
        "serviceRootUrl": "<GSS-Catalogue-URL>",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": "<client-id>",
            "tokenEndpoint": "<token-endpoint-url>",
            "type": "<authentication-type>"
        },
        "type": "csc",
        "retriesOn429": 10,
        "retryWaitOn429Ms": 5000
    },
    "taskList": [
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
            "pattern": ".*",
```

```
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<timebased-datastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<metadatastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "forceOnline": false
        },
        {
            "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
            "pattern": ".*",
            "stopOnFailure": false,
            "tryLimit": 5,
            "active": true,
            "onlyUseProvidedQL": true,
            "height": 45,
            "width": 45,
            "targetStores": "<quicklook-datastore-name>"
        }
    ],
    "tmpPath": "/path/to/tmp/folder",
    "errorManager": {
        "errorLocation": "/path/to/error/folder",
        "container": null,
        "type": "folder",
        "credentials": null
    }
}
```

Annex 2.3.6.6. JSON Example of CDH-Ingest – Consumer CSC (GSS) with Error Manager Swift Destination

```
{
    "name": "<consumer-name>",
    "parallelIngests": 10,
    "hosts": "<IP>:<PORT>",
    "groupId": "<topic-group-name>",
    "topics": "<topic-name>",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 40000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
        "serviceRootUrl": "<GSS-Catalogue-url>",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": "<client-id>",
            "tokenEndpoint": "<token-endpoint-url>",
            "type": "<authentication-type>"
        },
        "type": "csc",
        "retriesOn429": 10,
        "retryWaitOn429Ms": 5000
    },
    "taskList": [
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<swift_timebased_datastore_name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<metadatastore-name>"
        },
        {
```

```
            "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "forceOnline": false
        },
        {
            "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
            "pattern": ".*",
            "stopOnFailure": false,
            "tryLimit": 5,
            "active": true,
            "onlyUseProvidedQL": true,
            "height": 45,
            "width": 45,
            "targetStores": "<quicklook_swift_datastore_name>"
        }
    ],
    "tmpPath": "/path/to/tmp/folder",
    "errorManager": {
        "errorLocation": null,
        "container": "<container_error_name>",
        "type": "swift",
        "credentials": "<swift_credentials>"
    }
}
```

Annex 2.3.6.7. JSON Example of CDH-Ingest – Consumer CDSE with Error Manager Folder Destination

```
{
    "name": "<consumer-name>",
    "parallelIngests": 1,
    "hosts": "<IP>:<PORT>",
    "groupId": "<topic-group-name>",
    "topics": "<topic_name>",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 60000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
```

```
        "serviceRootUrl": "https://catalogue.dataspace.copernicus.eu/odata/v1",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": "cdse-public",
            "tokenEndpoint":
"https://identity.dataspace.copernicus.eu/auth/realms/CDSE/protocol/openid-connect/token",
            "type": "oauth2"
        },
        "type": "cdse",
        "retriesOn429": 10,
        "retryWaitOn429Ms": 5000
    },
    "taskList": [
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<timebased-datastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<metadatastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "forceOnline": false
        },
        {
            "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
            "pattern": ".*",
            "stopOnFailure": false,
            "tryLimit": 5,
            "active": true,
            "onlyUseProvidedQL": true,
            "height": 45,
```

```
            "width": 45,
            "targetStores": "<quicklook_datastore_name>"
        }
    ],
    "tmpPath": "/path/to/tmp/folder",
    "errorManager": {
        "errorLocation": "/path/to/error/folder",
        "container": null,
        "type": "folder",
        "credentials": null
    }
}
```

Annex 2.3.6.8. JSON Example of CDH-Ingest – Consumer CDSE with Error Manager Swift Destination

```
{
    "name": "<consumer-name>",
    "parallelIngests": 1,
    "hosts": "<IP>:<PORT>",
    "groupId": "<topic-group-name>",
    "topics": "<topic_name>",
    "topicPattern": null,
    "reprocess": false,
    "pollIntervalMs": 60000,
    "sourceDelete": false,
    "ingestThreads": 3,
    "source": {
        "sourceType": "fr.gael.gss.ingest.ingester.ConsumerOdataConf",
        "serviceRootUrl": "https://catalogue.dataspace.copernicus.eu/odata/v1",
        "auth": {
            "user": "<username>",
            "password": "<password>",
            "clientId": "cdse-public",
            "tokenEndpoint":
"https://identity.dataspace.copernicus.eu/auth/realms/CDSE/protocol/openid-connect/token",
            "type": "oauth2"
        },
        "type": "cdse",
        "retriesOn429": 10,
        "retryWaitOn429Ms": 5000
    },
    "taskList": [
        {
```

```
            "type": "fr.gael.gss.ingest.ingester.IngestInDataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<swift_timebased_datastore_name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.IngestInMetadataStores",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "targetStores": "<metadatastore-name>"
        },
        {
            "type": "fr.gael.gss.ingest.ingester.ExtractMetadata",
            "pattern": ".*",
            "stopOnFailure": true,
            "tryLimit": 5,
            "active": true,
            "forceOnline": false
        },
        {
            "type": "fr.gael.gss.ingest.ingester.CreateQuicklook",
            "pattern": ".*",
            "stopOnFailure": false,
            "tryLimit": 5,
            "active": true,
            "onlyUseProvidedQL": true,
            "height": 45,
            "width": 45,
            "targetStores": "<quicklook_swift_datastore_name>"
        }
    ],
    "tmpPath": "/path/to/tmp/folder",
    "errorManager": {
        "errorLocation": null,
        "container": "<container_error_name>",
        "type": "swift",
        "credentials": "<swift_credentials>"
    }
}
```

Annex 2.3.7. JSON Examples of CDH-Ingest – Datastores

Annex 2.3.7.1. JSON Example of CDH-Ingest – HFS Configuration

```
{
    "name": "<quicklook_datastore_name>",
    "permission": [
        "READ",
        "WRITE",
        "DELETE"
    ],
    "properties": {
        "property": [
            {
                "name": "STORE_ATTACHED_FILES",
                "value": "true"
            }
        ]
    },
    "path": "/path/to/folder",
    "depth": 0,
    "granularity": 2,
    "parentGroup": null,
    "type": "HFS"
}
```

Annex 2.3.7.2. JSON Example of CDH-Ingest – HFS (TimeBased) Configuration

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

```
                "name": "EVICT_ATTACHED_FILES",
                "value": "true"
            }
        ]
    },
    "name": "<timebased_datastore_name>",
    "filter": ".*",
    "policy": "BASIC_STORE_PRIORITY_POLICY",
    "children": [
        {
            "permission": [
                "READ",
                "WRITE",
                "DELETE"
            ],
            "properties": {
                "property": [
                    {
                        "name": "EVICT_REFERENCE",
                        "value": "true"
                    },
                    {
                        "name": "EVICT_ATTACHED_FILES",
                        "value": "true"
                    },
                    {
                        "name": "STORE_BY_NAME",
                        "value": "true"
                    },
                    {
                        "name": "SAVE_AS_MULTIPART",
                        "value": "false"
                    },
                    {
                        "name": "KEEP_PERIOD_SECONDS",
                        "value": "<keep_period_value>"
                    }
                ]
            },
            "name": "datastore_name",
            "path": "/path/to/folder",
            "depth": 0,
            "granularity": 2,
            "parentGroup": "<timebased_datastore_name>",
            "type": "HFS"
        }
```

```
    ],
    "type": "TIME_GROUP"
}
```

Annex 2.3.7.3. JSON Example of CDH-Ingest – SWIFT Configuration

```
{
    "name": "<quicklook_swift_datastore_name>",
    "permission": [
        "READ",
        "WRITE",
        "DELETE"
    ],
    "properties": {
        "property": [
            {
                "name": "STORE_ATTACHED_FILES",
                "value": true
            },
            {
                "name": "STORE_BY_NAME",
                "value": true
            }
        ]
    },
    "credentials": "<swift_credentials>",
    "container": "<container_name>",
    "prefixLocation": "instrument/productType/year/month/day",
    "parentGroup": null,
    "type": "SWIFT"
}
```

Annex 2.3.7.4. JSON Example of CDH-Ingest – SWIFT (TimeBased) Configuration

```
{
    "name": "<swift_timebased_datastore_name>",
    "type": "timeBasedDataStoreGroupConf",
    "permission": [
        "READ",
        "WRITE",
```

```
            "DELETE"
        ],
        "properties": {
            "property": [
                {
                    "name": "EVICT_REFERENCE",
                    "value": true
                },
                {
                    "name": "EVICT_ATTACHED_FILES",
                    "value": true
                }
            ]
        },
        "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": "<keep_period_value>"
                        }
                    ]
                },
                "name": "<swift_datastore_name>",
                "credentials": "<swift_credentials>",
                "filter": ".*",
                "containerPattern": {
                    "type": "mapperPattern",
                    "patternMapper": "<Sentinel_mission_tag>.*:<container_name>"
                },
```

```
            "prefixLocation": "instrument/productType/year/month/day",
            "parentGroup": "<swift_timebased_datastore_name>",
            "type": "SWIFT_GROUP"
        }
    ]
}
```

Annex 2.3.8. JSON Example of CDH-Ingest – Metadata Store

```
{
    "name": "<metadatastore_name>",
    "permission": [
        "READ",
        "WRITE",
        "DELETE"
    ],
    "properties": null,
    "strategies": null,
    "hosts": "http://<IP>:<PORT>/solr",
    "clientType": "SolrCloud",
    "user": null,
    "password": null,
    "collection": "<collection_name>",
    "defaultSort": null,
    "defaultTop": 100,
    "maxSkip": 10000,
    "storage": null,
    "visitorBuilder": "fr.gael.gss.core.store.solr.ProductSolrVisitorBuilder",
    "transformer": "fr.gael.gss.core.store.solr.ProductSolrTransformer"
}
```

Annex 2.3.9. JSON Example of CDH-Ingest – QUOTA

```
[
    {
        "name": "TOTAL_DOWNLOAD",
        "userId": "<preferred-username>",
        "value": 1000000000,
        "duration": 20
    },
    {
        "name": "TOTAL_DOWNLOAD_LINK",
        "userId": "<preferred-username>",
```

```
        "value": 50,
        "duration": 1
    },
    {
        "name": "PARALLEL_DOWNLOAD",
        "userId": "<preferred-username>",
        "value": 30,
        "duration": 0
    }
]
```

Annex 2.4. Examples of CDH-Notification – Component

Annex 2.4.1. Examples of consumer.properties Configuration for CDH-Notification

```
### Catalogue
catalogue.path =<catalogue URL>

### KAFKA
# MANDATORY - Comma separated list of kafka brokers (ip:port)
kafka.hosts =<kafka.host>:<kafka.port>

# MANDATORY - Semi-colon separated list of topics
kafka.topics = <kafka topic name>

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

# MANDATORY - Database information
# DB configuration
db.url=jdbc:postgresql://<server.ip>:<server.port>/<database.name>
db.user=<user.name>
db.password=<user.password>
db.poolSize=10
#db.socketTimeout=
#db.networkTimeout=

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

```
# OPTIONAL - Prefix for kafka notifications topics names
#kafka.topics.prefix = CDHN01-
```

Annex 2.4.2. Examples of Docker-Compose File for CDH-Notification

```
version: "3"
services:
  notification:
    image: "gaeldockerhub/cdh-notification:${TAG}"
# add port configuration in case of distributed env
    #ports:
     #- <out-port>:<in-port>
    environment:
       CONF_FILE: /notification/etc/consumer.properties

    volumes:
     - ./consumer-for-notification.properties:/notification/etc/consumer.properties
     - /data/GSS/cdh-compose-<version-tag>/notification/logs/:/notification/logs
```