Interface Control Document
Reference System core

Prepared (Author)

Signature : **Date :**

Name / Function **Pierre CUQ**
Reference System Engineer

Approved

Signature : **Date :**

Name / Function **Jean-Luc PASCAL**
Reference System Quality Manager

Pierre CUQ
Reference System Engineer

Released

Signature : **Date :**

Name / Function **Benjamin BRARD**

Reference System project manager

Document type	**Nb WBS**	**Keywords**
ICD	XXX	

CMMI-SVC	CM

Export Control Information

This document contains EU or/and US Export Controlled technology (data) :

YES □ **NO** ☒

If **YES** :

1/ European/French regulation controlled content

□ Technology contained in this document is controlled by the European Union in accordance with dual-use regulation 428/2009 under Export Control Classification Number [xExx]. ***(1)***

□ Technology contained in this document is controlled by Export Control regulations of French Munitions List under Export Control Classification Number [MLXX or AMAXX]. ***(1)***

2/ US Regulation controlled content

□ Technology contained in this document is controlled under Export Control Classification Number [xExxx] by the U.S. Department of Commerce - Export Administration Regulations (EAR). ***(1)***

□ Technology contained in this document is controlled by the U.S. Department of State - Directorate of Defense Trade Controls - International Traffic in Arms Regulations (ITAR). ***(1)***

(1) See applicable export control license/authorization/exception in Delivery Dispatch Note.

SUMMARY

This ICD describes the Reference System core components format.

CHANGE LOG

Issue/Revision	Date	Change Requests	Observations
01 draft A	04 / 01 / 2022		First version
01 draft B	26 / 01 / 2022	SSN_002, FSI_001, FSI_005, FSI_006	FCR - RIDs from Reading_sheet-doc-COPRS-ICD-ADST-001133963 Update Artifactory directories.
01	09 /03 / 2022		Update after RIDS meeting. ● Remove deploymentLabels property. ● Namespace is no more immutable. ● Add mongodb url. ● Apply RS-addon ZIP structure to RS core. ● Add the release note section. ● Add sample chapter ● Add rule for topic naming ● Update chapters number
1.1	08/04/02022	§4.1 §4.2	The date has been removed from the filename. The configuration parameter list has been added to the release note.
2.0			Rename version 2.0 . No content change.
3.0	02 / 05 / 2022	§4.4.2	Update : ● Allow multi-line DSL definition.
4.0	05 / 05 / 2022	§3.2.3 §3.2.4 §4.4.2.5 §4.4	Set Kafka, secret and ES URL (cf #373) Add a wiring chapter to define the topic name. Remove “description” and “name” fields. Add comment format.
5.0	13 / 05 / 2022	§3.2.4	Set a single topic for error-warning. Add the possibility to be connected to a trace-event topic.
6.0 draft A	22 / 07 / 2022	§2.1	Update the list of the identifiers of the core chains
6.0 draft B	29 / 08 / 2022	§1.2.4	Add compression-event to the list of topics

6.0	05/09/2022		Following story #504, add option to deploy the RS add-on on other namespace. **Remove draft.** Version for system RS v1.1.

TABLE OF CONTENTS

1 General Information

1.1 Document purpose

This document describes the Reference System core components format.

1.2 Glossary and abbreviations

Term	Meaning
CFI	Customer Furnished Item.
DPU	A Data Process Unit is a specific Process Unit that wraps a Sentinel Data Processor binaries and their static data.
DSL	Domain Specific Language
PU	A Process Unit is a microservice, packaged as ready-to-be-deployed Docker, representing the simplest step that can be included into a Reference System workflow.
RS	Reference System
RS add-on	A RS add-on is an autonomous package ready for deployment on cloud on top of RS platform. It provides a Sentinel processing chain.
RS core	A RS core is a specific RS add-on. It has the same format. It is deployed in the same way. It is part of the RS platform as a core service.
SCDF	Spring Cloud Data Flow is the workflow manager for RS platform.
SDP	A Sentinel Data Processor is a processing CFI provided by ESA.
Workflow manager	The workflow manager is responsible for creating a workflow instance from its source template. In particular, it gathers configuration values, negotiates execution resources and downloads the required Process Unit on the target nodes. SCDF is the workflow manager for the RS platform.

2 Applicables and referenced documents

2.1 Applicable documents

Identifier	Document name	Reference	Version
[RD1]	Non functional requirements for Cloud Deployment & Configuration	COPRS-SP-ADST-001046261	1.0

2.2 Reference documents

N/A

3 RS core into Reference System

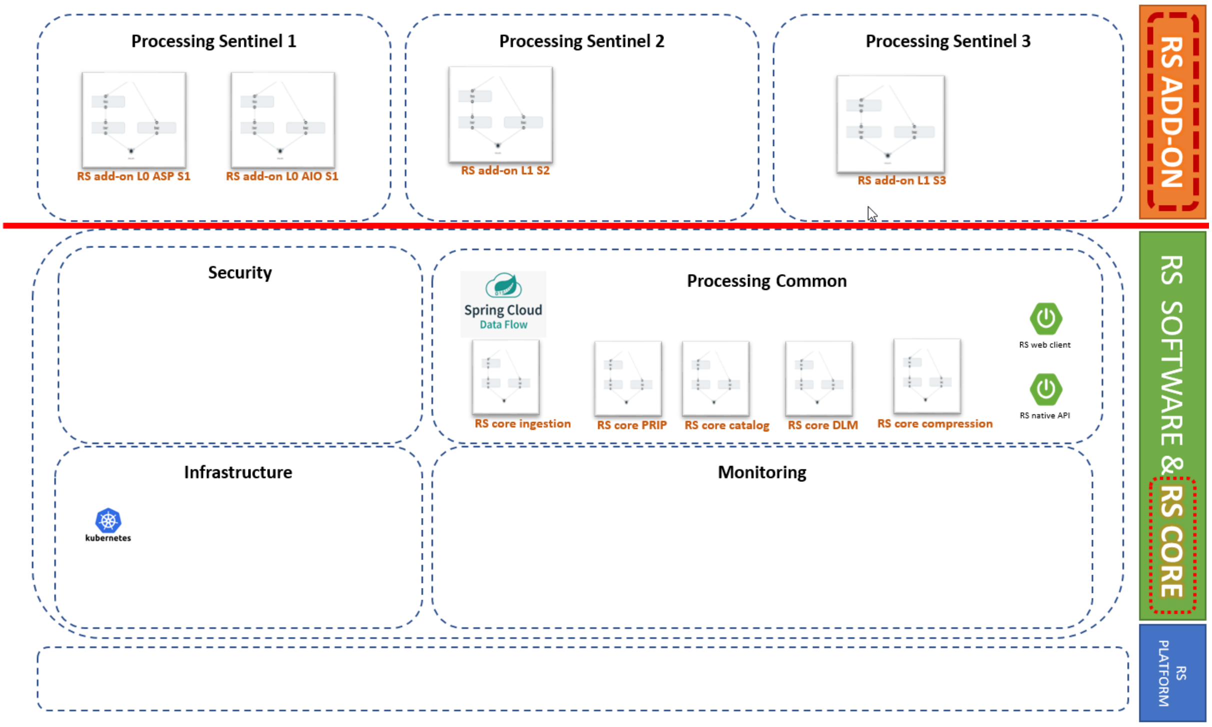
3.1 Reference System

Copernicus Reference System runs Sentinel 1, Sentinel 2 and Sentinel 3 workflows. Some workflows are common to all Sentinel processing. Workflows on the RS platform are managed by Spring Cloud Data Flow.

3.2 RS add-on and RS core

A RS add-on and RS core are both providing a Sentinel processing chain :

- A RS add-on is deployed on top of the RS platform.
- A RS core is part of the RS platform itself. It provides a common processing chain to Sentinel-1, Sentinel-2 and Sentinel-3.



RS add-on and RS core payload format are equal.

⚠ The RS core is provided with factory settings. The chain needs to be instantiated with an operational setting for deployment.

1.1 RS CORE STORAGE

COPRS Fonction 3 builds PU containers and pushes them to the public portion of COPRS Function 2 Artifactory.

COPRS Fonction 3 builds an RS core package which points to PU containers stored in Function 2 Artifactory. Source code RS core package is versioned on Function 2 GitHub.

The table below summarises storage location for RS core.

	Function 2 Artifactory	Function 2 GitHub
RS core	Public PU containers	RS core source code and template properties files

1.2 RS PLATFORM

1.2.1 Compatibility

RS core components shall be compliant with :

FOSS	Version
SCDF	>= 2.9.1

1.2.2 Container

All containers that are part of RS core shall fulfil requirements from [RD1], applicable to containers.

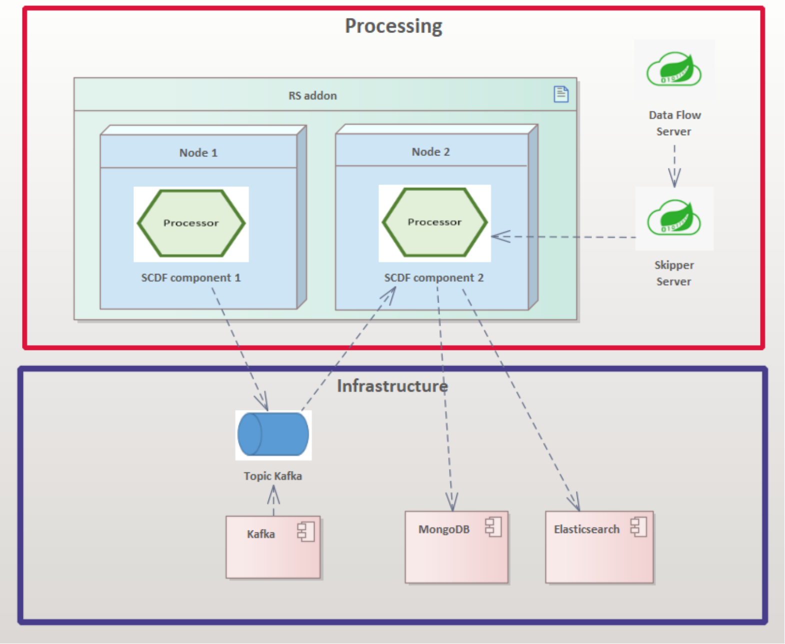
All containers that are part of RS core shall be stored on the following registry root path.

Registry name	Registry path
Jfrog Artifactory	https://artifactory.coprs.esa-copernicus.eu

For each RS core, we propose such sub directories :

Mission	Sub path
ALL	https://artifactory.coprs.esa-copernicus.eu/rs-docker/rs-core

1.2.3 Services



RS components can access storage service through:

Service name	Version	Namespace	URL	Secret
MongoDB	v5.0.3	infra	mongodb-0.mongodb-headless.database.svc.cluster.local, mongodb-1.mongodb-headless.database.svc.cluster.local, mongodb-2.mongodb-headless.database.svc.cluster.local	Yes
ElasticSearch	v7.15.2	infra	elasticsearch-processing-es-coordinating.database.svc.cluster.local	No

To access secret and Kafka messaging services, these properties shall be set :

- app.*.spring.kafka.bootstrap-servers: kafka-cluster-kafka-bootstrap.infra.svc.cluster.local:9092
- app deployer.*.kubernetes.imagePullSecrets: spring-cloud-dataflow-registry-dockersecret

1.2.4 Wiring

For nominal cases, a RS core can only be connected to the following topics :

- catalog-job
- catalog-event
- compression-event
- trace-event

Error and warning messages are pushed to topic : error-warning.

2 RS CORE STRUCTURE

A RS core payload format is the same as the RS add-on payload format.

2.1 FILENAME COMPRESSED FORMAT

A RS core component will be a ZIPPED file.

Filename:

The filename format is : RS_CORE_[%PROCESSORID%]_[%REL%].ZIP

where

- [%PROCESSORID%] is the identifier of the core chain : INGESTION, METADATA, DATALIFECYCLE, COMPRESSION, DISTRIBUTION, QUICKLOOK
- [%REL%] is the unique identifier of the current release (e.g. 3.2.1).

Filename sample:

- RS_CORE_INGESTION_1.2.3a.zip
- RS_CORE_METADA_2.3.3c.zip

By opening the ZIP file, we will found the following items:

- RS_CORE_[%PROCESSORID%]_[%REL%]_Release_Note.pdf
- **/RS_CORE_[%PROCESSORID%]_[%REL%]_Executables**
  - stream-application-list.properties
  - stream-definition.properties
  - stream-parameters.properties
  - **/additional_resources/** *(optional)*
    - KafkaTopic1.yml
    - KafkaTopic2.yml

The items in bold are directories.

2.2 Release note

The release note is a single PDF file.It describes briefly the product and details about specific changes from previous release. It provided information about the RS add-on resource needed for execution.

At the minimum, the following elements shall be provided:

Ressource	Value
CPU	Min number of vCore
Memory	Min number of GigaBytes
Disk volume needed	YES / NO
Disk access	ReadWriteOnce / ReadOnlyMany / ReadWriteMany / ReadWriteOncePod
Disk storage capacity	Min number of GigaBytes
Affinity between POD/Node	To be defined if needed.
Configuration parameters list	The list of configurable parameters for the RS core chain.

The primary target audience is the IVV team and Production Service development team.

2.3 Executables

This is the RS core payload.

Payload files are stored into a directory named : /RS_CORE_[%PROCESSORID%]_[%REL%]_Executables

The payload is composed of 3 main files and an optional directory:

- stream-application-list.properties
- stream-definition.properties
- stream-parameters.properties
- **/additional_resources/** *(optional)*

Directory name sample:

- /RS_CORE_INGESTION_1.3.8_Executables
- /RS_CORE_ METADATA_2.3.3c_Executables

2.4 Payload format

The RS core payload is composed of: :

1. SCDF stream containers list: PU and DPU container images.
2. SCDF stream definition : description of the workflow between the containers.
3. SCDF stream properties: properties for the POD build and the application dedicated
4. Services settings : directory with operators to set each service. This is an optional section.

On each properties files (item 1 to 3), comments can be added to the file with prefix #.

```
Any text
# my comment here
# another comment here
Any text
```

2.4.1 SCDF stream containers list

Filename : stream-application-list.properties

Content file format :
<type>.<name>=docker:<docker-image-path>/<imageName>:<version> (on line per application)

where ***<type>*** is equal to ***sink*** , ***source*** or ***processor***.

Sample RS core (with 3 PU) :

```
source.ingestion-trigger=docker:artifactory.coprs.esa-copernicus.eu/docker-name1:1.2.0
processor.ingestion-filter=docker:artifactory.coprs.esa-copernicus.eu/docker-name2:2.4.0.BUILD-SNAPSHOT
processor.ingestion-worker=docker:artifactory.coprs.esa-copernicus.eu/docker-name3:3.5.0
```

2.4.2 SCDF stream definition

2.4.2.1 Application

A stream is defined by using a Unix-inspired Pipeline syntax. The syntax uses vertical bars, known as “pipes”, to connect multiple commands. In Data Flow, the Unix command is replaced by a Spring Cloud Stream application and each pipe symbol represents connecting the input and output of applications over messaging middleware Apache Kafka.

Sample :

```
applicationName1 | applicationName2 | applicationName3
applicationName2 | applicationName4
applicationName3 | applicationName4 | applicationName5
```

2.4.2.2 Named destination

Instead of referencing a source or sink application, you can use a named destination. A named destination corresponds to a specific destination name in the middleware broker Kafka. When using the | symbol, applications are connected to each other with messaging middleware destination names created by the Data Flow server. In keeping with the Unix analogy, you can redirect standard input and output using the less-than (<) and greater-than (>) characters. To specify the name of the destination, prefix it with a colon (:).

Sample 1: `:myDestination > applicationName1`

Sample 2: `applicationName2 > :myDestination`

See : https://dataflow.spring.io/docs/feature-guides/streams/named-destinations/ for details.

2.4.2.3 Fan-In and Fan-Out

By using named destinations, you can support fan-in and fan-out use cases. Fan-in use cases are when multiple sources all send data to the same named destination.

The fan-out use case is when you determine the destination of a stream based on some information that is known only at runtime.

See : https://dataflow.spring.io/docs/feature-guides/streams/fanin-fanout/ for details.

2.4.2.4 Wiring

In cases with multiple input and output bindings, Data Flow cannot make any assumptions about the flow of data from one application to another. Therefore, you need to set the binding properties to "wire up" the application. The *Stream Application DSL* uses a "double pipe" (instead of the "pipe symbol") to indicate that Data Flow should not configure the binding properties of the application. Think of `||` as meaning "in parallel".

Sample : `applicationName1 || applicationName2 || applicationName3`

See : https://dataflow.spring.io/docs/feature-guides/streams/stream-application-dsl/ for details.

2.4.2.5 Stream Format

Filename : stream-definition.properties

Content format :

```
:myDestination1 > applicationName1 | applicationName2 || applicationName3  >:myDestination2
applicationName1 | applicationName8
```

Sample :

```
ingestion-trigger | ingestion-filter | ingestion-worker > :ingested-element
ingestion-filter  | other-application
```

2.4.2.6 Stream name

When the "RS core" is deployed, it receives a name.

In a general case, the stream is defined by several lines. As a consequence, each line of the RS core is identifies as follow :

```
<RS-ADD-ON-NAME>-partx   where x>0. Start from 1 and is incremented.
```

Sample :

To deploy the RS core "INGESTION" with the name "**sentine1-xbip-MTI**".

```
sentinel1-xbip-MTI-part1
sentinel1-xbip-MTI-part2
sentinel1-xbip-MTI-part3
sentinel1-xbip-MTI-part4
```

2.4.3 SCDF stream properties

All properties of the stream are grouped on a single file named : `stream-parameters.properties` .

Properties fall into three groups:

- Deployer Properties: These properties control how the apps are deployed to the target platform and use a deployer prefix.
- Application binding properties (optional section) : binding properties (if use || on DSL definition, see §2.6.2.4).
- Application custom properties: These properties control or override how the application behaves and are set during stream creation.

2.4.3.1 Deployer properties

(source : https://docs.spring.io/spring-cloud-dataflow/docs/current/reference/htmlsingle/#configuration-kubernetes-deployer)

Kubernetes properties shall be draft as follow :

```
deployer.<application>.kubernetes.<property>=<value>
```

Mandatory properties to be set

Property	Description	Immutable	Example
namespace	Namespace to use	NO	myNs
livenessProbeDelay	Delay in seconds when the Kubernetes liveness check of the app container should start checking its health status.	YES	1234
livenessProbePeriod	Period in seconds for performing the Kubernetes liveness check of the app container.	YES	1234
livenessProbeTimeout	Timeout in seconds for the Kubernetes liveness check of the app container. If the health check takes longer than this value to return it is assumed as 'unavailable'.	YES	1234
livenessProbePath	Path that app container has to respond to for liveness check.	YES	/myProbe
livenessProbePort	Port that app container has to respond on for liveness check.	YES	1234
readinessProbeDelay	Delay in seconds when the readiness check of the app container should start checking if the module is fully up and running.	YES	1234

Property	Description	Immutable	Example
readinessProbePeriod	Period in seconds to perform the readiness check of the app container.	YES	1234
readinessProbeTimeout	Timeout in seconds that the app container has to respond to its health status during the readiness check.	YES	1234
readinessProbePath	Path that the app container has to respond to for readiness check.	YES	/myProbe
readinessProbePort	Port that the app container has to respond to for readiness check.	YES	1234
limits.memory	The memory limit, maximum needed value to allocate a pod, Default unit is mebibytes, 'M' and 'G" suffixes supported	YES	1234
limits.cpu	The CPU limit, maximum needed value to allocate a pod	YES	1234
requests.memory	The memory request, guaranteed needed value to allocate a pod.	YES	1234
requests.cpu	The CPU request, guaranteed needed value to allocate a pod.	YES	1234
maxTerminatedErrorRestarts	Maximum allowed restarts for apps that fail due to an error or excessive resource use.	YES	1234

Optional properties to be set

Property	Description	Immutable	default
probeCredentialsSecret	The secret name contains the credentials to use when accessing secured probe endpoints.	NO	<none>
statefulSet.volumeClaimTemplate.storageClassName	Name of the storage class for a stateful set	YES	<none>
statefulSet.volumeClaimTemplate.storage	The storage amount. Default unit is mebibytes, 'M' and 'G" suffixes supported	YES	<none>
environmentVariables	List of environment variables to set for any deployed app container	YES	<none>
volumeMounts	volume mounts expressed in YAML format. e.g. `[{name: 'testhostpath', mountPath: '/test/hostPath'}, {name: 'testpvc', mountPath: '/test/pvc'}, {name: 'testnfs', mountPath: '/test/nfs'}]`	YES	<none>
volumes	The volumes that a Kubernetes instance supports are specified in YAML format. e.g. `[{name: testhostpath, hostPath: { path: '/test/override/hostPath' }},{name: 'testpvc', persistentVolumeClaim: { claimName: 'testClaim', readOnly: 'true' }}, {name: 'testnfs', nfs: { server: '10.0.0.1:111', path: '/test/nfs' }}]`	YES	<none>
secretRefs	The name of the secret(s) to load the entire data contents into individual environment variables. Multiple secrets may be comma separated.	NO	<none>
secretKeyRefs.envVarName	The environment variable name to hold the secret data	NO	<none>
secretKeyRefs.secretName	The secret name to access	NO	FALSE
secretKeyRefs.dataKey	The key name to obtain secret data from	NO	<none>
configMapRefs	The name of the ConfigMap(s) to load the entire data contents into individual environment variables. Multiple ConfigMaps are comma separated.	YES	<none>

Property	Description	Immutable	default
configMapKeyRefs.envVarName	The environment variable name to hold the ConfigMap data	YES	<none>
configMapKeyRefs.configMapName	The ConfigMap name to access	YES	<none>
configMapKeyRefs.dataKey	The key name to obtain ConfigMap data from	YES	<none>
maximumConcurrentTasks	The maximum concurrent tasks allowed for this platform instance.	YES	<none>
statefulSetInitContainerImageName	A custom image name to use for the StatefulSet Init Container	YES	<none>
initContainer	An Init Container expressed in YAML format to be applied to a pod. e.g. `{containerName: 'test', imageName: 'busybox:latest', commands: ['sh', '-c', 'echo hello']}`	YES	<none>

By default all the PODs of the stream are deployed into the namespace “**processing**”.

If you want to deploy the POD of the stream on another namespace, add the following configuration line:

```
spring.cloud.dataflow.skipper.platformName=<namespace>
```

2.4.3.2 Application binding properties

This is an optional section in case the DSL definition include “||“ (see §2.6.2.4).

You can use default binding method if there is a single input/output:

```
app.<application>.spring.cloud.stream.bindings.input.destination=<kafkaQueueName>
app.<application>.spring.cloud.stream.bindings.output.destination=<kafkaQueueName>
```

Or map your own method to complete the wiring:

```
app.<application>.spring.cloud.stream.bindings.<method>.destination=<kafkaQueueName>
```

2.4.3.3 Application custom properties

Custom application properties shall be draft as follow :

```
app.<application>.<property>=<value>
```

The RS core will provide only factory settings.

These settings will be updated for each SCDF chain. For example, from the RS core “ingestion”, we will build the SCDF chains : “ingestion S1 MTI”, “ingestion S1 SGS”, “ingestion S1 MPS”, “ingestion S1 EDRS-A”, “ingestion S2 SGS”, “ingestion S2 MPS”, “ingestion S2 MTI”, “ingestion S2 EDRS-A”, “ingestion S3 SGS”, …

Sample n°1 :
For a RS core ingestion, properties can be :

- app.ingestion-trigger.station.ipAddress=1.2.3.4
- app.ingestion-trigger.station.ipPort=1212
- app.ingestion-trigger.station.name=STATION_NAME
- app.ingestion-trigger.station.secret=mySecretName
- app.ingestion-trigger.station.login=myLogin
- app.ingestion-trigger.station.scrambling=NO
- app.ingestion-trigger.station.rootDirectoryPath=/a/b/c
- app.ingestion-trigger.station.startingDate=2022-01-10T12:00:00
- app.ingestion-filter.filter = [{start: '2022-01-04T12:00:00.000', stop: '2022-01-04T19:00:00.000’} ,{start: '2022-01-11T12:00:00.000', stop: '2022-01-11T19:00:00.000’} ,{start: '2022-01-18T12:00:00.000', stop: '2022-01-18T19:00:00.000’}]
- app.ingestion-worker.station.ipAddress=1.2.3.4
- app.ingestion-worker.station.ipPort=1212
- etc…

2.4.4 Services settings

Settings for Kafka service are stored in a specific directory.

/additional_resources/

For the time being, only Kafka settings can be updated thanks to the operator.

2.4.4.1 Service Kafka settings

One YAML file per queue setting located here : **/additional_resources/KafkaTopic**

The filename is free. Extension must be YAML. One file per configuration.

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaTopic
metadata:
  name: <my_topic_name>
  labels:
    strimzi.io/cluster: kafka-cluster
    app.kubernetes.io/instance: <application_name>
    app.kubernetes.io/managed-by: additional_resources
spec:
  partitions: <2>
  replicas: <4>
  config:
    retention.bytes: <89478485>
    retention.ms: <"-1">
    segment.bytes: <22369621>
    cleanup.policy: <delete>
    min.insync.replicas: <2>
```

🟩 To be replaced. Mandatory.

🟧 Default value. The change is optional.

2.4.4.2 Service ElasticSearch settings

The application is allowed to create its index directly.

2.4.4.3 Service MongoDB settings

The application is allowed to create its index directly.

3 SAMPLE

RS_CORE_INGESTION_4.5.6_2022-08-10.zip