ESA EO FRAMEWORK - BUREAU D'ETUDES

ICD for PRIP

Reference: CAP-TN-036-BE - Version 1.2

June 2026

Table of Contents

Table of figures

CHANGE LOG

Reason for change	Issue	Date
First Issue	1.0	20/01/2024
Corrections to address comments	1.1	02/06/2025
Integrate Feedbacks from LTA ICD and comments from ESA. Add OD PRIP section	1.2	17/06/2025

CHANGE RECORD

Reason for change	page(s)	Paragraph(s)
Document creation	All	All
Version 1.1	4	Add list of abbreviations
	5 & 7	Add missing links in tables
	20	Add note in STAC-PRIP-ITEM-REQ-0050
	21	Add column “Queryable” in properties table
	25	Replace “shall” by “should” in STAC-PRIP-ITEM-REQ-0085
	26	Update STAC-PRIP-ITEM-REQ-0101
	27	Add new Req : STAC-PRIP-API-REQ-0203
	29 & 31	Add note in STAC-PRIP-API-REQ-0305 & STAC-PRIP-API-REQ-0410
	30	Remove STAC-PRIP-API-REQ-0340 and add a recommendation instead, and rename STAC-PRIP-API-REQ-0345 into STAC-PRIP-API-REQ-0340
	37	Modify service URL
	Several	Add .zarr extension in asset URLs
Version 1.2	23/24	Add the following properties into the table: • created • mission • sat:orbit_cycle
	26	STAC-PRIP-ITEM-REQ-0085 to STAC-PRIP-ITEM-REC-0085 New requirement STAC-PRIP-ITEM-REQ-0060

	27	Add 2 asset's properties: file:size and file:checksum STAC-PRIP-ITEM-REQ-0101: remove mission from bucket name and add a note about special character
	28	Update version of product, view and sar extensions Add file info extension into the table
	29	STAC-PRIP-API-REQ-0305 updated (Multipart upload)
	31	Add STAC-PRIP-API-REC-0335
	43	Add requirements for OD PRIP
	57	Add requirements for Subscription and notifications mechanisms
		Fix of several typos

APPROVALS

Role	Name	Signature
Written by:	Bureau d'Etudes Team	

1 Introduction

1.1 List of abbreviations

Name	Definition
API	Application Programming Interface
CDSE	Copernicus Data Space Ecosystem
CSC	Copernicus Space Component
EOF	ESA Earth Observations Framework
EOPF	EO Processor Framework. The new EOPF format is based on Zarr standard.
JSON	JavaScript Object Notation
LTA	Long Term Archive
OD Processing	On-Demand Processing
OGC	Open Geospatial Consortium
PRIP	Production Interface delivery Point
Pub/Sub	Publish / Subscribe
STAC	Spatio Temporal Asset Catalog : The STAC specification is a common language to describe geospatial information, so it can more easily be worked with, indexed, and discovered.

1.2 Purpose and Scope

The purpose of this document is to specify an https RESTful API through which Sentinel (or Auxiliary) data products may be discovered and downloaded by authorised users from a Production Interface delivery Point (PRIP). PRIPs are intended as the standard pick-up points for systematically produced Sentinel data products generated by the production services within EOF-CSC.

This document extends the specifications contained [STAC_EOF_ICD] and defines a set of requirements to be made applicable in the frame of STAC adoption for the definition and operations of pick-up point of the Production Services [RD-ARC], i.e. the Production service Interface Point (PRIP).

The API defined in this document is meant to be compliant with the STAC specification version 1.1.0.

Compliance to the ICD can be considered as an API contract with a set of STAC API style calls, which have to be supported, that shall follow STAC API query conventions such as URL syntax, filter syntax, content accessing conventions, etc.

However, STAC is not suitable for Processing and Subscription operations. Those endpoints will then be defined through a more suitable standard, “OGC API – Processes v1.0”. For notifications management, the "OGC API Publish-Subscribe" concepts will be supported, despite of the fact that this standard is still a draft version.

This document is structured as follows:

1. **Introduction**, this chapter and the list of applicable and reference documents and resources,
2. **Objectives and needs**, an explanation of the objectives and how they are achieved,
3. **PRIP Interface description**, a detailed description and related requirements of the PRIP interface

4. **Annex A** provides explanations and links to the conformance test suite

1.3 Applicable Documents

ID	Description
STAC_EOF_ICD	STAC Requirements and Recommendation for EOF Services CAP-TN-030-BE_STAC Version 1.0
STAC_ITEM_SPEC	STAC Item Specification v1.1.0 https://github.com/radiantearth/stac-spec/tree/v1.1.0/item-spec
STAC_COLLECTION_SPEC	STAC Collection Specification v1.1.0 https://github.com/radiantearth/stac-spec/tree/v1.1.0/collection-spec
STAC_CATALOG_SPEC	STAC Catalog Specification V1.1.0 https://github.com/radiantearth/stac-spec/tree/v1.1.0/catalog-spec
STAC_API_CORE_SPEC	STAC API – Core Specification V1.0.0 https://github.com/radiantearth/stac-api-spec/tree/release/v1.0.0/core
STAC_API_FEATURE_SPEC	STAC API – Collections and Features Specification v1.0.0 https://github.com/radiantearth/stac-api-spec/tree/release/v1.0.0/ogcapi-features#stac-api---features
STAC_API_ITEM_SEARCH_SPEC	STAC API – Item Search v1.0.0 https://github.com/radiantearth/stac-api-spec/tree/release/v1.0.0/item-search
STAC_API_FILTER_EXTENSION	STAC API – Filter Extension Specification https://github.com/stac-api-extensions/filter
STAC_API_SORT_EXTENSION	STAC API – Sort Extension Specification https://github.com/stac-api-extensions/sort
STAC_API_FIELDS_EXTENSION	STAC API – Fields Extension Specification https://github.com/stac-api-extensions/fields
GEOJSON_SPEC	The GeoJSON Format – RFC7946 https://datatracker.ietf.org/doc/html/rfc7946
GEOFOOTPRINT_RECOM	Recommendations for GeoJSON Footprints Ref: CAP-TN-027-ESA-BE Version 1.0
JSON_SCHEMA	JSON Schema Specification JSON Schema - Specification [#section] (json-schema.org)
AUTHENTICATION_SERVICE_ICD	Central Authentication Service ICD

TBD

1.4 Reference Documents and Resources

ID	Description
RD-ARC	ESA EO Operations Framework (EOF) - CSC - Ground Segment Architecture Ref: ESA-EOPG-EOPGC-TN-7 Version 1.5
EOPF_SPECIFICATION	Earth Observation Product Format https://eopf.copernicus.eu/eopf-products-and-adfs/eopf-adf-format-def-product-tutorials/
CEOS_STAC_RECOM	CEOS EO collection and granule discovery best practices with STAC https://github.com/ceos-org/stac-collection-and-granule-discovery-best-practices
STAC_BEST_PRACTICES	STAC Best Practices https://github.com/radiantearth/stac-spec/blob/master/best-practices.md
STAC_EXTENSIONS	STAC extensions https://stac-extensions.github.io/
STAC_API_EXTENSIONS	STAC API Extensions https://stac-api-extensions.github.io/
STAC_ALTERNATE_ASSET_EXTENSION	Alternate Assets Extension Specification https://github.com/stac-extensions/alternate-assets
STAC_TIMESTAMP_EXTENSION	Timestamps Extension Specification https://github.com/stac-extensions/timestamps
STAC_EOPF_EXTENSION	EOPF STAC Extension https://github.com/CS-SI/eopf-stac-extension
STAC_STORAGE_EXTENSION	Storage Extension Specification https://github.com/stac-extensions/storage
STAC_PRODUCT_EXTENSION	Product Extension Specification https://github.com/stac-extensions/product
STAC_PROCESSING_EXTENSION	Processing Extension Specification https://github.com/stac-extensions/processing
STAC_FILE_INFO_EXTENSION	File Info Extension Specification https://github.com/stac-extensions/file
STAC_AUTHENTICATION_EXTENSION	Authentication Extension Specification https://github.com/stac-extensions/authentication

ID	Description
STAC_EO_EXTENSION	Electro Optical Extension Specification https://github.com/stac-extensions/eo
STAC_SATELLITE_EXTENSION	Satellite Extension Specification https://github.com/stac-extensions/sat
STAC_SCIENTIFIC_EXTENSION	Scientific Extension Specification https://github.com/stac-extensions/scientific
STAC_GRID_EXTENSION	Grid Extension Specification https://github.com/stac-extensions/grid
STAC_SAR_EXTENSION	SAR Extension Specification https://github.com/stac-extensions/sar
STAC_VIEW_EXTENSION	View Geometry Specification https://github.com/stac-extensions/view
S3_API	S3 API Reference https://docs.aws.amazon.com/AmazonS3/latest/API/Type_API_Reference.html
OPENID_CONNECT_SPEC	OpenID Connect Core 1.0 incorporating errata set 2 https://openid.net/specs/openid-connect-core-1_0.html
CQL2_SPEC	Common Query Language (CQL2) https://docs.ogc.org/is/21-065r2/21-065r2.html#cql2-json
FILTER_CLASS	Filter Conformance Class URI https://portal.ogc.org/files/96288#_requirements_class_filter
FEATURE_FILTER_CLASS	Feature Filter Conformance Class URI https://portal.ogc.org/files/96288#_requirements_class_features_filter
BASIC_CQL2_CLASS	Simple CQL Conformance Class URI http://www.opengis.net/spec/cql2/1.0/conf/basic-cql2
ITEM_SEARCH_FILTER_CLASS	Item Search Filter Conformance Class URI https://api.stacspec.org/v1.0.0-rc.3/item-search#filter
CQL2_JSON_CLASS	CQL2 JSON Conformance Class URI http://www.opengis.net/spec/cql2/1.0/conf/cql2-json
DATA_FLOW_CONFIG	Copernicus Ground Segment - Sentinels Data Flow Configuration Ref: ESA-EOPG-EOPGC-TN-58 Version 1.2
DATA_PORTFOLIO	ESA EO Operations Framework (EOF) – CSC – Copernicus Sentinel Data Portfolio Ref: ESA-EOPG-EOPGC-TN-12 Version 1.4

ID	Description
OGC_API_PROCESSES_STANDARD	OGC API - Processes - Part 1: Core Ref: 18-062r2 Version: 1.0.0 https://docs.ogc.org/is/18-062r2/18-062r2.html
OGC-API_Publish Subscribe_STANDARD	OGC API Publish-Subscribe Workflow - Part 1: Core https://docs.ogc.org/DRAFTS/25-030.html
PRIP_ODATA_ICD	Production Interface Delivery Point Specification Ref: ESA-EOPG-EOPGC-IF-3 version 2.0
LTA_ICD	Long Term Archive AIP Specification Ref: CAP-TN-037-BE Version 1.0-rc3

2 Objectives and needs

2.1 PRIP Description

PRIPs are pick-up points for newly published Sentinel data products, located inside the CSC Ground Segment, which allow clients to straightforwardly discover and retrieve available products through standard interfaces, in this case a STAC RESTful API. Clients of the PRIPs are intended to be bulk downloaders whose principal use case requires systematic (or at least regular) retrieval of newly published Sentinel data products. Examples of PRIP clients include the Copernicus Data Space Ecosystem (which in turn makes the products available to end users), Long Term Archives (LTAs), and possibly clients requiring data for Instrument and Product Performance investigations.

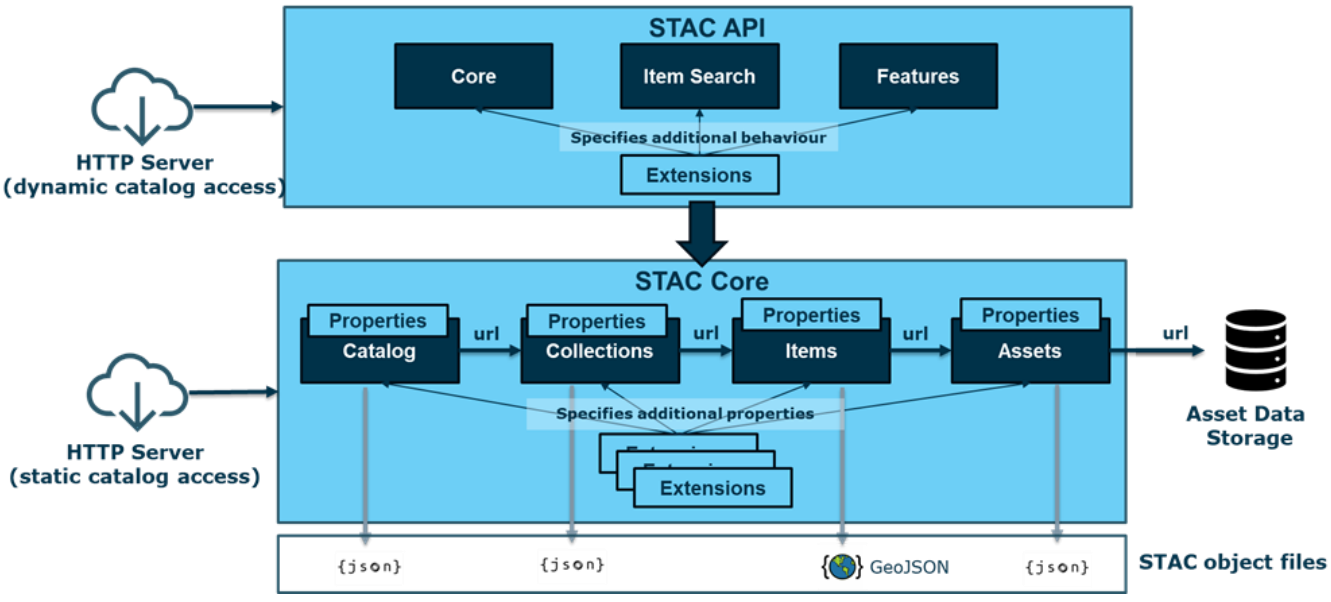
PRIPs are the first point of retrieval for newly published Sentinel data products and, as such, they are co-located with the production service. PRIPs make new products available in the agreed packaged unit, with any compression as described in each product unit definition and metadata; the products are only being published once the product package is fully available. The published products are maintained in a rolling archive covering a time span of e.g. 3 to 5 days, thereby allowing the recovery of data in case of delays, connectivity issues, etc.

In the typical use scenario, the API allows authorised users to discover, through a simple set of filters, the list of products matching specified criteria (e.g. since the last check) and to download those products.

2.2 STAC Description

STAC is an open specification that defines a common structure and an API for describing and cataloguing spatio-temporal assets.

The following figure highlights the main concepts of the STAC specification:



The STAC specification is composed of two main sub-parts:

- The STAC Core, that specifies the STAC Objects that constitute the STAC Data Model, and
- The STAC API, that specifies the set of operations that can be performed on the STAC Objects leveraging an HTTP API.

This document establishes requirements and recommendations for the implementation of STAC in EOF PRIP instances, independently of the specific Sentinel Mission served by each PRIP. Mission-specific requirements and tailoring are defined in an independent document, named "tailored ICD" in the following.

Therefore, it is expected that any PRIP instance involved in the applicable Sentinel missions complies with:

1. The requirements and recommendations defined in [**STAC_EOF_ICD**],
2. The requirements and recommendations contained in the present document

2.3 OGC API – Processes description

Reference to [LTA_ICD], chapter 2.3

2.4 Scope of the document

This document is applicable to the following functions that are provided by PRIP instances:

- Query PRIP Product Catalogue
- Download PRIP Products
- On-Demand Processing
- Subscription and Notification
- Export PRIP Catalogue (in a next version)

Compared to the legacy PRIP ICD based on OData [PRIP_ICD], the current document does not address the following PRIP functions:

- Production Metrics
- Retrieve PRIP Production Report (for end-to-end monitoring purposes)

This specification is assumed to be applicable in conjunction with the adoption of the new Sentinel product format, EOPF.

Each statement is formulated according to the following example:

> **<ID> – <Title>**
>
> Text of the statement.

The first line includes the following fields:

- For Query functionalities: <ID> Id of the requirement or recommendation in the form **STAC-PRIP-<CAT>-REQ-<XXXX> (for a requirement) or STAC-PRIP-<CAT>-REC-<XXXX> (for a recommendation)**
  - <CAT> is the related STAC category:
    - CORE: General requirement related to STAC Core
    - API: PRIP API
    - CAT: Catalog
    - COL: Collection
    - ITEM: Item
    - ASSET: Asset

  - <XXXX> is the four-digit ID of the statement.
- For OD processing functionalities: <ID> Id of the requirement or recommendation in the form **OD-PRIP-PRO-REQ-<XXXX> (for a requirement) or OD-PRIP-PRO-REC-<XXXX> (for a recommendation)**
- <Title> The title of the statement.

The following lines are the body (text) of the requirement or recommendation. For a requirement, we use the modal “shall” whereas a “recommendation” is expressed with the modal “should”.

3 PRIP Interface description

3.1 Architecture overview

The following diagram describes the high-level architecture that underpins the PRIP interface.

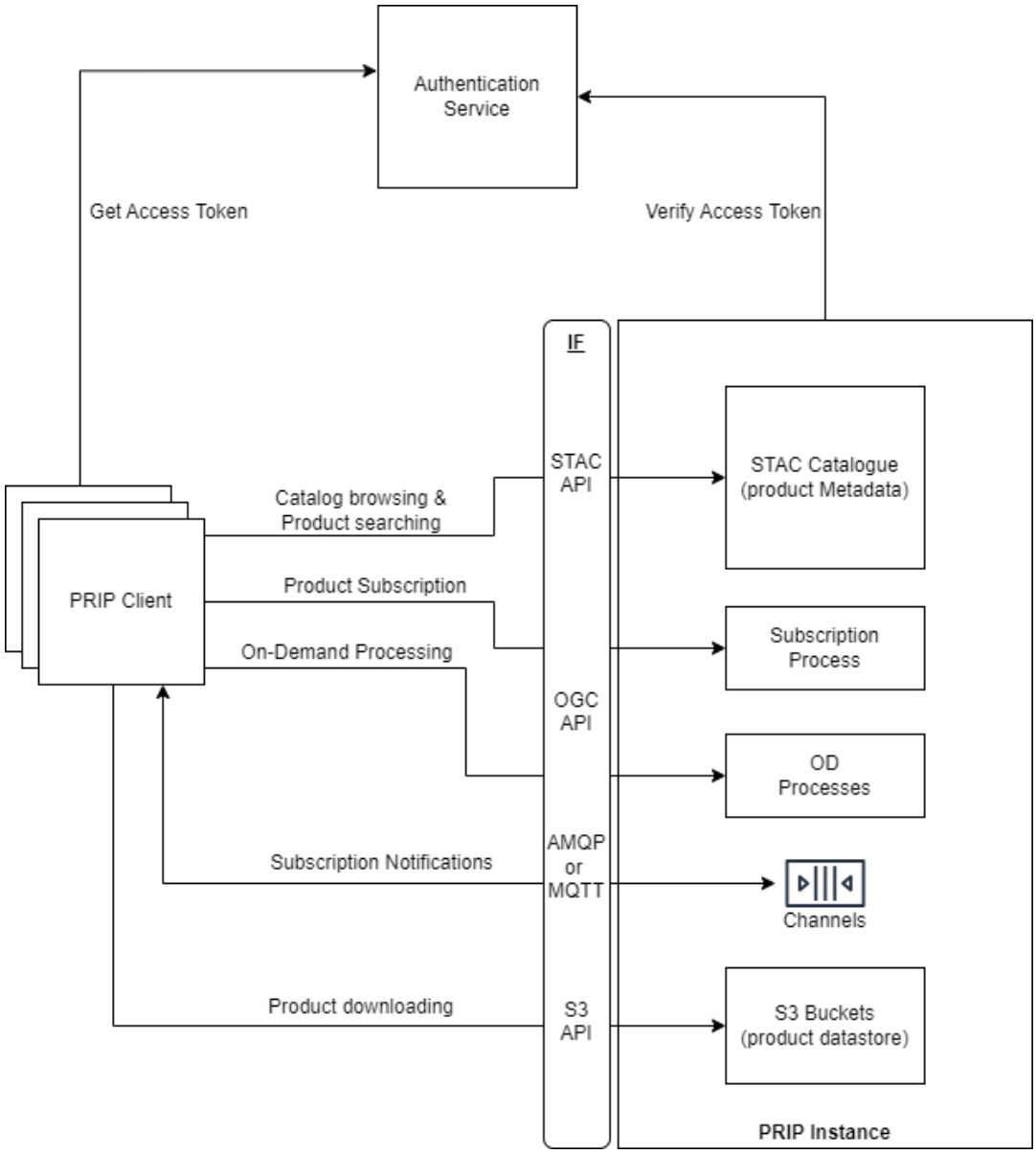


Figure 1: PRIP Interfaces

This architecture highlights the following components:

- The **PRIP Instance,** which embeds:
  - The STAC Catalogue, that exposes a standard STAC API for browsing and searching the collections and products managed by the instance and for retrieving their metadata.
  - OGC API - Processes for OD processing and Subscription
  - Pub/Sub channels for Notification sending

- A set of S3 Buckets, that store the managed products and expose the S3 API to allow the downloading of those products.

- The **PRIP Clients,** that use the various APIs (STAC Catalogue API, S3 API, OGC API) for searching, ordering and downloading products that match their area of interest.
- The **Authentication Service** that supports the centralized authentication service by providing an OIDC interface so that:
  - PRIP Clients retrieve an access token that shall be included in any request sent to the STAC Catalog API and OGC API
  - PRIP Instance validate the access tokens found in the incoming requests to ensure they are valid.

3.2 Nominal use scenario description

The nominal use scenario for discovery and download of products from a PRIP is shown in the figure below:

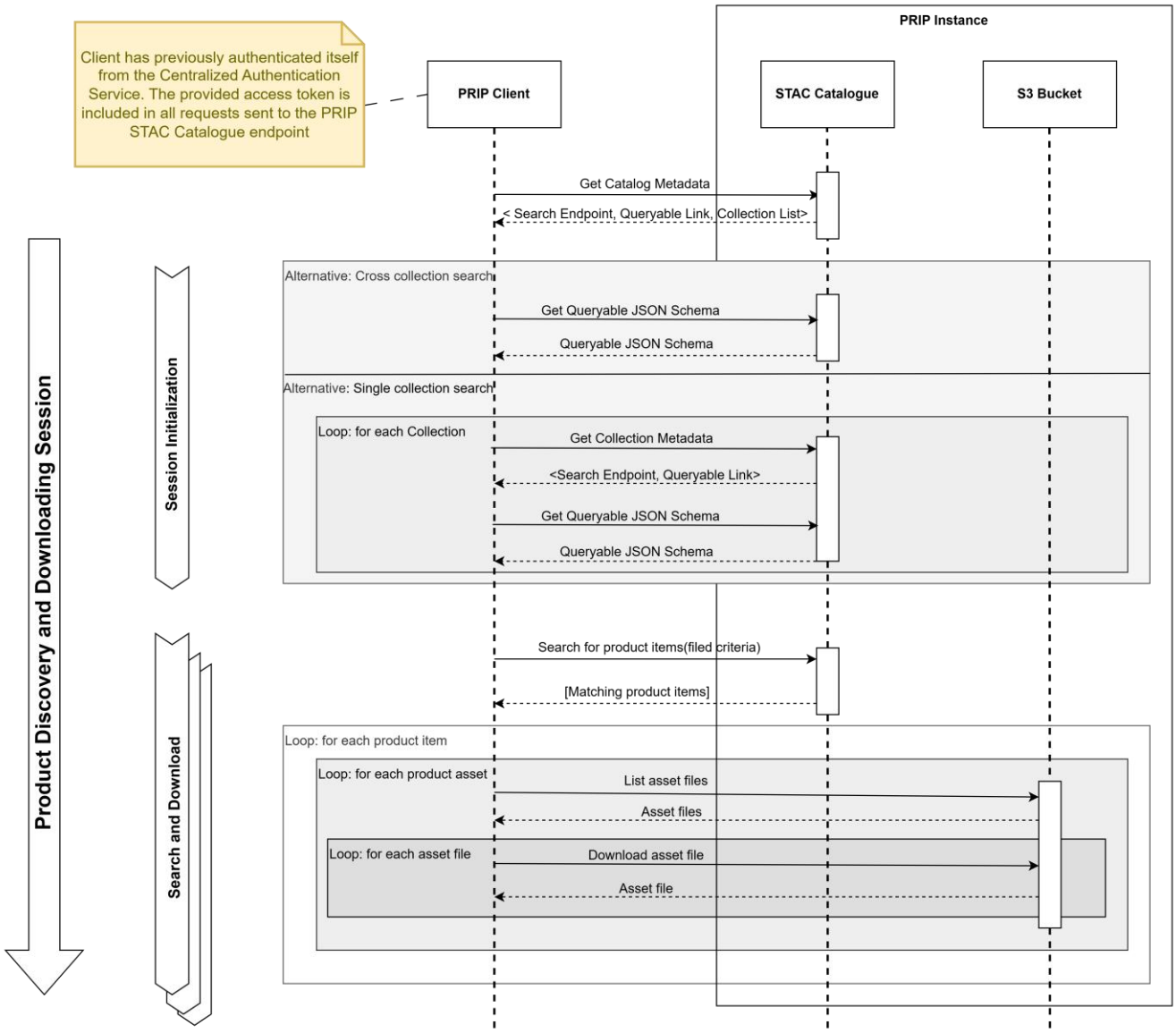


Figure 2: Sequence diagram for nominal scenario

It is mandatory that the PRIP Client retrieves from the Centralized Authentication Service an Access Token. This Access Token shall be included in all requests sent to the STAC Catalogue endpoint. The latter is responsible for validating this token and ensuring the client is authorized to access the requested information. For the sake of clarity and concision, these interactions are not shown in the diagram above.

First, during the session initialization, the PRIP Client shall retrieve from the STAC Catalogue root page the Catalogue metadata, which provide in particular:

- The HTTP endpoint to perform cross collection searching.
- The link to the Queryable JSON Schema that specifies the product metadata fields that can be provided as query parameters for future search requests.
- The list of links to the managed product collections.

From this point, the PRIP Client is able to perform two kinds of queries for searching products:

- Cross collection search queries.
- Single collection search queries.

For cross collection search queries, the PRIP Client shall retrieve the Queryable JSON Schema provided in the STAC Catalogue root page.

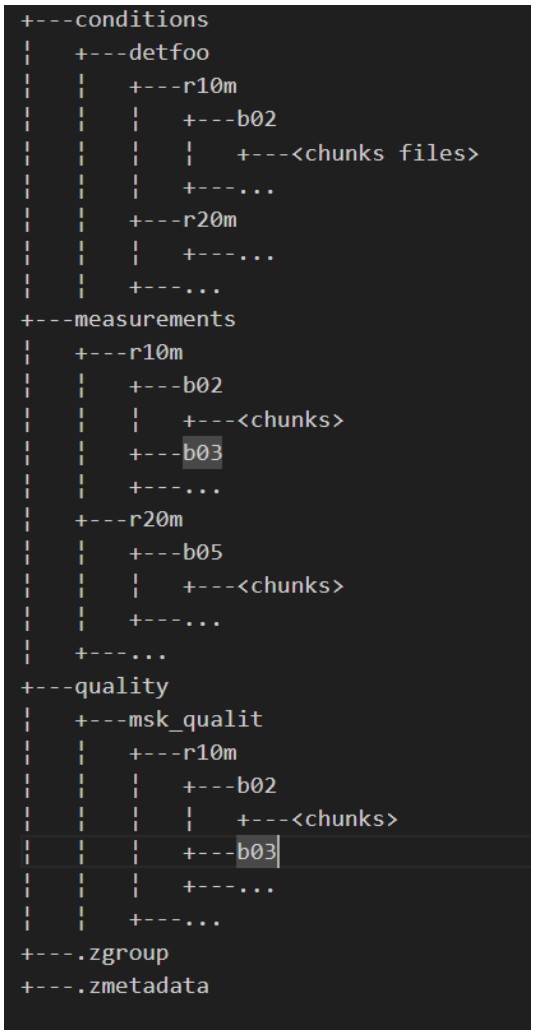
For single collection search queries, the PRIP Client shall first retrieve the related collection metadata. From those metadata, it shall extract the corresponding Queryable JSON schema and the search endpoint. This shall be performed for any collection that will be requested in the next steps.

Once the PRIP Client has retrieved the relevant Queryable JSON schema and search endpoint, it can perform product search queries, ensuring that provided metadata field criteria comply with the Queryable JSON Schema.

As a result of a search query, the PRIP Client retrieves a set of STAC Items. Each of them includes properties backed by STAC Core specification or STAC extensions and a list of STAC Assets. Each asset provides a link to the corresponding files stored in S3 buckets. From those STAC Assets, the PRIP Client is able to download the related files by querying S3 buckets. As with STAC Catalog access, all requests sent to the S3 bucket must be authenticated by leveraging the authentication mechanism provided by the object storage provider (not shown in the diagram above).

Note that the PRIP Client may repeat those searches and download requests as many times as needed for the ongoing session.

As part of the EOPF specification, each Product is formatted as a Zarr file structure. Here is a sample of such a structure:



At the root of the file structure, ***.zmetadata*** file identifies this folder as a Zarr content. Under the `measurements` folder, the various band data for each resolution can be found.

The Zarr format allows a client to download and exploit:

- Either the entire Zarr content,
- Or an individual group of measures, for example the data of a given band.

Thus, for the purpose of this ICD, Product Items returned in response to a Catalogue query include a STAC Asset for the downloading of the entire Zarr content and an Asset for each individual band or resolution. The detailed list of exposed assets is mission specific and shall be defined in the relevant EOPF product specification.

This implies that the Zarr file structure is stored as is in a S3 bucket, that is each file of the Zarr content is copied as a S3 object, keeping the relative path from the root folder as a S3 prefix. From the PRIP client point of view, retrieving an asset requires to:

- list all S3 objects sharing the prefix provided by the asset link, leveraging the ListObjectsV2 S3 operation
- download each object whose reference has been returned by the previous operation, leveraging the S3 GetObject operation. The PRIP client shall ensure to keep the file structure defined by the full Object key when storing an object on the local file system.

3.3 PRIP Data Model

The main feature implemented by PRIP is to allow remote EOF Services, such as LTA or CDSE, to query the PRIP Product Catalogue and download products referenced in the responses of those queries.

For this purpose, PRIP leverages a Data Model represented by the following figure:

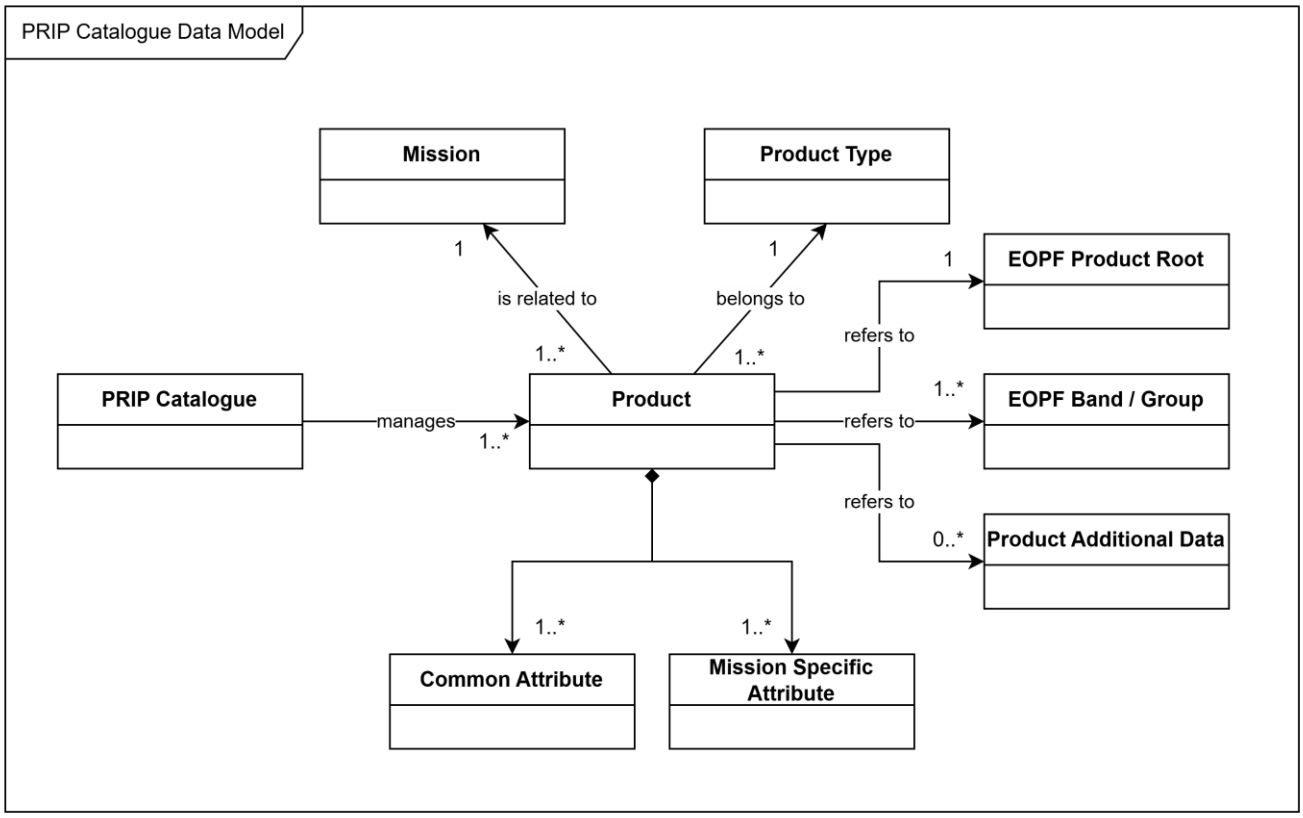


Figure 3: PRIP Catalogue Data Model

Each PRIP Catalogue instance manages a set of Products related to a given mission. Each Product belongs to a Product Type and is described by a set of common attributes and mission specific attributes. Each Product maintains:

- a reference to the full EOPF Product Zarr structure,
- a set of references to the EOPF product assets
- a set of references to production additional data.

Those references allow a PRIP client to download the related content.

Based on this high-level data model, the following table provides the relationship between each entity of the PRIP data Model and the STAC metamodel:

PRIP Data Model entities	Related STAC Metamodel element
PRIP Catalogue Instance	STAC Catalogue
Product Type	STAC Collection
Product	STAC Item
Product attributes	STAC Item properties
Reference to Product Files	STAC Asset

The following sections state the requirements applicable to any PRIP instance to properly map the PRIP Data Model to the STAC Core metamodel.

3.3.1 STAC Catalogue requirements

STAC-PRIP-CAT-REQ-0010 – Mapping PRIP Catalogue instance to STAC Catalogue

The PRIP instance shall implement and expose a STAC Catalogue for the purpose of querying the products managed by the PRIP.

The STAC Catalogue for Production Services will typically manage the production of a single satellite unit.

STAC-PRIP-CAT-REQ-0011 – Compliance with high level STAC Core requirements

The PRIP STAC Catalogue shall comply with the STAC Core requirements stated in [STAC_EOF_ICD].

Here is a sample of the returned JSON response when retrieving the Catalogue landing page.


```
{
  "type": "Catalog",
  "id": "s2a-prip",
  "title": "Sentinel 2A PRIP Catalogue",
  "description": "PRIP Catalogue for Sentinel 2A",
  "stac_version": "1.1.0",
  "conformsTo": [
    "http://www.opengis.net/spec/cql2/1.0/conf/cql2-json",
    "http://www.opengis.net/spec/cql2/1.0/conf/cql2-text",
    "https://api.stacspec.org/v1.0.0/item-search",
    "https://api.stacspec.org/v1.0.0/ogcapi-features",
    "http://www.opengis.net/spec/ogcapi-features-3/1.0/conf/features-filter",
    "https://api.stacspec.org/v1.0.0-rc.3/ogcapi-features/extensions/transaction",
    "https://api.stacspec.org/v1.0.0/item-search#sort",
    "https://api.stacspec.org/v1.0.0/collections",
    "https://api.stacspec.org/v1.0.0/item-search#fields",
    "https://api.stacspec.org/v1.0.0/item-search#query",
    "http://www.opengis.net/spec/ogcapi-features-3/1.0/conf/filter",
    "http://www.opengis.net/spec/cql2/1.0/conf/basic-cql2",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/oas30",
    "https://api.stacspec.org/v1.0.0-rc.2/item-search#filter",
    "https://api.stacspec.org/v1.0.0-rc.2/item-search#context",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/core",
    "https://api.stacspec.org/v1.0.0/core",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/geojson"
  ],
  "links": [
    {
      "rel": "self",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/"
    },
```

```
    {
      "rel": "root",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/"
    },
    {
      "rel": "data",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/collections"
    },
    {
      "rel": "conformance",
      "type": "application/json",
      "title": "STAC/OGC conformance classes implemented by this server",
      "href": "https://s2a-prip.copernicus.esa.int/conformance"
    },
    {
      "rel": "search",
      "type": "application/geo+json",
      "title": "STAC search",
      "href": "https://s2a-prip.copernicus.esa.int/search",
      "method": "GET"
    },
    {
      "rel": "search",
      "type": "application/geo+json",
      "title": "STAC search",
      "href": "https://s2a-prip.copernicus.esa.int/search",
      "method": "POST"
    },
    {
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/queryables",
      "type": "application/schema+json",
      "title": "Queryables",
      "href": "https://s2a-prip.copernicus.esa.int/queryables",
      "method": "GET"
    },
    {
      "rel": "child",
      "type": "application/json",
      "title": "Sentinel-2 Level-1C",
      "href": "https://https://s2a-prip.copernicus.esa.int/collections/sentinel-2-
level-1C"
    },
    {
      "rel": "child",
      "type": "application/json",
      "title": "Sentinel-2 Level-2A",
```

```
      "href": "https://https://s2a-prip.copernicus.esa.int/collections/sentinel-2-level-2A"
    },
    {
      "rel": "service-desc",
      "type": "application/vnd.oai.openapi+json;version=3.0",
      "title": "OpenAPI service description",
      "href": "https://s2a-prip.copernicus.esa.int/api"
    },
    {
      "rel": "service-doc",
      "type": "text/html",
      "title": "OpenAPI service documentation",
      "href": "https://s2a-prip.copernicus.esa.int/api.html"
    }
  ],
  "stac_extensions": []
}
```


3.3.2 STAC Collection requirements

STAC-PRIP-COL-REQ-0020 – Mapping PRIP Sentinel Missions to STAC Collections

The PRIP instance shall implement and expose a STAC Collection for each Product Type it manages.

STAC-PRIP-COL-REQ-0030 – PRIP STAC Collection mandatory properties

The PRIP instance shall ensure that the following STAC Collection properties are set as specified:

- `"id"`: product type identifier
- `"title"`: display name of the collection
- `"description"`: short description of the collection
- `"links"`: links to related STAC entities; it shall include at least the following relations:
  - `"root"`: the link to the root page, that is the root URL to reach the PRIP Catalogue instance
  - `"parent"`: same as root
  - `"self"`: the link to retrieve the current Collection definition
- `"items"`: the link to access to the STAC Items belonging to the Collection
- `"queryables"`: the link to the JSON Schema that defines all the criteria that are supported by the query function. This schema shall be defined by the mission specific configuration.
- `"item_assets"`: the list of common assets for each STAC Item belonging to this collection, and their description
- `"stac_version"`: the STAC Core specification version; shall be set to 1.1.0

- **"stac_extensions"**: the list of STAC Core extensions used to define the current Collection.

Here is an example of the expected JSON response when retrieving a STAC Collection managed by the PRIP instance:


```
{
  "id": "sentinel-2-l1c",
  "type": "Collection",
  "links": [
    {
      "rel": "items",
      "type": "application/geo+json",
      "href": "https://s2a-prip.copernicus.esa.int/collections/sentinel-2-l1c/items"
    },
    {
      "rel": "parent",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/"
    },
    {
      "rel": "root",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/"
    },
    {
      "rel": "self",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/collections/sentinel-2-l1c"
    },
    {
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/queryables",
      "href": "https://s2a-prip.copernicus.esa.int/collections/sentinel-2-
l1c/queryables",
      "type": "application/schema+json",
      "title": "Queryables"
    }
  ],
  "title": "Sentinel-2 Level-1C",
  "description": "The Level-1C product is composed of 110x110 km2 tiles (ortho-images
in UTM/WGS84 projection). Earth is subdivided on a predefined set of tiles, defined in
UTM/WGS84 projection and using a 100 km step. However, each tile has a surface of
110x110 km² in order to provide large overlap with the neighbouring. The Level-1C
product results from using a Digital Elevation Model (DEM) to project the image in
cartographic geometry. Per-pixel radiometric measurements are provided in Top Of
Atmosphere (TOA) reflectances along with the parameters to transform them into
radiances.",
  "item_assets": {
```

```
    "product": {
      "gsd": 10,
      "type": "application/vnd+zarr",
      "roles": [
        "data",
        "reflectance"
      ],
      "title": "Full EOPF Product",
      "auth:refs": [
        "s3"
      ]
    },
    "b02": {
      "gsd": 10,
      "name": "Band 2",
      "type": "application/vnd+zarr",
      "roles": [
        "data",
        "reflectance"
      ],
      "title": "Blue (band 2) - 10m",
      "auth:refs": [
        "s3"
      ]
    }
    // ... other asset descriptions
  },
  "auth:schemes": {
    "s3": {
      "type": "s3"
    },
    "oidc": {
      "type": "openIdConnect",
      "openIdConnectUrl": "https://identity.copernicus.eu/auth/realms/EOF/.well-
known/openid-configuration"
    }
  },
  "stac_version": "1.1.0",
  "stac_extensions": [
    "https://stac-extensions.github.io/item-assets/v1.0.0/schema.json",
    "https://stac-extensions.github.io/authentication/v1.1.0/schema.json",
    "https://stac-extensions.github.io/projection/v1.1.0/schema.json",
    "https://stac-extensions.github.io/product/v0.1.0/schema.json",
  ]
}
```

3.3.3 STAC Item and Asset requirements

The following requirements are applicable to any PRIP instance, whatever the related Sentinel mission.

STAC-PRIP-ITEM-REQ-0040 – STAC Items available in each collection

For each STAC collection (corresponding to a product type) it exposes, the PRIP instance shall publish and make available each product belonging to this collection. Each product shall be described as a distinct STAC Item.

STAC-PRIP-ITEM-REQ-0050 – STAC Items common properties

Any Items delivered by the PRIP instance shall at least include the properties listed in following table.

Note : all the properties related to time information shall have a microsecond precision.

Meaning of the table's columns:

- **Field Name**: Full name of the field
- **JSON path in Item**: JSON path expression of the field in the STAC Item
- **STAC Core or Extension**: Set to STAC Core when the field is defined by STAC Core specification or the Full Name of the STAC Extension that defines the related field
- **Provider**: 'EOPF' if the value is retrieved from the EOPF product metadata; 'PRIP' when the value is directly provided by the PRIP instance.

Field Name	JSON path in Item	STAC Core or Extension	Provider	Expected value	Queryable
Id	$.id	STAC Core	EOPF	Name of the corresponding product without the file extension (generally the name of the EOPF product).	yes
Item Type	$.type	STAC Core	EOPF	Shall be set to `"Feature"`	
Geo Footprint	$.geometry	STAC Core	EOPF	GeoJSON geometry that represents the product geo footprint. The inner `"type"` and `"coordinates"` attributes shall be set accordingly.	yes
Bounding Box	$.bbox	STAC Core	EOPF	The bounding box coordinates of the product.	
Origin Datetime	$.properties.eopf:origin_datetime	EOPF	PRIP	Timestamp of the availability of all CADU data on the CADIP/XBIP.	
Creation Date	$.properties.created	STAC Core	PRIP	Creation date and time of the corresponding STAC item into the PRIP instance Catalogue.	
Publication Date	$.properties.published	Timestamps	PRIP	Timestamp of the availability of the current Item in the PRIP instance Catalogue.	yes
Update Date	$.properties.updated	STAC Core	PRIP	If STAC Item properties have been updated, timestamp of the last update.	
Expiration Date	$.properties.expires	Timestamps	PRIP	Timestamp when the Item will no longer be available in the PRIP instance Catalog	

Field Name	JSON path in Item	STAC Core or Extension	Provider	Expected value	Queryable
Datetime	$.properties.datetime	STAC Core	EOPF	Timestamp of the satellite observation related to the product. In case of a time range acquisition, it shall be set to the middle of the start and end time of the acquisition, as per STAC-CORE-ITEM-REQ-0170.	yes
Start Datetime	$.properties.start_datetime	STAC Core	EOPF	Start datetime of the satellite observation related to the product. In case of a single time acquisition, it shall be set to the Datetime field, as per STAC-CORE-ITEM-REQ-0170.	yes
End Datetime	$.properties.end_datetime	STAC Core	EOPF	End datetime of the satellite observation related to the product. In case of a single time acquisition, it shall be set to the Datetime field, as per STAC-CORE-ITEM-REQ-0170.	yes
Product Type	$.properties.product:type	Product	EOPF	See allowed product types in [Erreur ! Source du renvoi introuvable.]	yes
Product Timeliness	$.properties.product:timeliness	Product	EOPF	Timeliness of the product.	
Product Timeliness Category	$.properties.product:timeliness_category	Product	EOPF	Timeliness category of the product.	
Production Type	$.properties.processing:lineage	Processing	EOPF	Information on the production type.	
Mission	$.properties.mission	Stac Core	EOPF	Name of the space program to which the constellation belongs (ex: copernicus)	
Constellation	$.properties.constellation	Stac Core	EOPF	Name of the constellation, e.g.: `sentinel-1 or sentinel-2`.	

Field Name	JSON path in Item	STAC Core or Extension	Provider	Expected value	Queryable
Platform[1]	$.properties.platform	STAC Core	EOPF	Platform involved in the data acquisition of the product. It shall be set according to STAC-CORE-GEN-REC-0020. It is mission specific and shall be defined in tailored ICDs. e.g: sentinel-1a	yes

Instruments	$.properties.instruments	Core	EOPF	List of instrument names, e.g.:
				`"instruments": [`
				`"olci"`
				`]`	

Orbit cycle	$.properties.sat:orbit_cycle	Satellite	EOPF	The number of repeat cycle done by the satellite at the time of the acquisition	
Absolute Orbit Number	$.properties.sat:absolute_orbit	Satellite	EOPF	Satellite absolute orbit number at the time of observation	yes
Relative Orbit Number	$.properties.sat:relative_orbit	Satellite	EOPF	Satellite relative orbit number at the time of observation	yes
Orbit State	$.properties.sat:orbit_state	Satellite	EOPF	Satellite orbit state at the time observation, may be `"ascending"` or `"descending"`	yes
Processing Level	$.properties.processing:level	Processing	EOPF	Depending on the Sentinel missions, possible values are: • `"L1"` • `"L1B"` • `"L1C"` • `"L2"`	

[1] It is the concatenation of the following current OData properties: platformShortName and platformSerialIdentifier

Field Name	JSON path in Item	STAC Core or Extension	Provider	Expected value	Queryable
				• `"L2A"`	
Processing Version	$.properties.processing:version	Processing	EOPF	Processing algorithm version, e.g: `"05.10"`	yes
Processing Date	$.properties.processing:datetime	Processing	EOPF	Processing date and time of the corresponding product.	
Processing Facility	$.properties.processing:facility	Processing	EOPF	The name of the PRIP instance that produced the product.	
Processing Software	$.properties.processing:software	Processing	EOPF	A dictionary with name/version for key/value describing one or more applications or libraries that were involved during the production of the data for provenance purposes.	
STAC Version	$.stac_version	Core	PRIP	Shall be set to `"1.1.0"`	
STAC Extensions	$.stac_extensions	Core	PRIP	Shall include the list of all the STAC extension schema URL used to describe the Item (see STAC-PRIP-CORE-REQ-300).	
Links	$.links	Core	PRIP	The links related to the current Item. See STAC-CORE-ITEM-REQ-0150 from **[STAC_EOF_ICD]**	
Collection	$.collection	Core	PRIP	Name of the parent collection (shall be identical to the Product Type attribute).	yes

STAC-PRIP-ITEM-REQ-0060 – STAC Items $.id field's uniqueness

The $.id field of an item shall be unique across all collections in the STAC catalog hosted by The PRIP instance, and implementations must prevent the creation of STAC items with the same ID in multiple collections.

STAC-PRIP-ITEM-REQ-0070 – Expected STAC Assets

Any items delivered by the PRIP instance shall include at least the following Assets:

- Asset related to the full EOPF Product
- Asset related to the individual or group of bands of the Product

STAC-PRIP-ITEM-REQ-0080 – Name of the STAC Asset related to the EOPF product

The name (JSON key) of the STAC Asset related to the full EOPF product package shall be `"product"`.

STAC-PRIP-ITEM-REC-0085 – Thumbnail asset

Each STAC item exposed by the PRIP instance should include an asset representing a thumbnail of the full EOPF product. The PRIP instance should ensure the download of the thumbnail file from the corresponding URL.

Note: This thumbnail is useful when browsing the PRIP instance STAC catalog through the STAC Browser.

Note 2: See STAC-CORE-ASSET-REQ-0300 for proper implementation of this recommendation.

STAC-PRIP-ITEM-REQ-0087 – Exposed assets related to sub-elements of the EOPF product

Each STAC Item exposed by the PRIP instance shall include and allow the download of all the assets identified in the [Erreur ! Source du renvoi introuvable.] for the corresponding STAC collection (a.k.a. product type).

STAC-PRIP-ITEM-REQ-0090 – STAC Asset properties

The STAC Assets included in Items published by PRIP Instance shall include the properties listed in the following table:

Field Name	JSON path in Item	STAC Core or Extension	Expected value
Download URL	$.assets.<asset_name>.href	STAC Core	S3 URL of the corresponding ZARR folder in the following form: `s3://<bucket>/<path>` The <path> is an S3 prefix allowing to list all the S3 objects matching this prefix.
Title	$.assets. <asset_name>.title	STAC Core	The title of the asset

Type	$.assets. <asset_name>.type	STAC Core	The media type be set to `"application/vnd+zarr"`
Roles	$.assets. <asset_name>.roles	STAC Core	The role list shall include following items: • `"data"` • `"metadata"`
Local Folder Name	$.assets. <asset_name>.file:local_path	File Info	Name of the full folder path, for example : `"<product_id>.zarr/measurements/r10m/b02/"`
File size	$.assets.<asset_name>.file: size	File Info	Size of the file: only relevant for asset targeting the full zarr product
File checksum	$.assets.<asset_name>.file: checksum	File Info	File checksum: only relevant for asset targeting the full zarr product
Authentication scheme	$.assets.<asset_name>.auth:refs	Authentication	Reference to the authentication scheme defined in the auth:schemes of the item properties

STAC-PRIP-ITEM-REQ-0100 – STAC Asset href

The STAC Asset href property is a S3 compliant URL. It shall have the following form: `s3://<bucket_name>/<product_path>`

The `<product_path>` shall have one of the following forms:

- `<product_id>.zarr/` to point to the Full EOPF Product content identified by <product_id>
- `<product_id>.zarr/<folder>/<subfolder>/` to point to the ZARR sub-folder with the relative path `<folder>/<subfolder>` (the number of levels of the subfolder hierarchy is not limited)

STAC-PRIP-ITEM-REQ-0101 – Bucket naming convention

The STAC Asset shall be stored within a bucket whose name matches the following form:

`prip-<provider>-<productType>`

with provider = the name of Service provider

Note about spacial character : when the name of a product type contains one or several '_' characters (forbidden in bucket names), they shall be replaced with '-' characters, or removed if it's a trailing character.

Example for a Zarr product with ID S02MSIL2A_20230629T120347_0000_A064_TC64.zarr, stored in bucket “prip-<provider>-sentinel-2-l2a” (note that “<provider>” must be replaced by the service provider name):

Asset Type	S3 url
Full EOPF Product	s3://prip-<provider>-sentinel-2-l2a/ S02MSIL2A_20230629T120347_0000_A064_TC64.zarr/
Band 02	s3://prip-<provider>-sentinel-2-l2a/ S02MSIL2A_20230629T120347_0000_A064_TC64.zarr/measurements/reflectance/r10m/b02/

3.3.4 STAC Extensions

STAC-PRIP-CORE-REQ-0102 – STAC Extensions

The PRIP instance shall implement the following STAC extensions and the related minimal version

Extension Name	Minimal version	Reference
Satellite	1.1.0	**[STAC_SATELLITE_EXTENSION]**
Product	1.0.0	**[STAC_PRODUCT_EXTENSION]**
Processing	1.2.0	**[STAC_PROCESSING_EXTENSION**STAC_PRODUCT_EXTENSION**]**
Scientific Citation	1.0.0	**[STAC_SCIENTIFIC_EXTENSION]**
Electro Optical	2.0.0	**[STAC_EO_EXTENSION]**
Grid	1.1.0	**[STAC_GRID_EXTENSION]**
View Geometry	1.1.0	**[STAC_VIEW_EXTENSION]**
SAR	1.3.0	**[STAC_SAR_EXTENSION]**
EOPF	1.2.0	**[STAC_EOPF_EXTENSION]**
Timestamps	1.1.0	**[STAC_TIMESTAMP_EXTENSION]**
Authentication	1.1.0	**[STAC_AUTHENTICATION_EXTENSION]**
File Info	2.1.0	[STAC_FILE_INFO_EXTENSION]

3.4 Search and download product API

This section formalizes the requirements applicable to PRIP Instances and PRIP Clients to implement PRIP Search and download products based on the principles described in §3.2.

3.4.1 PRIP instance requirements

This section defines the requirements applicable to any PRIP instance for allowing PRIP Clients to search and download products leveraging:

- OpenID Connect and JWT standards for authentication and authorization
- STAC API specification for searching capabilities
- AWS S3 API for downloading products

3.4.1.1 Search capability requirements

STAC-PRIP-API-REQ-0200 – STAC API requirements

The PRIP instance shall be compliant with all the requirements specified in section §4 of document [STAC_EOF_ICD]. This includes:

- General Requirements
- Simple search capabilities
- Advanced search capabilities

STAC-PRIP-API-REQ-0203 – STAC API limit parameter

The PRIP instance shall allow to configure the maximum number of objects (collections or items) returned by a STAC query. This limit shall not be less than 100.

STAC-PRIP-API-REQ-0205 – STAC API Extensions

The PRIP instance shall implement the following STAC API extensions and the related minimal version

Extension Name	Minimal version	Reference
Filter	1.0.0-rc4	[STAC_API_FILTER_EXTENSION]
Sort	1.0.0	[STAC_API_SORT_EXTENSION]
Field	1.0.0	[STAC_API_FIELDS_EXTENSION]

STAC-PRIP-API-REQ-0207 – Queryable fields

The PRIP instance shall allow clients to create filters based on the following fields:

- The common properties that are listed in STAC-PRIP-ITEM-REQ-0050 and marked as “queryable”
- Mission specific properties that are defined in the [Erreur ! Source du renvoi introuvable.] for the corresponding product type

Note: It implies that the queryable schemas managed by PRIP instance shall be set accordingly.

3.4.1.2 Authentication and authorization requirements

STAC-PRIP-API-REQ-0208 – Compliance to Central Authentication Service

The PRIP instance shall comply with [AUTHENTICATION_SERVICE_ICD] to ensure the authentication of the client.

STAC-PRIP-API-REQ-0210 – Access control to the STAC Catalogue

The PRIP instance shall implement an authorization mechanism that allows to grant (or deny) individual clients at different levels:

- global access to the PRIP Catalogue;
- access to each individual PRIP Collection.

STAC-PRIP-API-REQ-0220 – Global access control semantics

Global access control shall be applied priorly to collection access control.

Global access is required to:

- retrieve the metadata of the Catalogue object
- retrieve the metadata of the underlying Collections, the metadata of the included Items, and to download the related asset files.

If global access is denied for a Client, any STAC API requests it sends shall result in a 403 HTTP Status code response and the corresponding error message. In this case, access to the underlying collections shall be denied too.

STAC-PRIP-API-REQ-0230 – Collection access control semantics

Collection access is required to:

- retrieve the metadata of the Collection object;
- retrieve the metadata of the Items included in this Collection and download the related asset files.

STAC-PRIP-API-REQ-0240 – Role based access control

The authorization mechanism shall rely on the “Client roles” specified through the corresponding JWT claims. The JWT is provided as a Bearer token in the incoming request.

STAC-PRIP-API-REQ-0250 – Configurable access control

The authorization mechanism implemented by the PRIP instance shall be configurable.

STAC-PRIP-API-REQ-0260 – Zero downtime access control configuration change

It shall be possible to update the access control configuration with no downtime.

3.4.1.3 Download capability requirements

STAC-PRIP-API-REQ-0300 – S3 data storage

The PRIP instance shall store products it manages in S3 buckets.

STAC-PRIP-API-REQ-0305 – S3 object checksum

The PRIP instance shall use the checksum feature of the underlying S3 infrastructure to associate a checksum with each object stored in S3 buckets. In case the Multipart upload method is used, it shall:

- choose the “full object checksum” type,
- specify a suitable algorithm for checksum calculation (CRC64NVME, CRC32 or CRC32C)

Note: This checksum is used by the PRIP client to check the file integrity of the downloaded objects.

STAC-PRIP-API-REQ-0310 – Keeping the Zarr file Structure

The PRIP instance shall preserve the Zarr file structure when storing a product in a S3 bucket. This implies that:

- Each individual file of the Zarr product is stored as a S3 object;
- The file relative path from the root folder of the Zarr product is the prefix of the corresponding S3 object;
- `"/"` is the delimiter for any listing operations.

STAC-PRIP-API-REQ-0320 – S3 API endpoint for downloading products

The PRIP instance shall expose a S3 API endpoint to allow PRIP clients to download the products it manages.

STAC-PRIP-API-REQ-0330 – S3 API endpoint authentication and authorization

The PRIP instance shall authenticate any requests to the S3 API endpoint it manages. It shall authorize download requests consistently with respect to STAC-PRIP-API-REQ-210.

Note: as per STAC-PRIP-ITEM-REQ-0101, each asset file is backed by an S3 bucket related to the collection/product type.

STAC-PRIP-API-REC-0335 – download quotas exceeded

The PRIP instance should implement a quota management on downloaded assets by each individual client

In case an LTA client has consumed its allowed quotas, an HTTP status code 429 (“Too Many Requests) is sent in the response with the necessary information, including the datetime when the quotas will be reset.

Note: For each client, it should be possible to configure the size of data per unit of time to download (for example, n Tbytes per month). Quota adjustment should be managed at operations level.

3.4.2 PRIP Client requirements

STAC-PRIP-API-REQ-0340 – STAC compliance

PRIP Clients shall compls with STAC Core 1.1.0 and STAC API 1.0.

STAC-PRIP-API-REQ-0350 – PRIP Catalogue search

When searching for products managed by a PRIP instance, the PRIP Client shall apply:

- [STAC_API_FILTER_EXTENSION] for cross-collection search
- [STAC_API_FEATURE_SPEC] for single collection search
- The compliance classes provided in the STAC Catalogue root page in the `conformsTo` property.

STAC-PRIP-API-REQ-0360 – PRIP Catalogue search

When searching for products managed by a PRIP instance, the PRIP Client shall apply:

- [STAC_API_FILTER_EXTENSION] for expressing criteria on the resulting product Items
- [STAC_API_FIELDS_EXTENSION] for defining the fields that shall be included in or excluded from the resulting STAC items
- [STAC_API_SORT_EXTENSION] for defining the sorting logic on the resulting STAC items

STAC-PRIP-API-REQ-0365 – Compliance to Queryables schemas

When searching for products managed by a PRIP instance, the PRIP Client shall verify that the fields provided as search criteria comply with:

- the Queryable JSON schema provided in the STAC Catalogue root page when performing cross-collection search;
- the Queryable JSON schema provided in a Collection page when performing a search within that collection.

STAC-PRIP-API-REQ-0370 – PRIP Client authentication for STAC browsing

When searching for products managed by a PRIP instance, the PRIP Client shall include in any HTTP request it sends to the PRIP instance an access token previously gathered from the centralized Authentication Service. This token shall be provided in the request an Authorization Bearer token HTTP Header, of the following form:

- `Authorization: Bearer <token>`

STAC-PRIP-API-REQ-0380 – Keeping an access token up to date

The PRIP Client is responsible for keeping its access token up to date implementing the refresh token OIDC flow.

STAC-PRIP-API-REQ-0390 – S3 Compliance

The PRIP Client shall comply with [S3_API] when downloading products from a PRIP instance S3 API endpoint.

STAC-PRIP-API-REQ-0391 – PRIP Client authentication for S3 accesses

The PRIP Client shall implement the authentication mechanism related to the PRIP instance S3 API endpoint.

Note: This mechanism is specific to each PRIP instance, as it relies on the S3 implementation provided by the Cloud provider hosting the PRIP instance. Therefore, it shall be specified in tailored ICDs.

STAC-PRIP-API-REQ-0400 – Asset download

The PRIP Client shall use the S3 URL provided in any PRIP Asset as a prefix . In order to retrieve the content from this prefix, the PRIP Client shall:

- List all the objects matching this prefix, using the ListObjectsV2 operation.
- Download all the objects from the previous list operation, using the GetObject operation
- Store locally each object as a file, preserving the folder structure reflected in the object key and the `"/"` delimiter

STAC-PRIP-API-REQ-0410 – object checksum verification

The PRIP client shall use the checksum feature of the S3 API to verify the integrity of each downloaded object.

3.4.3 Use case examples

This section provides some examples of interactions between a PRIP Client and a PRIP Instance. For the examples, the following CLI executables are used:

- curl to interact with the STAC Catalogue endpoint
- AWS CLI to interact with the S3 endpoint

In the following examples, it is assumed that:

- The PRIP Client has already retrieved from the Centralized Authentication Service an access token whose content is <TOKEN>
- The STAC Catalogue endpoint is `https://s2a-prip.copernicus.esa.int/`
- aws-cli is configured with the proper S3 endpoint URL, access key and secret key to access the PRIP S3 endpoint.

3.4.3.1 Retrieving the STAC Catalogue root page

The following command allows to retrieve the Catalogue metadata, needed for the next operations:

```
curl -H "Authorization: Bearer <TOKEN>" -H "Accept: application/json" https://s2a-prip.copernicus.esa.int/
```

The response has the following form:


```
{
  "type": "Catalog",
  "id": "s2a-prip",
  "title": "Sentinel 2A PRIP Catalogue",
  "description": "PRIP Catalogue for Sentinel 2A",
  "stac_version": "1.1.0",
  "conformsTo": [
    "http://www.opengis.net/spec/cql2/1.0/conf/cql2-json",
    "http://www.opengis.net/spec/cql2/1.0/conf/cql2-text",
    "https://api.stacspec.org/v1.0.0/item-search",
    "https://api.stacspec.org/v1.0.0/ogcapi-features",
    "http://www.opengis.net/spec/ogcapi-features-3/1.0/conf/features-filter",
    "https://api.stacspec.org/v1.0.0-rc.3/ogcapi-features/extensions/transaction",
    "https://api.stacspec.org/v1.0.0/item-search#sort",
    "https://api.stacspec.org/v1.0.0/collections",
    "https://api.stacspec.org/v1.0.0/item-search#fields",
    "https://api.stacspec.org/v1.0.0/item-search#query",
    "http://www.opengis.net/spec/ogcapi-features-3/1.0/conf/filter",
    "http://www.opengis.net/spec/cql2/1.0/conf/basic-cql2",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/oas30",
    "https://api.stacspec.org/v1.0.0-rc.2/item-search#filter",
    "https://api.stacspec.org/v1.0.0-rc.2/item-search#context",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/core",
    "https://api.stacspec.org/v1.0.0/core",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/geojson"
  ],
  "links": [
    {
      "rel": "self",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/"
    },
    {
      "rel": "root",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/"
    },
    {
      "rel": "data",
      "type": "application/json",
      "href": "https://s2a-prip.copernicus.esa.int/collections"
    },
    {
      "rel": "conformance",
      "type": "application/json",
      "title": "STAC/OGC conformance classes implemented by this server",
      "href": "https://s2a-prip.copernicus.esa.int/conformance"
```

```
    },
    {
      "rel": "search",
      "type": "application/geo+json",
      "title": "STAC search",
      "href": "https://s2a-prip.copernicus.esa.int/search",
      "method": "GET"
    },
    {
      "rel": "search",
      "type": "application/geo+json",
      "title": "STAC search",
      "href": "https://s2a-prip.copernicus.esa.int/search",
      "method": "POST"
    },
    {
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/queryables",
      "type": "application/schema+json",
      "title": "Queryables",
      "href": "https://s2a-prip.copernicus.esa.int/queryables",
      "method": "GET"
    },
    {
      "rel": "child",
      "type": "application/json",
      "title": "Sentinel-2 Level-1C",
      "href": "https://https://s2a-prip.copernicus.esa.int/collections/sentinel-2-
level-1C"
    },
    {
      "rel": "child",
      "type": "application/json",
      "title": "Sentinel-2 Level-2A",
      "href": "https://https://s2a-prip.copernicus.esa.int/collections/sentinel-2-
level-2A"
    },
    {
      "rel": "service-desc",
      "type": "application/vnd.oai.openapi+json;version=3.0",
      "title": "OpenAPI service description",
      "href": "https://s2a-prip.copernicus.esa.int/api"
    },
    {
      "rel": "service-doc",
      "type": "text/html",
      "title": "OpenAPI service documentation",
      "href": "https://s2a-prip.copernicus.esa.int/api.html"
    }
```

```
  ],
  "stac_extensions": []
}
```

From this response, the PRIP Client can extract:

- The conformance classes found in the **"conformsTo"** property
- The cross-collection search endpoint URL defined by the **"rel": "search"** link, that is **https://s2a-prip.copernicus.esa.int/search**
- The cross-collection Queryable JSON schema URL defined by the **"rel": "http://www.opengis.net/def/rel/ogc/1.0/queryables"** link, that is **https://s2a-prip.copernicus.esa.int/queryables**
- The collection endpoint URL defined by the **"rel": "data"** link, to retrieve the list of collections managed by the PRIP Instance, that is **https://s2a-prip.copernicus.esa.int/collections**

3.4.3.2 Retrieving the Collections list

The following command retrieves the list of collections managed by the PRIP Instance and the related metadata needed for the next operations:

```
curl -H "Authorization: Bearer <TOKEN>" -H "Accept: application/json" https://s2a-prip.copernicus.esa.int/collections
```

The response has the following form:

```
{
  "collections": [
    {
      "id": "sentinel-2-l1c",
      "type": "Collection",
      "links": [
        {
          "rel": "items",
          "type": "application/geo+json",
          "href": "https://s2a-prip.copernicus.esa.int/collections/sentinel-2-l1c/items"
        },
        {
          "rel": "parent",
          "type": "application/json",
          "href": "https://s2a-prip.copernicus.esa.int/"
        },
        {
          "rel": "root",
          "type": "application/json",
          "href": "https://s2a-prip.copernicus.esa.int/"
        },
        {
          "rel": "self",
          "type": "application/json",
```

```
          "href": "https://s2a-prip.copernicus.esa.int/collections/sentinel-2-l1c"
        },
        {
          "rel": "http://www.opengis.net/def/rel/ogc/1.0/queryables",
          "href": "https://s2a-prip.copernicus.esa.int/collections/sentinel-2-
l1c/queryables",
          "type": "application/schema+json",
          "title": "Queryables"
        }
      ],
      "title": "Sentinel-2 Level-1C",
      "description": "The Level-1C product is composed of 110x110 km2 tiles (ortho-
images in UTM/WGS84 projection). Earth is subdivided on a predefined set of tiles,
defined in UTM/WGS84 projection and using a 100 km step. However, each tile has a
surface of 110x110 km² in order to provide large overlap with the neighbouring. The
Level-1C product results from using a Digital Elevation Model (DEM) to project the
image in cartographic geometry. Per-pixel radiometric measurements are provided in Top
Of Atmosphere (TOA) reflectances along with the parameters to transform them into
radiances.",
      "item_assets": {
        …… // list of assets
      },
      "auth:schemes": {
        "s3": {
          "type": "s3"
        },
        "oidc": {
          "type": "openIdConnect",
          "openIdConnectUrl": "https://identity.copernicus.eu/auth/realms/EOF/.well-
known/openid-configuration"
        }
      },
      "stac_version": "1.1.0",
      "stac_extensions": [
        "https://stac-extensions.github.io/item-assets/v1.0.0/schema.json",
        "https://stac-extensions.github.io/authentication/v1.1.0/schema.json",
        "https://stac-extensions.github.io/projection/v1.1.0/schema.json",
        "https://stac-extensions.github.io/product/v0.1.0/schema.json",
      ]
    },
    {
      "id": "sentinel-2-l2a",
      "type": "Collection",
      // ... other collection properties
    }
  ]
}
```

From the collection list, the PRIP Client shall extract for each collection:

- The collection search endpoint URL defined by the `"rel": "items"` link, that is `https://s2a-prip.copernicus.esa.int/collections/sentinel-2-l1c/items` for the first collection
- The cross-collection Queryable JSON schema URL defined by the `"rel": "http://www.opengis.net/def/rel/ogc/1.0/queryables"` link, that is `https://s2a-prip.copernicus.esa.int/sentinel-2-l1c/queryables` for the first collection

3.4.3.3 Searching for Items within all the collections

The following command allows to search for all items whose publication date is after 1st January 2024 and request that the results be sorted according to their publication date. The results are returned according to a page size of 100 items (limit query parameter).

```
curl -X POST -d @query.json  -H "Authorization: Bearer <TOKEN>" -H "Content-Type: application/json" -H "Accept: application/geo+json" https://s2a-prip.copernicus.esa.int /search?limit=100
```

The query.json file has the following content:

```
{
  "filter-lang": "cql2-json",
  "filter": {
    "op": "and",
    "args": [
      {
        "op" : ">=",
        "args": [ { "property": "published" }, "2024-01-01:T00:00:00Z" ]
      },
      {
        "op" : "=",
        "args": [ { "property": "collection" }, "sentinel-2-l1c" ]
      }
    ]
  },
  "sortby": [
      {
          "field": "published",
          "direction": "asc"
      }
  ]
}
```

The response has the following form:

```
{
    "type": "FeatureCollection",
    "numberMatched": 12734,
    "numberReturned": 100,
    "features": [
        {
```

```
            "id": "S2B_MSIL1C_20240604T101559_N0510_R065_T44XNR_20240604T111607",
            "bbox": [
                80.99872,
                80.920299,
                87.993274,
                81.956919
            ],
            "geometry": {
                "type": "Polygon",
                "coordinates": [
                    [
                        [
                            80.9987196457332,
                            81.9569194040658
                        ],
                        [
                            87.9932736707368,
                            81.8974237140705
                        ],
                        [
                            87.2429668849209,
                            80.920298611386
                        ],
                        [
                            80.9988581739129,
                            80.9732636725462
                        ],
                        [
                            80.9987196457332,
                            81.9569194040658
                        ]
                    ]
                ]
            },
            "type": "Feature",
            "links": [
                // links
            ],
            "auth:schemes": {
                "s3": {
                    "type": "s3"
                }
            },
            "assets": {
                Product:, // description of asset 1
               B01:…, // description of asset 2
                ….,
            },
```

```
            "properties": {
                // item properties
            },
            "stac_extensions": [
                "https://stac-extensions.github.io/eo/v1.1.0/schema.json",
                "https://stac-extensions.github.io/raster/v1.1.0/schema.json",
                "https://stac-extensions.github.io/view/v1.0.0/schema.json",
                "https://stac-extensions.github.io/file/v2.1.0/schema.json",
                "https://stac-extensions.github.io/processing/v1.2.0/schema.json",
                "https://stac-extensions.github.io/product/v0.1.0/schema.json",
                "https://stac-extensions.github.io/storage/v1.0.0/schema.json",
                "https://stac-extensions.github.io/timestamps/v1.1.0/schema.json",
                "https://stac-extensions.github.io/authentication/v1.1.0/schema.json"
            ],
            "stac_version": "1.1.0"
        },
        // other item
    ]
}
```

From this response, the PRIP Client can identify from the asset list of each Item, the assets matching the content it wants to download. Each asset associated with an item is defined with a specific key. For example, the asset representing the full Zarr product is defined with the “Product:” key and its download’s URL targets a .zarr file:

s3://s2b_products/S2B_MSIL1C_20240604T101559_N0510_R065_T44XNR_20240604T111607.zarr

As another example, the asset corresponding to the product’s band2 is defined with the “B02:” key and its URL is:

s3://s2b_products/S2B_MSIL1C_20240604T101559_N0510_R065_T44XNR_20240604T111607.zarr/measuremens/r10m/b02/

From the previous result, the PRIP Client can read the total number of items that match its query from the **"numberMatched"** property. It may retrieve the results of any given page by specifying the page number as a query parameter. The following example demonstrates how to retrieve page number 5 considering 100 items per page:

```
curl -X POST -d @query.json  -H "Authorization: Bearer <TOKEN>" -H "Content-Type:
application/json" -H "Accept: application/geo+json"
https://pgstac.demo.cloudferro.com/search?limit=100&page=5
```

3.4.3.4 Downloading Zarr content

The following command allows to download the ZARR content of an asset whose href property is **s3://s2b_products/S2B_MSIL1C_20240604T101559_N0510_R065_T44XNR_20240604T111607.zarr**

```
aws s3 cp --recursive
s3://s2b_products/S2B_MSIL1C_20240604T101559_N0510_R065_T44XNR_20240604T111607.zarr/ .
```

Note: It is assumed that the AWS CLI is configured with the expected S3 API endpoint and authentication credentials.

The corresponding ZARR file structure is stored in the current working directory.

3.5 On-Demand Processing API

The On-Demand Processing API allows a client to request the on-the-fly processing of a parent (input) product into a higher level (output) product. More specifically, this is achieved by creating a ProductionOrder entity which specifies the product to be used as input, the workflow to apply, and the options if any for customising the workflow. In the context of the On-Demand Processing, a workflow corresponds to a specific version of a product processor.

There are two basic scenarios in which OD processing may be applied. The first is that a product that has been previously produced and catalogued has been evicted from the data access service and has to be recreated. The second is that a higher level processing may be applied to a product available in the catalogue.

The nominal scenario can be described in the following steps:

- Step 1 - Workflow Query: allows to get the relevant Workflows to apply on a given input product. This step is described in §3.5.2.
- Step 2 - ProductionOrder submission: allows to create a ProductionOrder towards the OD PRIP Service. This step is described in §3.5.3.
- Step 3 - ProductionOrder Processing: following receipt of a ProductionOrder, the OD PRIP Service generates a request for a product Order to retrieve the input product and sends to the LTA Service. The interfaces between the OD PRIP and LTA are fully covered in the LTA ICD [LTA_ICD] and hence are not described in detail in this document.
- Step 4 - Output Product Retrieval: once the ProductionOrder is completed, the output product can be identified and can be downloaded, either through a notification sent by the OD PRIP to the client, or through poll requests from the client. This mechanism is described in §3.5.3.

3.5.1 Conformance with OGC API Processes

The OD PRIP API shall be compliant with OGC API Processes v1.0 standard. The requirements described in [LTA_ICD] (chapter 2.3) are related to general requirements for conformance with this OGC API standard and describe common operations to list available processes, describe a process, execute a process, get a job status and gather a job result. They are valid for OD PRIP too.

The OD PRIP service shall define various processes to implement OD processing functionalities.

OD-PRIP-PRO-REQ-010 – supported Processes

The OD PRIP service shall support and define the following OGC processes:

- **workflowQuery** to get available Workflows
- **productionOrder** to request the execution of a Workflow for output product generation

Note: Each process is fully described in a dedicated section further in the document.

3.5.2 WorkflowQuery Process

Following the initial identification of the input product to which on-demand processing is to be applied, the PRIP Client may query the OD PRIP Service for relevant Workflows to apply, if this is not already known. The query response provides information on the identified Workflows, including the unique identifier, name, textual description and input and output products types.

This operation is illustrated in the sequence diagram below.

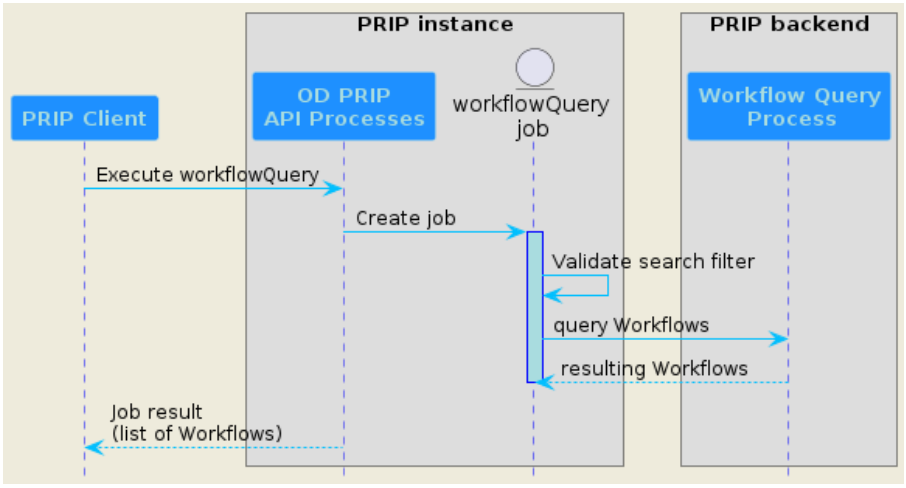


Figure 4: Sequence diagram of the synchronous workflowQuery process

The following requirements relates to the process description and characteristics, the expected inputs parameters, and the outputs provided in the result in case of a successful execution.

3.5.2.1 Process definition

OD-PRIP-PRO-REQ-020 – workflowQuery Process Definition

The OD PRIP service shall define and implement a specific process called "**workflowQuery**" to retrieve the list of available Workflows with optional search filtering. The workflowQuery process shall allow execution in synchronous mode and will return results as value. It shall be described with the following properties:

- id: value set to "workflowQuery".
- title: value set to "list all processing Workflows available to the user"
- description: value set to an understandable description of what the process performs

3.5.2.2 Process inputs/outputs

OD-PRIP-PRO-REQ-030 – workflowQuery Process inputs

The workflowQuery process shall support the following optional inputs:

- **filter**: Search filter on Workflow attributes (e.g. name, inputProductType, ..)
- **workflowOptions**: flag to include workflows options in the response

The following table details the list of inputs.

Input name	Input description	Input type	Occurrences
filter	Search filter on one Workflow attribute among the following ones: name or inputProductType. The filter shall be conform to the CQL2 JSON standard.	object	0 to 1
workflowOptions	If true, the Workflows options will be included in the response. Default value : false	boolean	0 to 1

OD-PRIP-PRO-REQ-040 – workflowQuery Process outputs

When successful, the workflowQuery process shall return a **list of available Workflows**, returned as in-line values (outputTransmission set to “value”) in a JSON response document. Each Workflow object has the following attributes (mandatory attributes in bold):

- **id**: Local unique identifier for the Workflow instance within the OD PRIP service (UUID format)
- **name**: Short name of the workflow
- **description**: Textual description of the workflow, including details of the processor version and configuration applicable
- **inputProductType**: Type of the Input Product
- **outputProductType**: Type of the Output Product
- **workflowVersion**: Version number applicable to the workflow
- workflowOptions: client-selectable WorkflowOptions. For each option, the following attributes are specified:
  - **name**: The name of the option
  - **description**: Textual description of the option.
  - **type**: The type of the option
  - default: The default value of the option, if there is one
  - **value**: Array representing all possible values for the option.

The response JSON document is retrieved through a job results query once it has reached a *successful* status (reference to [LTA_ICD] / OGC-LTA-API-REQ-070).

The following table summarizes the list of outputs.

Ouput name	Input description	Input type	Occurrences
workflowList	List of available Workflows corresponding to the optional search criteria defined in the "filter" input parameter. Each workflow is an object with various attributes (listed in OD-PRIP-PRO-REQ-040)	object	0 to N

3.5.2.3 Example of process definition

The following illustrates a definition of the workflowQuery process:

```
{
  "id": "workflowQuery",
  "title": "OD PRIP Workflows retrieval",
  "description": "This synchronous process sends back a list of supported Workflows for input
product types",
  "version": "1.0.0",
  "jobControlOptions": [
    "sync-execute"
  ],
  "outputTransmission": [
    "value"
```

```
  ],
  "inputs": {
    "filter": {
      "title": "Filter to search Workflows",
      "description": "Filter in CQL2 JSON standard, to search on Worflows attributes",
      "minOccurs": 0,
      "maxOccurs": 1,
      "schema": {
        "$ref": "https://github.com/opengeospatial/ogcapi-
features/blob/master/cql2/standard/schema/cql2.json"
      }
    },
    "workflowOptions": {
      "title": "Flag for WorkflowOptions display",
      "description": "Used to return WorkflowOptions",
      "minOccurs": 0,
      "maxOccurs": 1,
      "schema": {
        "type": "boolean"
      }
    }
  },
  "outputs": {
    "workflowList": {
      "title": "List of available Workflows",
      "description": "List of Workflows meeting search filter (if defined)",
      "schema": {
        "type": "array",
        "minItems": 0,
        "maxItems": N,
        "items": {
          "$ref": "#/$defs/workflow"
        },
        "$defs": {
          "workflow": {
            "type": "object",
            "required": [
"id","name","description","InputProductType","OutputProductType","workflowVersion" ],
            "properties": {
              "id": {
                "type": "string",
                "format": "uuid",
                "description": "Unique ID of the Workflow"
              },
              "name": {
                "type": "string",
                "description": "Workflow short name"
              },
```

```
            "description": {
              "type": "string",
              "description": "Workflow description"
            },
            "inputProductType": {
              "type": "string",
              "description": "Main product type"
            },
            "outputProductType": {
              "type": "string",
              "description": "Output product type"
            },
            "workflowVersion": {
              "type": "string",
              "description": "Workflow version"
            },
            "workflowOptions": {
              "type": "array",
              "minItems": 0,
              "maxItems": M,
              "description": "List of Workflow options",
              "items": {
                "$ref": "#/$defs/options"
              },
              "$defs": {
                "options": {
                  "type": "object",
                  "required": [ "name","description","type","value" ],
                  "properties": {
                    "name": {
                      "type": "string",
                      "description": "Option name"
                    },
                    "type": {
                      "type": "string",
                      "description": "Option type"
                    },
                    "description": {
                      "type": "string",
                      "description": "Option description"
                    },
                    "default": {
                      "type": "string",
                      "description": "The default value of the option"
                    },
                    "value": {
                      "type": "array",
                      "minItems": 1,
```

```
                        "maxItems": K,
                        "description": "all possible values of the option",
                        "items": {
                          "type": "string"
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "response": "document",
    "links": [
      {
        "href": "https://s2a-prip.copernicus.esa.int/odp/workflowQuery/execution",
        "rel": "http://www.opengis.net/def/rel/ogc/1.0/execute",
        "title": "Execute endpoint"
      }
    ]
}
```


3.5.2.4 Process execution

OD-PRIP-PRO-REQ-050 – workflowQuery Process execution

The workflowQuery process shall complete with a "*successful*" status if the "*filter*" input parameter is valid, even if none Workflows are returned back (empty list of Workflows). Otherwise, the process shall return with a "*failed*" status with an explicit error message.

As the process is synchronous, the results are directly sent back to the user in case of a successful execution.

3.5.3 ProductionOrder Process

Having selected the input product and the Workflow to be triggered plus applicable options, the Client submits a request to create a ProductionOrder towards the OD PRIP Service.

This operation is illustrated in the sequence diagram below.

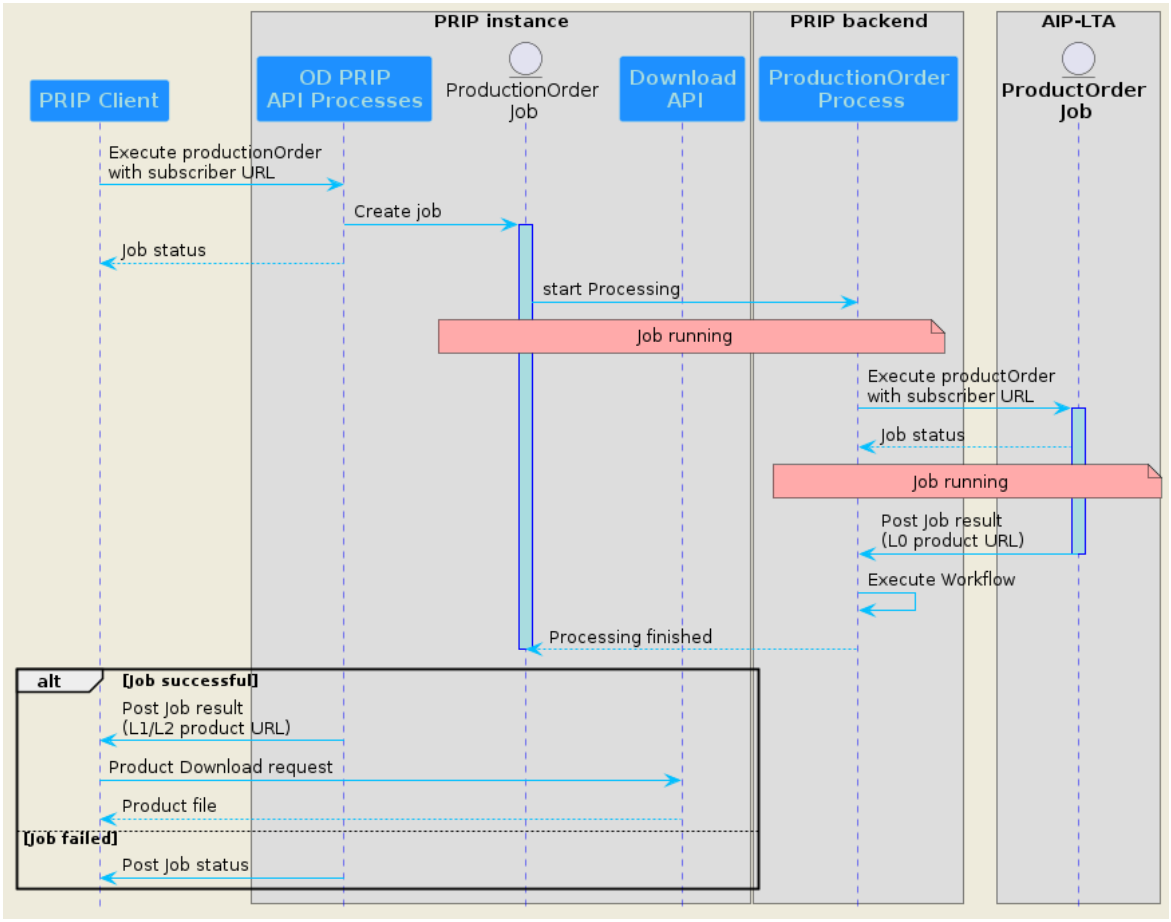


Figure 5: Sequence diagram of the asynchronous productionOrder process. Its execution consumes the AIP endpoint (LTA service) if the input product type is a L0 level one.

The following requirements relates to the process description and characteristics, the expected inputs parameters, and the outputs provided in the result in case of successful execution.

3.5.3.1 Process definition

OD-PRIP-PRO-REQ-060 – productionOrder Process Definition

The OD PRIP service shall define and implement a specific process called "**productionOrder**" to trigger a specific Workflow to execute to generate an output product from a given input product. The productionOrder process shall allow execution in asynchronous mode and will return results as value. It also shall be described with the following properties:

- id: value set to "productionOrder".
- title: value set to "Trigger a processing workflow to generate an output product from a given input product"
- description: value set to an understandable description of what the process performs

3.5.3.2 Process inputs/outputs

OD-PRIP-PRO-REQ-070 – productionOrder Process inputs

The productionOrder process shall support the following mandatory inputs:

- **inputProductReference**: Complex type used to unambiguously identify the input product
- **workflowId**: local unique identifier for the Workflow instance within the OD PRIP which is applicable to the productionOrder
- **workflowName**: Short name of the workflow

The productionOrder process shall support the following optional inputs:

- priority: Priority of the ProductionOrder. It is an integer from 1-100, where 100 is the highest priority.
- WorkflowOptions: Selection of applicable options from the Workflow (in the form "name:value")

The following table details the list of inputs.

Input name	Input description	Input type	Occurrences
inputProductReference	Either the product identifier ("id" STAC field) or the sensing range period (dates interval).	string or array of string	1
workflowId	Local unique identifier of the Workflow instance	string format uuid	1
workflowName	Short name of the workflow	string	1
priority	Priority of the ProductionOrder (value between 1 and 100)	integer	0 to 1
workflowOptions	Selection of applicable options from the Workflow	array of string	0 to 1

OD-PRIP-PRO-REQ-080 – productionOrder Process outputs

Once the associated job is successful, the workflowQuery process shall return a **list of available Workflows** as in-line values (outputTransmission set to “value”) in a JSON response document. Each Workflow object has the following attributes (mandatory attributes in bold):

- **productId**: Unique identifier of the output product generated by the Workflow (UUID).
- **productName**: name of the output product.
- **productionOrderId:** Local unique identifier for the ProductionOrder instance within the OD PRIP.
- evictionDate: Date when the Product related to the order will be removed from the OD PRIP.
- orderOutputSize: Actual size in bytes (B) of the output Product composing the Order.

The response JSON document is retrieved through a successful job "results query" (reference to [LTA_ICD] / OGC-LTA-API-REQ-070), or through a HTTP callback if a valid subscriber URI has been passed in the execution request.

Note 1: the "productionOrderId" attribute is a duplicate of the "jobID" generated by the OD PRIP service (related to OGC API - Processes standard).

Note 2: other fields specified in previous version of this ICD, like "status", "statusMessage", "submissionDate" and "CompletedDate" are not specified here because they are directly managed by OGC API - Processes standard. Their equivalent names are respectively "status", "message", "created" and "finished".

The following table summarizes the list of outputs.

Ouput name	Input description	Input type	Occurrences
productId	Identifier of the output product generated by the Workflow processor	string format uuid	1
productName	name of the output product	string	1
productionOrderId	Identical to Job ID generated by OD PRIP service	string format uuid	1
evictionDate	Date when the Product related to the order will be removed from the OD PRIP	string format date	0 to 1
orderOutputSize	Actual size in bytes (B) of the output Product	number	0 to 1

3.5.3.3 Example of process definition

The following illustrates a definition of the productionOrder process:

```
{
  "id": "productionOrder",
  "title": "Product generation with a given Workflow",
  "description": "This asynchronous process executes the required Workflow and generates a higher level product from an input parent product",
  "version": "1.0.0",
  "jobControlOptions": [
    "async-execute"
  ],
  "outputTransmission": [
    "value"
  ],
  "inputs": {
    "inputProductReference": {
        "oneOf": [
        {
          "title": "Product identifier",
          "description": "Input Product identifier",
          "minOccurs": 1,
          "maxOccurs": 1,
          "schema": {
            "type": "string"
          }
        },
        {
          "title": "Sensing range period",
          "description": "Interval of dates with start date and and date",
          "minOccurs": 1,
```

```
        "maxOccurs": 1,
        "schema": {
            "type": "array",
            "minItems": 2,
            "maxItems": 2,
            "items": {
              "type": "string",
              "format": "date-time"
          }
        }
      }
    ]
  },
  "workflowId": {
    "title": "Workflow identifier",
    "description": "Local unique identifier for the Workflow instance to execute",
    "minOccurs": 1,
    "maxOccurs": 1,
    "schema": {
      "type": "string",
      "format": "uuid"
    }
  },
  "workflowName": {
    "title": "Workflow name",
    "description": "Short name of the workflow to execute",
    "minOccurs": 1,
    "maxOccurs": 1,
    "schema": {
      "type": "string"
    }
  },
  "priority": {
    "title": "Order priority",
    "description": "Priority of the ProductionOrder",
    "minOccurs": 0,
    "maxOccurs": 1,
    "schema": {
      "type": "integer",
      "minimum": 1,
      "maximum": 100
    }
  },
  "workflowOptions": {
    "title": "Workflow options",
    "description": "Selection of applicable options from the Workflow",
    "schema": {
      "type": "array",
```

```
      "items": {
        "$ref": "#/$defs/options"
      },
      "$defs": {
        "options": {
          "type": "object",
          "required": [ "name","value" ],
          "properties": {
            "name": {
              "type": "string",
              "description": "Option name"
            },
            "type": {
              "type": "string",
              "description": "Option value"
            }
          }
        }
      }
    }
  }
},
"outputs": {
  "productId": {
    "schema": {
      "type": "string",
      "format": "uuid"
    }
  },
  "productName": {
    "schema": {
      "type": "string"
    }
  },
  "productionOrderId": {
    "schema": {
      "type": "string",
      "format": "uuid"
    }
  },
  "evictionDate": {
    "schema": {
      "type": "string",
      "format": "date-time"
    }
  },
  "orderOutputSize": {
    "schema": {
```

```
        "type": "number"
      }
    }
  },
  "response": "document",
  "links": [
    {
      "href": "https://s2a-prip.copernicus.esa.int/odp/productionOrder/execution",
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/execute",
      "title": "Execute endpoint"
    }
  ]
}
```


3.5.3.4 Process execution

OD-PRIP-PRO-REQ-090 – productionOrder Process execution

After the submission of a new productionOrder, a new job is created ("*accepted* " status). Before the workflow can be executed, several conditions shall be checked:

- The *workflowId* input parameter corresponds to a valid Workflow.
- The *workflowId* input parameter is consistant with the *InputProductReference* input parameter
- User quotas are not exceeded (number of concurrent orders and maximum number of authorized orders in a defined period)

If all those conditions are met, the job shall launch the requested Workflow, and switches to a "*successful*" status as soon as the output product has been generated.

Otherwise, the job terminates with a "*failed*" status. The content of that response shall be based upon the OpenAPI 3.0 schema exception.yaml. The type of the exception shall correspond to the reason of the failure, e.g. InvalidParameterValue for invalid input value.

3.5.4 Use case examples

3.5.4.1 WorkflowQuery example

The following example is a workflowQuery execute request's payload, with a CQL2 filter's criteria based on inputProductType field.

https://s2a-prip.copernicus.esa.int/odp/workflowQuery/execution


```
{
  "inputs": {
    "filterParam": {
       "filter-lang": "cql2-json",
       "filter": {
          "op": "like",
          "args": [
             { "property": "inputProductType" },
```

```
              "%S2MSI0%"
            ]
        }
    }
    "workflowOptions": true
  },
  "response": "document"
}
```

And a successful job result:

```
{
  "workflowList": [
    {
      "id": "UUID-1",
      "name": "Sentinel-2-L0-S2MSI1C",
      "description": "Product processing from Sentinel-2 L0 to L1C. Processor V2.3.6",
      "inputProductType": "S2MSI0",
      "outputProductType": "S2MSI1C",
      "workflowVersion": "2.1",
      "workflowOptions": [
        {
          "name": "ProductFormat",
          "type": "String",
          "description": "Product Format option for the production output",
          "value": [
            "SAFE",
            "ZARR"
          ]
        }
      ]
    },
    {
      "id": "UUID-2",
      "name": "Sentinel-2-L0-S2MSI2A",
      "description": "Product processing from Sentinel-2 L0 to L2A. Processor V2.4.6",
      "inputProductType": "S2MSI0",
      "outputProductType": "S2MSI2A",
      "workflowVersion": "2.2",
      "workflowOptions": [
        {
          "name": "ProductFormat",
          "type": "String",
          "description": "Product Format option for the production output",
          "value": [
            "SAFE",
            "ZARR"
          ]
        }
```

```
      ]
    }
  ]
}
```

3.5.4.2 ProductionOrder example

The following example is a productionOrder execute request's payload, with a specified Workflow option and a URL for notification purpose.

https://s2a-prip.copernicus.esa.int/odp/productionOrder/execution

```
{
   "inputs": {
    "inputProductReference": "S02MSIL0__20251111T182840_0001_A123_T000",
    "workflowId": "UUID-1",
    "workflowName": "Sentinel-2-L0-S2MSI1C",
    "priority": 50,
    "workflowOptions": [
      {
        "name": "ProductFormat",
        "value": "zarr"
      }
    ]
  },
  "response": "document",
  "subscriber": {
    "successUri": "https://prip_client_url"
  }
}
```

And a successful job result:

```
{
  "productId": "UUID-1",
  "productName": "S2A_MSIL1C_20240604T101559_N0510_R065_T44XNR_20240604T111607",
  "productionOrderId": "JOB-UUID",
  "evictionDate": "2026-07-01:T00:00:00Z",
  "orderOuputSize": 734003200
}
```

3.6 Product Subscription API

The Subscription function allows the request for automated notifications of future products published into the PRIP STAC Catalogue, according to a set of filter parameters supplied within the subscription request. Once a notification has been received about a new product availability, the client can request the download of the product.

Like for OD Processing, the PRIP service shall expose a REST API compliant with OGC Processes – API v1.0. In this respect, it shall define various processes to meet the Subscription functionalities, each process being available for PRIP client execution.

OGC-PRIP-PRO-REQ-100 – supported Processes for subscription

The PRIP instance shall support and define the following processes for subscription management:

- **subscriptionCreate** to create a new subscription for new products notifications
- **subscriptionCancel** to cancel an active subscription

Note: Each process is fully described in the next chapters.

The first paragraph relates to subscription management, and the second one relates to the sending of notifications to users with active subscrition(s).

Note: Generic requirements related to common OGC Processes – API endpoints remain valid for Subscription functionalities (see reference document [LTA_ICD] and requirements OGC-LTA-API-REQ-020 to OGC-LTA-API-REQ-090).

3.6.1 Subscription creation

The “Subscription Create” operation allows to create a new subscription to receive notifications for new products from a given collection. This is illustrated in the sequence diagram below.

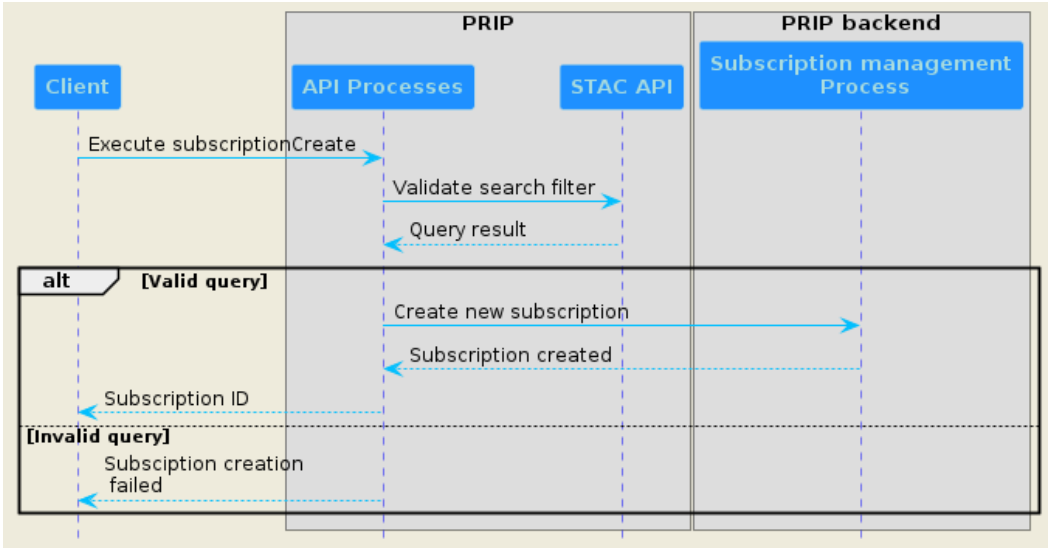


Figure 6: Sequence diagram of “subscriptionCreate” synchronous process

This operation is described through a specific Process implemented in the PRIP instance, called subscriptionCreate.

3.6.1.1 Process definition

The following requirements relates to the process description and characteristics, the expected inputs parameters, and the outputs provided in the job results in case of successful execution.

OGC-PRIP-PRO-REQ-110 – subscriptionCreate Process Definition

The PRIP instance shall define and implement a specific process called "**subscriptionCreate**" to support the creation of a new subscription in order to be notified on new products events. The subscriptionCreate process shall allow execution in synchronous mode and will return a unique subscription ID, to which the client can refer in subsequent operations. In case of failure caused by invalid query or any other reason, the response shall have a HTTP error code that corresponds to the reason of the failure. The content of that response shall be based upon the OpenAPI 3.0 schema exception.yaml. The type of the exception shall correspond to the reason of the failure, e.g. InvalidParameterValue for invalid input value.

The process shall be defined with the following properties:

- id: value set to "subscriptionCreate".
- title: value set to "PRIP new subscription creation"
- description: value set to an understandable description of what the process performs

3.6.1.2 Process inputs/outputs

OGC-PRIP-PRO-REQ-120 – subscriptionCreate Process inputs

The subscriptionCreate process shall support the following mandatory inputs:

- **filterParam**: filter expressed in CQL2 JSON to allow the PRIP instance to run a STAC query with filter extension. This query shall be used to filter on products for future notifications.
- **collection**: The collection for which the event is monitored. shall be a valid STAC collection.

The subscriptionCreate process shall support the following optional inputs:

- **bbox**: Bounding box for which the STAC query shall search products. The spatial operator implemented by default shall be "*s_intersects*". If a spatial operator is present in "filterParam", the *bbox* input parameter is ignored.

The following table details the list of inputs.

Input name	Input description	Input type	Occurrences
filterParam	Represents filter expressed in a CQL2 JSON language	object	1
collection	Collection for which the event is monitored; shall be constrained by an enum corresponding to supported STAC collections.	string	1
bbox	Bounding box for which the STAC query shall search products. Ignored if filterParam already contains a spatial operator.	bounding box	0 to 1

OGC-PRIP-PRO-REQ-130 – subscriptionCreate Process outputs

The subscriptionCreate process shall support the following outputs returned as in-line values (outputTransmission set to “value”) in a JSON document response:

- **subscriptionId**: a unique ID of the created subscription. This ID shall be used by the client to cancel the subscription, and to consume corresponding notification messages.

As the process is synchronous, the response is directly returned after the execution query.

The following table details the list of outputs parameters.

Output name	Input description	Output type	Occurrences
subscriptionId	Unique ID of the created subscription	string format uuid	1

3.6.1.3 Example of process definition

The following JSON content illustrates a definition of the subscriptionCreate process.

```
{
  "id": "subscriptionCreate",
  "title": "PRIP new subscription creation",
  "description": "This synchronous process creates a new subscription to monitor the publication of
products into a STAC collection. A notification is sent in case of matching with the filterParam
specified.",
  "version": "1.0.0",
  "jobControlOptions": [
    "sync-execute"
  ],
  "outputTransmission": [
    "value"
  ],
  "inputs": {
    "filterParam": {
      "title": "cql2-json filter",
      "description": "Filter for STAC query in CQL2 JSON",
      "minOccurs": 1,
      "maxOccurs": 1,
      "schema": {
        "$ref": "https://github.com/opengeospatial/ogcapi-
features/blob/master/cql2/standard/schema/cql2.json"
      }
    },
    "collection": {
      "title": "STAC Collection",
      "description": "Collection in which to monitor",
      "minOccurs": 1,
      "maxOccurs": 1,
      "schema": {
```

```
        "type": "string",
        "enum": [
          "sentinel2-l1c",
          "sentinel2-l2a"
        ]
      }
    },
    "bbox": {
      "title": "Bounding Box of STAC query",
      "description": " Bounding Box for which the query searches products",
      "minOccurs": 0,
      "maxOccurs": 1,
      "schema": {
        "allOf": [
          {
            "format": "ogc-bbox"
          },
          {
            "$ref":
https://schemas.opengis.net/ogcapi/processes/part1/1.0/openapi/schemas/bbox.yaml"
          }
        ]
      }
    }
  },
  "outputs": {
    "subscriptionId": {
      "title": "Unique subscription ID",
      "description": "Identifier of the created subscription",
      "schema": {
        "type": "string",
        "format": "uuid"
        }
     }
  },
  "response": "raw",
  "links": [
    {
      "href": "https://s2a-prip.copernicus.esa.int/processes/subscriptionCreate/execution",
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/execute",
      "title": "Execute endpoint"
    }
  ]
}
```

3.6.2 Subscription cancelation

The “subscription Cancel” operation allows to delete an existing subscription. This is illustrated in the sequence diagram below.

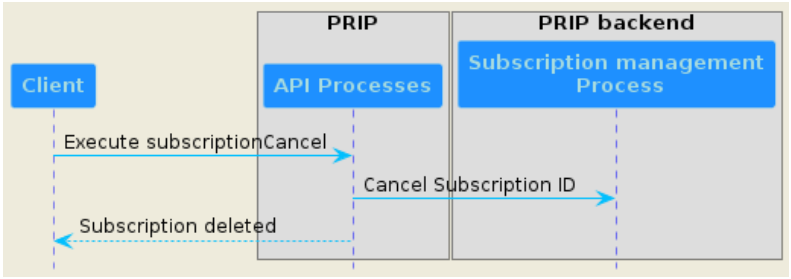


Figure 7: Sequence diagram of Subscription cancel process

This operation is described through a specific Process implemented in The AIP instance, called subscriptionCancel.

3.6.2.1 Process definition

The following requirements relates to the process description and characteristics, the expected inputs parameters, and the outputs provided in the job results in case of successful execution.

OGC-PRIP-PRO-REQ-140 – subscriptionCancel Process Definition

The PRIP instance shall define and implement a specific process called “**subscriptionCancel**” to support the cancelation of an active subscription. The subscriptionCancel process shall allow execution in synchronous mode and will return the subscription ID of the canceled subscription in case of success. In case of failure caused by invalid query, the response shall have a HTTP error code that corresponds to the reason of the failure. The content of that response shall be based upon the OpenAPI 3.0 schema exception.yaml. The type of the exception shall correspond to the reason of the failure, e.g. InvalidParameterValue for invalid *subscriptionId*.

The process shall be defined with the following properties:

- id: value set to “subscriptionCancel”.
- title: value set to “PRIP subscription cancelation”
- description: value set to an understandable description of what the process performs

3.6.2.2 Process inputs/outputs

OGC-PRIP-PRO-REQ-150 – subscriptionCancel Process inputs

The subscriptionCancel process shall support the following mandatory input:

- **subscriptionId**: unique identifier of the subscription to cancel

The following table details the list of inputs.

Input name	Input description	Input type	Occurrences
subscriptionId	Unique identifier of the active subscription to cancel	string	1

OGC-PRIP-PRO-REQ-160 – subscriptionCancel Process outputs

The subscriptionCancel process shall support the following outputs returned as in-line values (outputTransmission set to “value”) in a JSON document response:

- **subscriptionId**: a unique ID of the active subscription. This ID shall be used by the client to cancel the subscription, and to consume corresponding notification messages.

As the process is synchronous, the response is directly returned after the execution query.

3.6.3 Notification sending

For each active subscription, the PRIP instance will monitor if conditions are met (collection and product search filter). In case of matching, a notification message is sent for each new product published into the STAC Catalogue. The process is illustrated in the sequence diagram below.

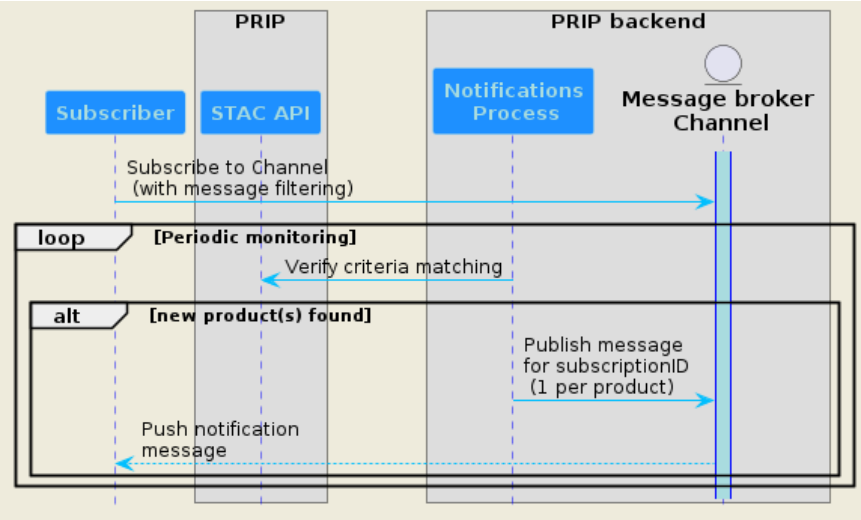


Figure 8: Notification sending sequence diagram (example for new products in 1 STAC collection)

OGC-PRIP-PRO-REQ-170 – Pub/Sub broker and notification channels

The PRIP instance shall conform to "OGC API Publish-Subscribe Workflow - Part 1: Core[2]", which relies on Pub/Sub protocol for notification message publishing. The supported protocol shall be either AMQP 1.0 or MQTT 5.0. The PRIP instance shall setup a broker and instantiate one dedicated channel per STAC collection which supports subscription mechanism. It shall also provide the following information :

- host: the server hostname and port
- protocol: Pub/Sub protocol supported by the server (mqtt or amqp)
- protocolVersion: version of the Pub/Sub protocol supported by the server
- description: string describing the host
- security: reference to supported authentication types

It shall also declare all the channels to which a consumer can subscribe.

[2] This standard is still draft (https://docs.ogc.org/DRAFTS/25-030.html)

OGC-PRIP-PRO-REQ-180 – Notification messages publishing

The PRIP instance shall publish a notification message to the Pub/Sub broker each time a client's subscription meets product filtering. It shall publish this notification on the channel corresponding to the collection the product belongs to. The client must subscribe to each channel related to the collection set in its active subscriptions.

Example: A client creates a subscription for new sentinel2-l1c products. Then he subscribes to the AMQP channel "sentinel2-l1c" and listens for notification messages. As soon as a new Sentinel2-l1c product is published and matches with its subscription's filter, he receives a notification message on the "sentinel2-l1c" channel.

OGC-PRIP-PRO-REQ-190 – AMQP Stream Filtering

The PRIP instance shall publish a notification message on the Pub/Sub broker for each product corresponding to a client's subscription.

Optionally, if the broker is AMQP compliant, this message could be addressed to the right consumer through the "AMQP Stream Filtering" mechanism: the *x-stream-filter-value* header is set to the "subscriptionID" value for which the notification must be delivered.

Then the consumer can perform effective message filtering, thanks to *x-stream-filter* argument set to a string or an array of strings if he needs consuming messages for several subscription IDs.

OGC-PRIP-PRO-REQ-200 – Notification message payload for new STAC items

The Pub/Sub broker shall publish a notification message with the following CloudEvents (JSON) formatted payload for a new product published into a STAC collection.

```
{
   "id": "UUID",     # unique message ID
   "specversion": "1.0",
   "type": "org.ogc.api.collection.item.create",
   "subscriptionid": "SubscriptionID",[3]      # allows a consumer to filter messages based on it's subscription ID
   "source": " https://s2a-prip.copernicus.esa.int/collections/collection",
   "subject":"https://s2a-prip.copernicus.esa.int/collections/collection_id/items/item_id",
   "time": "2025-08-27T12:36:33Z",
   "datacontenttype": "application/json",
   "data": {
      "type": "application/geo+json",
       "rel": "collection.item",
      "href": "https://s2a-prip.copernicus.esa.int/collections/collection_id/items/item_id"
   }
}
```

[3] This field is added for LTA service needs and is not in line with to "OGC API Publish-Subscribe" standard. It's useful if AMQP Stream Filtering cannot be implemented (see OGC-LTA-PRO-RECO-220)

Annex A - Conformance test suite

The Conformance Test Suite related to this specification is provided by the following tools:

- The STAC PRIP API validator tool, used to validate compliance with this document and which uses the STAC API validator tool to verify the conformance of the STAC API,
- The STAC PRIP API data model validation with Postman tool, used to validate the requirements of the data model and the STAC API of a PRIP instance as stated by this document with the Postman client,
- The STAC PRIP API data model validation with NodeJS tool, also used to validate the requirements of the data model and the STAC API of a PRIP instance as stated by this document with NodeJS. It is experimental; the previous tool should be privileged.

Each tool is managed in its own Gitlab Repository and each version of the tool indicates the version of the PRIP specification it is related to.

Those tools are managed under PRIP CTS Gitlab group of projects.