Copernicus Core Services Interface - Documentation

Deliverable D2.2

DATE
31 December 2020

ISSUE
1.0

GRANT AGREEMENT
no 870337

DISSEMINATION LEVEL
PU

PROJECT WEB-SITE
http://cure-copernicus.eu/

AUTHORS
Michal Opletal (GISAT)

CONTRIBUTORS
Katerina Jupova, Michal Opletal, Tomas Soukup (GISAT)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Purpose of the document

This document presents a technical description of the CURE Copernicus Core Services Interface (CCSI) and its implementation. The aim of this development activity is to provide a unified interface to streamline search and locate Copernicus Services products and other resources as requested by CURE cross-cutting applications. This will help to automate the process of data input into CURE applications and the CURE System, which will significantly enhance their replication potential for the future.

The CCSI is stored in this GitHub repository: https://github.com/gisat/cure_core_interface, and it will constantly be updated after the delivery of this report in order to allow the efficient development of the CURE System in WP4.

This document leverages a CCSI concept introduced in D2.1, worked out into the first operational implementation. In particular, this document describes:

- the way how to query Copernicus Core Services Interface to collect the available products and metadata available
- the way Copernicus Core Services Interface implements OpenSearch standards
- parameters of Copernicus Core Services Interface response
- class and implementation of Copernicus Core Services Interface application

Described application components and their classes may be upgraded or changed during the application development and operation. For this purpose, this document is considered as living one and will be revised upon the application change requests.

1.2 Text convention

- Parameter *name* and parameter *value*, URI is represent by italic

2 COPERNICUS CORE SERVICES INTERFACE

This chapter shortly explains the concept of the CURE Copernicus Core Services Interface and how it is implemented, including specification of supported protocol, protocol standards and access to the Interface.

2.1 Copernicus Core Services Interface concept

Products of Copernicus Services can be accessed from various access points like DIASes or CDSAPI etc. These access points differ from each other according to which Copernicus Services are hosted on the access point, how can be searched and in the form of standard response. Copernicus Core Services Interface is intended to ease access for CURE applications and users to Earth Observation products, especially the Copernicus

Data, from the Sentinel satellites, the Copernicus Core Services´ datasets hosted by various providers such as Creodias, Mundi, Onda and others.

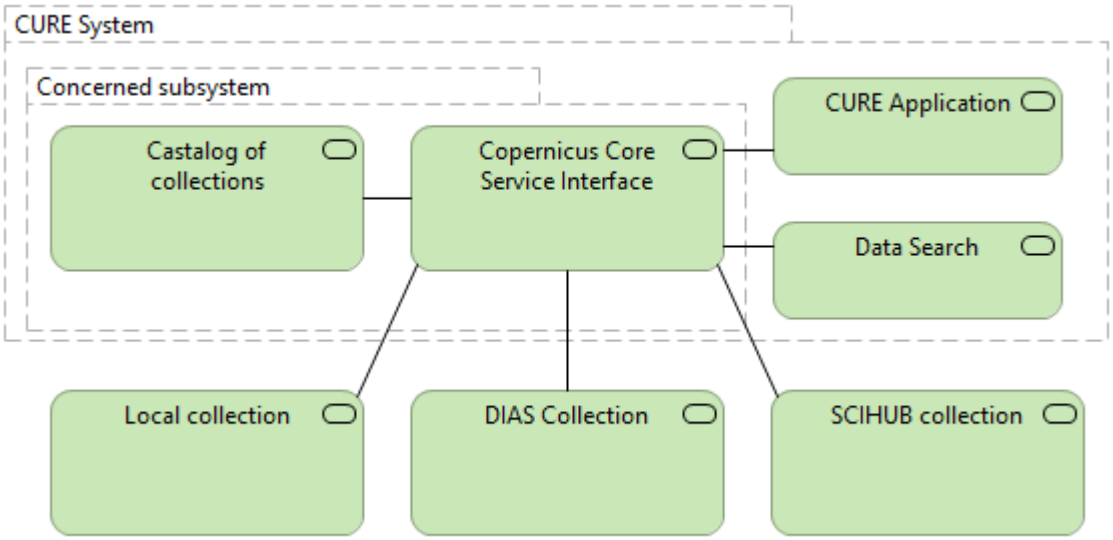

Figure 1. Position of Copernicus Core Services Interface in the CURE system.

On figure 1 is shown schema basic interaction with CCSI within the CURE system as was described in preliminary specification of CCSI in document D2.1

2.2 Copernicus Core Services Interface implementation

CCSI main components

Copernicus Core Services Interface is a RESTfull application offering search API following OpenSearch specification. By the API can be searched all registered catalogues. All received feeds for a given search query are parsed and returned in a standardized form containing product metadata, geographic information and links for downloadable or mountable content. The application does not provide only the search over DIASes but allows to search in other resources like local and city open data APIs upon their registration into the CCSI. These APIs can be integrated into the CCSI in the semi-automatic process if follows the certain specification. For resource registration, users with certain permission are allowed to register new resources.

2.3 Supported protocol

Copernicus Core Services Interface support OpenSearch API protocol:

- with query options as Parameters

- with also query options as Search Terms

2.4 Protocol standards

The OpenSearch protocol implemented in CCSI follows standards defined in the following documents:

Table 1. Implemented protocol standards

Document ID	Document Name	Issue	Link
	OpenSearch	1.1	http://www.opensearch.org/Home
OGC 10-157r3	Earth Observation Metadata Profile of Observations & Measurements	1.0	https://portal.opengeospatial.org/files/?artifact_id=47040
OGC 10-032r8	OGC OpenSearch Geo and Time Extensions	1.0	http://www.opengis.net/doc/IS/opensearchgeo/1.0
OGC 10-026r8	OGC OpenSearch Extension for Earth Observation	1.0	http://docs.opengeospatial.org/is/13-026r8/13-026r8.html

2.5 Access

Copernicus Core Services Interface access in the current version is open, and the querying access point can be reached through HTTP queries.

3 OPENSEARCH API

This chapter is dedicated to the description of the implementation of OpenSearch specification within the Copernicus Core Services Interface. In this chapter is described the API endpoints structure, search parameters, building of queries and standard responses.

3.1 Building an OpenSearch endpoint URI

Copernicus Core Services Interface is queryable with OpenSearch queries through HTTP GET requests. CCSI has two basic variations of Open Search URIs addressing the resource/products exposed by the Open Search Service:

- URI addressing all catalogues/resources registered in CCSI:

<hostname>/<path>/<response from>/search?

- URI addressing specific catalogue or resource registered in CCSI:

<hostname>/<path>/<recourse>/<response from>/search?

where:
<hostname>/<path> is service root

<response from> is the requested format of responses and has two options *json* or *atom*.

URI for response in atom format:

<hostname>/<path>/atom/search?

URI for response in json format:

<hostname>/<path>/json/search?

<recourse> is a unique name of a registered catalog or resource. Together with the rest of URI define the endpoint specific for selected catalogue/resource. This endpoint accept only catalogue/resource specific parameters

3.2 OpenSearch Description Document

Implemented OpenSearch protocol is self-descriptive. Each endpoint exposes its own description document (OSDD). Description document provides definition of available collections and parameters and their possible values.

URI for all description document describing search parameter for querying all catalogues/resources is a form:

<hostname>/<path>/<response from>/search/description.xml

This description document provide register of all parameters that are accept by global endpoint

URI for all description document describing search parameter for querying specific catalogues/resources is in a form:

<hostname>/<path>/<response from>/<recourse>/search/description.xml

This description document provides a register of recourse specific parameters that are accepted by endpoint.

3.3 OpenSearch query parameters

Parameters are limited to a short list of metadata filters.

3.3.1 Firtering catalogues and resources

Copernicus Core Services Interface providing access to various catalogues and resources. Selection of particular resources is provided by parameters:

- *recourse*
  Recourse is a metadata filter used for selection of catalogues and resources. Is an option parameter that acquires only certain values. These values are identical with *resource* 's unique name defining each endpoint. Parameter is a multi-value parameter when multiple values are separated by ",".
- collection
  Recourse is a metadata filter used for selection of collection. Certain collection can be provided by different recourses. Is an option parameter that acquires only certain values. One *resource* can have a multiple collections. Parameter is a multi-value parameter when multiple values are separated by ",".

3.3.2 Temporal filtering

Date metadata can be filtered through temporal filtering. It provided by two parameters:

- *timestart*
- *timeend*

Expected format of parameter is in form "yyyy-mm-ddThh:nn:ss" that can be shortened for more time precise elements.

<hostname>/<path>/<recourse>/<response from>/search?timestart=2020-12-18T12:00:00

alternatively

<hostname>/<poth>/<recourse>/<response from>/search?timestart=2020-12-18

3.3.3 Geographical filtering

Geographical metadata can be filtered by parameters:

- *geometry*

Geometry parameter accepts geometry in WKT format coordinates in decimal degrees (EPSG:4326). Accepted geometries are: polygon, linestring, point

<hostname>/<path>/<recourse>/<response from>/search?geometry=POLYGON((-4.53 29.85,26.75 29.85,26.75 46.80,-4.53 46.80,-4.53 29.85))

- *bbox*

Bbox parameter filter data base on geographical bounding box. Coordinates are expected in decimal degrees (EPSG:4326) in order west, south, east, north. e.g. bbox=-61.3,14.3,-60.8,14.9

*<hostname>/<path>/<recourse>/<response from>/search?*bbox=-61.3,14.3,-60.8,14.9

- lat, lon, radius
  Lat, Lon, Radius parameters have to be provided together. Lat, Lon parameters are expected in decimal degrees (EPSG:4326). Radius as a float in meters

*<hostname>/<path>/<recourse>/<response from>/search?*lat=-61.3&lon=14.3&radius=1000

3.3.4 Filtering for specific product

Specific product can be selected by providing *productid*

*<hostname>/<path>/<recourse>/<response from>/search?*productid=z_cams_c_ecmf_20200616120000_prod_fc_sfc_062_gtco3

3.3.5 Entries filtering

Received entries for given query can be filtered by parameters:

- maxrecords
  Maxrecord parameter defines the number of entries per page. expected type is integer. Default value is 50

*<hostname>/<path>/<recourse>/<response from>/search?*maxrecords=50

- startindex

Startindex parameter defines from which index the entries will be returned. Minimum value is 1

*<hostname>/<path>/<recourse>/<response from>/search?*startindex=5

- page

Page parameter defines from which page will be returned. Minimum value is 0

*<hostname>/<path>/<recourse>/<response from>/search?*page=5

3.3.6 Entries sorting

Response entries sorting can be provided by three parameters:

- sortorder
  Sortorder parameter define if the entries will be sorted in ascending or descending order. Parameter *sortorder* is option parameter and accepted values are: asc,desc or ascending,descending. Default value is *descending*

<hostname>/<path>/<recourse>/<response from>/search?sortorder=desc

- sortby
  Sortby parameter define which parameters will be used for sorting. Parameter is a multi-value parameter when multiple values are separated by ",". When multiple values are used, they are treated as an order in which entries will be displayed

- Default sort key is *starttime*

*<hostname>/<path>/<recourse>/<response from>/search?*sortby=startime,custom:*resolution*

- preferdrecourse
  CCSI provides access to multiple resources. Parameter *preferdrecourse* defines from which resource the entries will be displayed first. Parameter is a multi-value parameter when multiple values are separated by ",". When multiple values are used, they are treated as an order in which entries will be displayed

*<hostname>/<path>/<recourse>/<response from>/search?*preferdrecourse=creodias,mundi,

3.3.7 Special parameters

- custom

Copernicus Core Services Interface is intended to provide access to various catalogues/resources. The exposed parameters for filtering and specification of products provided by single recourse may differ from others. This resource specific parameters are registered with the prefix "*custom:*" e.g. parameter for the specification of orbitderection is labelled as *custom:orbitdirection*. List of custom parameters and their specification, pattern, optional or default values is accessible via description dokuments.

<hostname>/<path>/<recourse>/<response from>/search?custom:orbitdirection= descending

- *solr*
  Some registered catalogues/resources allow use of Apache Lucene free text search. Parameter *solr* is a boolean parameter that defines the searchterm following the free text search convention. If *solr=true,* search query is injected only into the resources that supports free text search. Default value is *false*

<hostname>/<path>/<recourse>/<response from>/search?searchterm=(platformname:Sentinel-1 AND producttype:SLC AND sensoroperationalmode:SM)&solr=true

3.3.8 Query string format

Query string accepted by CCSI OpenSearch API:

- is expected in form *parameter=value*

.../<response from>/search?searchterm=value

- query string is not case sensitive. Accepted response arguments are converted to lowercase
- multiple parameters a separated by "&" letter

.../<response from>/search?searchterm=value&productid=value

- is accepted multiple choice parameters i.e parameters with multiple values. Multiple values are separated by "," letter

.../<response from>/search?collection=cams,clms

3.3.9 OpenSearch search terms

Parameter *searchterm* can query all the non-standard queryable keywords. Together with parameter *solr* can also content free text search query.

3.4 OpenSearch response

Copernicu Core Services Interface provides results in atom or json form depending on the requested endpoint.

3.4.1 Atom response.

The global answer for the requested query is embedded in a <feed> XML element. Sub elements of <feed> consisted of ten elements defining the response head. Entries are embedded by <entry> XML element.

Response head

<title> - name of the service
<author> - author of service
<id> - uuid of request query
<totalResults> - total count of all founded entries
<startIndex> - number of the first returned results, default is 1
<itemsPerPage> - number of returned entries per page, defined by maxrecords parameter. Default value of return entries is 50.

In order to help the browsing process, the OpenSearch result provides useful links through the <link> XML element.

<link rel="self"> - refers to current query
<link rel="firs"> - refers to first page of requested entries
<link rel="next"> - refers to next page of requested entries
<link rel="last"> - refers to last page of requested entries

Important parameters of response entry

The requested response consisted from XML elements describing product metadata. These metadata differs between the resources. If metadata from the original resource does not match or is missing in the list of standard Copernicus Core Services Interface. They are not printed out. Following list of XML elements represents the base set for every entry.

<link rel=”enclousure”> - provides link to downloadable content
<link rel=”path”> - provides link to mountable location, if exists
<link rel=”search”> - provides link to entry itself
<id> - provides product id from original recourse
<ccsi:status> - refers if the product is available online
<gml: *> - geographic reference in Geography Markup Language format

Example of response

<feed xmlns="http://www.w3.org/2005/Atom" xmlns:dc="http://purl.org/dc/elements/1.1/” xmlns:gml="http://www.opengis.net/gml" xmlns:ccsi="http://192.168.99.102:5000/ccsi">
<title>Copernicus Core Service Interface search results</title>
<subtitle>Displaying 1 results</subtitle>
<updated>2020-12-18T00:00:00</updated>
<author>
<name>Copernicus Core Service Interface</name>
</author>
<id>8b734ed6-4917-47ea-a597-0ea010658942</id>
<totalResults>1</totalResults>
<startIndex>0</startIndex>
<itemsPerPage>1</itemsPerPage>
<Query role="request" searchTerms="searchterm=water&catalogue=mundi&maxrecords=1"/>
<link rel="search" type="application/opensearchdescription+xml" href="http://192.168.99.102:5000/atom/search/description.xml"/>
<link rel="self" type="application/atom+xml" href="http://192.168.99.102:5000/atom/search?searchterm=water &catalogue=mundi&maxrecords=1&collection=clms&startindex=1&page=1"/>
<link rel="first" type="application/atom+xml" href="http://192.168.99.102:5000/atom/search&searchterm=water& catalogue=mundi&maxrecords=1&collection=clms&startindex=1&page=1"/>
<link rel="next" type="application/atom+xml" href="http://192.168.99.102:5000/atom/search&searchterm=water &catalogue=mundi&maxrecords=1&collection=clms&startindex=1&page=1"/>
<link rel="last" type="application/atom+xml" href="http://192.168.99.102:5000/atom/search&searchterm=water &catalogue=mundi&maxrecords=1&collection=clms&startindex=1&page=1"/>

```
<entry>
<id>1dcfa016-e904-410d-ab57-1ba1607a4587</id>
<title>Corine Land Cover 2000 - 2006 changes (raster 100m) - version 18, Mar. 2016</title>
<category term="Land cover"/>
<category term="land use"/>
<category term="land cover"/>
<category term="landscape"/>
<category term="landscape alteration"/>
<category term="Copernicus"/>
<category term="CLCC2000-2006"/>
<category term="EEA39"/>
<category term="geospatial data"/>
<category term="environment"/>
<category term="imageryBaseMapsEarthCover"/>
<link rel="enclosure"
href="https://cs-clms.obs.otc.t-systems.com/CLMS_products/
pan-European/CLC/LCC2000-2006/raster/g100_ch00_06_V18_5.zip"/>
<link rel="search" type="application/atom+xml"
href=”http://192.168.99.102:5000/atom/search&?
uid=1dcfa016-e904-410d-ab57-1ba1607a4587"/>
<dc:identifier
<dc:identifier>1dcfa016-e904-410d-ab57-1ba1607a4587</identifier>
<dc:date >2018-01-12T17:36:17Z</date>
<dc:creator>sdi.eea</creator>
<gml:Polygon srsName="EPSG:4326">
<gml:outerBoundaryIs>
<gml:LinearRing>
<gml:coordinates>
-29.086205435,12.994105341
-29.086205435,12.993797401
-29.086664115,12.993797401
-29.086664115,12.994105341
-29.086205435,12.994105341
</gml:coordinates>
</gml:LinearRing>
</gml:outerBoundaryIs>
</gml:Polygon>
<published>2018-09-28T14:26:45Z</published>
<ccsi:status>ONLINE</status>
</entry>
</feed>
```

3.4.2 Json response

Json feed representation is based on the basis of FeatureCollection, where FeatureCollection represents the feed and each Feature represents the single entry. Representation violates standard GeoJSON format standard described in ISSN: 2070-1721 (https://tools.ietf.org/html/rfc7946#section-7.1) when the Feature property attribute is used as head containing head attributes similar to xml representation.

Each feature follows the Geojson standards and can be considered as valid forma. Product metadata are located in feature property attribute

Transformation between xml and json follows the rule that the tags are serialized as attributes. Similar tags with different attributes are serialized as a list of tag attributes dictionaries. Tag text field is serialized as a list.

3.5 Error handling

Table 2. Error Handling

Error type	description
400 Bad Request	Raquest has an invalid syntax
413 Request Entity Too Large	The request originates too many returnable hits
500 Internal Server Error	
501 Not Implemented	Unsupported operator
503 Service Unavaila	Service is temporarily not available
504 Gateway Timeout	Failing to produce a answer within a giving

4 COPERNICUS CORE SERVICES INTERFACE APPLICATION

In this chapter are described the processes and operation used in the Copernicus Core Service application. These processes include the implemented resources search logic, request parsing and validation, transformation operation, response parsing and registration of resources.

4.1 Validation of request and finding relevant resource

Copernicus Core Services Interface is intended to provide access to various catalogues resources. In order to accomplish this task and not wasting processing time to search over the resources that have not requested products is applied simple logic to find relevant resources.

- The incoming request is parsed into a single dictionary containing request parameters and their respective values
- The dictionary is compared against the cache. If it is find same request, cached response is return
- It provides a check if the request contains only valid parameters against the central register of parameters. Concurrently if the parameter is valid, from the central register of resources are taken reference on the resource that has requested parameter and added into the resources pool. If any of parameter is invalid, the process is aboard with custom error page 501 with an error message
- All parameter values are validated against the resources in the resources pool. If the value of parameter is invalid If any of parameter is invalid, the process is aboard with custom error page 501 with an error message. Exception from this rule is validation of option parameters like *custom:producttype.* If the option parameter has not requested value, then the resource is removed from the pool and another parameter is validated against the rest of resources.
- Then if some executed send_request method on the interface of remaining resources in resources pool

4.2 Transformation

4.2.1 Request transformation

To provide communication between base Copernicu Core Services Interface API and resources API is implemented a resource mapping table that maps resource parameters on base api parameters. Parameters that do not exist are registered into the central parameter register. Together with mapped parameters and their specification is a registered transformation function that transforms parameter values into values expected by a particular resource. Most common transformation function is the identity function.

4.2.2 Response transformation

Incoming responses are parsed into the standard type representing single entry. LXML and GDAL library was implemented to parse xml type responses and correctly handle various geographic reference formats used in resources responses.

Similarly to request transformation, mapping tables are used to map resource response attributes into standard entry type. High level of nonstandard responses and ill formating led to implementation of transformation functions that are applied on the response to make them parsable and importable into standard entry type.

4.3 Registration

Registration process is used to integrate new resources into the Copernicus Core Services interface. Registration process of a new resource consists of setting the properties of four resource components.

- Connection
  Connection class provides the connection between CCSI and resources. Injected properties are:
  - resource base url
  - type of authentication and alternatively credentials
  - rule function to handle specific query string generation
  In mouse cases registration of this component is automatic
- Service parameters
  Class represents resource API. Consisting prom specification of parameter names, definition of expected values, selection of transformation function
  Registration is a semi-automatic process, because most of the information is provided by OpenSearch description documents.
- Request mapping table
  Provide 1 to 1 table between base API parameters and resource parameters
- Entry mapping table
  Provides the definition for the parser factory.

4.4 Backoffice

CCSI backoffice was intended to facilitate the process of registration of new resources. The idea of registration was in displaying several forms that allow the setting of mandatory resources components. Current status of CCSI BackOffice is UNFINISHED due to the problems with ill-formatted responses and parser. More development is needed to accomplish this part involving implementation of the parser testing in the registration process. Registration how it was intended works in the case of registration open data API like Heraklion where parser is not involved.

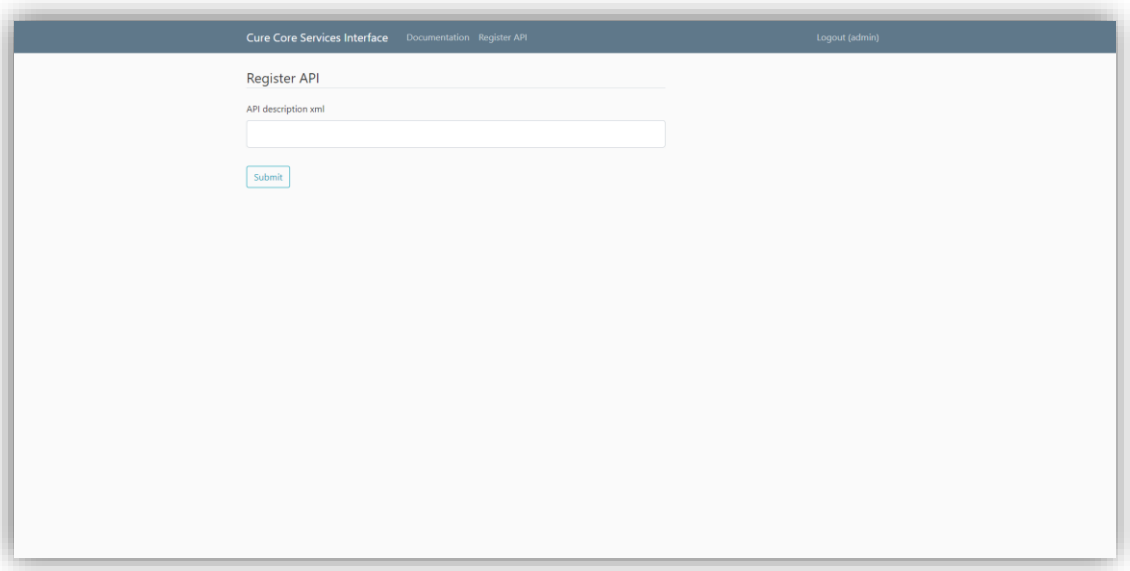

Figure 2. CCSI Backoffice - Register API template

4.5 List of registres resources

The following list are listed resources integrated into CCSI to date [2020/12/18]

- scihub - Scihub endpoint OpenSearch API specification
- creodias - Sentinel 1 endpoint OpenSearch description
- creodias - Sentinel 2 endpoint OpenSearch description
- creodias - Sentinel 3 endpoint OpenSearch description
- mundi - Sentinel 1 endpoint OpenSearch description
- mundi - Sentinel 2 endpoint OpenSearch description
- mundi - Sentinel 3 endpoint OpenSearch description
- mundi - CLMS endpoint OpenSearch description
- ondata - global endpoint OpenSearch description
- Heraklion - weather online devices endpoint OpenData endpoints

5 CONCLUSION

This document presents the current version of the CURE Copernicus Core Services Interface (CCSI) and its components and functionalities. The interface described here represents the first version of the CURE Copernicus Core Services Interface, as released in M12, in accordance with the planned project schedule.

Current version of the Interface is able to provide a link between the CURE system (including cross-cutting applications) and various repositories storing the Copernicus Services´ products or other (third-party) datasets which will serve as inputs for CURE applications. Current development status of Interface allows using it in particular for searching data resources requested by CURE cross-cutting applications - i.e. Copernicus products in particular Copernicus Core Services datasets and Sentinel satellite imageries or city open data. Moreover, it allows for registering other resources into the CCSI if these resources expose their API or OpenSearch description document.

The registration process of the new resources in the current version of CCSI is not at the state allowing to register resources without knowledge of application and potential changing in application code. Facilitation of the registration process will be an object of further development

This first version of the Interface will be further developed, improved and adjusted in accordance with initial specification and planned project schedule and also according to specific needs of CURE applications. This will be done in close iteration with the work conducted in the frame of WP3 and WP4. The CCSI will be integrated with other components of the CURE system, which may lead to identification of additional requirements on the Interface. These requirements will then be addressed in the next phases of the Interface development in WP2.

Among others, in the next phases of the Interface development, emphasis will be put on the implementation of PostgreSQL catalogue and application of crawler to Creodias with aim of indexing Copernicus products located on this infrastructure. Special attention will be paid to the development of a robust parser of resources OpenSearch responses with the aim to handle ill-formatting and different implementation of OpenSearch specification that is the main source of obstacles and delays.

The updated version of the Interface will then be described in the next (and final) WP2 deliverable - D2.3 Copernicus Core Services Interface Update (M30).