Situation-Aware Linked heTerogeneous Enriched Data

D3.1: Report on development of Scorpio Context Broker – CKAN connector

Work package	WP 3
Task	Task 3.1
Due date	30/04/2023
Submission date	28/04/2023
Deliverable lead	UC
Version	1.0
Authors	Laura Martín (UC), Víctor González (UC), Jorge Lanza (UC), Pablo Sotres (UC), Juan Ramón Santana (UC)
Reviewers	Benjamin Hebgen (NEC), Luis Sánchez (UC)

Abstract	This document, developed by the SALTED project, represents the D3.1 deliverable on the development of the Scorpio Context Broker – CKAN connector. The focus of this document is the thorough definition of all the components that allow a connection between the SALTED private platform and a public CKAN instance. Furthermore, D3.1 includes information on the connector that

This project is co-financed by the Connecting Europe Facility of the European Union under the Action Number 2020-EU-IA-0274.

	allows CKAN datasets to be harvested by the EDP for publication.
Keywords	CKAN, EDP, Open Data

Table of Contents

Table of Figures

1 INTRODUCTION

1.1 SCOPE OF DOCUMENT

One of the key goals of SALTED is the publication of the NGSI-LD high-value data generated within the project in the European Data Portal (EDP). This document describes the required connectors that interconnect the SALTED architecture to the EDP, by leveraging a Comprehensive Knowledge Archive Network (CKAN) public instance deployed in between.

The document details the implementation and deployment of the concerned connectors, their internal components, and the data models they use. The resulting platform, which includes the Scorpio Broker, the CKAN instance, and the connectors, is aligned with the main output expected from Work Package 3 in general, and Task 3.1 in particular.

1.2 TARGET AUDIENCE

This document is mainly intended for internal use, although it is publicly available in order to raise awareness of the SALTED project and its contribution to the EDP. The target audience is the SALTED technical team including all partners involved in the delivery of Work Packages 1, 2 and 4, and especially those involved in the activities of Work Package 3. The document provides a thorough definition of the infrastructure connecting the private SALTED development platform with the public endpoints that will allow users to access the data provided by the project.

1.3 STRUCTURE OF THE DOCUMENT

We first describe the SALTED Platform to EDP connector architecture and the two main data management components (CKAN and EDP) in Section 2. Sections 3, 4, and 5 review the internal architectures of the 3 sub-connectors developed and implemented, outlining their main functionalities and the components involved. Finally, Section 6 summarises the major conclusions drawn from the development and implementation of the main connector architecture.

2 SALTED PLATFORM TO EDP CONNECTOR IMPLEMENTATION

2.1 ARCHITECTURE OVERVIEW

The SALTED architecture, regarding the publishing of enriched data in the European Data Portal[1], consists of a set of connectors that bridge the components involved. These connectors allow the assessment and forwarding of the entities stored in each of the so-called Satellite Scorpio Context Brokers through the Federation setup previously explained in [1].

Figure 1 represents the high-level architecture implemented for the SALTED Platform – EDP connector. This architecture encompasses the entire life cycle of data collections, from their registration and description to their publication on the EDP. As shown in Figure 1, there are four main components bound together by three connectors, depicted as grey arrows. Each of these connectors will contribute to the description of the final datasets using the advantages of the NGSI-LD information model [2], such as semantic and linked data principles. This allows the SALTED Platform to offer high-quality data with high-quality metadata, as the EDP requires a certain level of data quality regarding description and accessibility of the datasets.

The data provided to the EDP is grouped by the NGSI-LD Entity Type (e.g. Temperature or AirQualityObserved). Each dataset will be presented in different formats (JSON and JSON-LD) and with different temporal contexts (real time and last month or week).

The first component in the proposed architecture, the Dataset Registry, will be explained in the next section along with the first connector, whereas the second component, Federator Scorpio CB, has already been defined in [1]. Regarding the latter ones, CKAN and EDP, these will be both outlined in the next subsections to show the benefits of using each and the interaction possibilities they offer.

Additionally, all the available connectors in the architecture will be described in further detail in the upcoming sections.

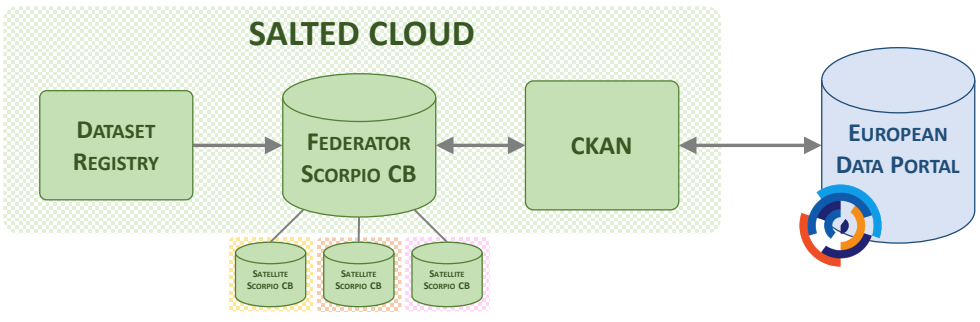

Figure 1. SALTED Platform to EDP connector architecture

[1] https://data.europa.eu/en

2.1.1 European Data Portal

The European Data Portal (EDP) [3] is the central point of access to European open data. Its main objectives are to give open access and foster the reuse of data, support the release of higher-quality data, and promote the advantages deriving from the availability of open data.

This organization shares the mission, vision and values of the Publications Office of the European Union (OP) on open data. Its mission is to ensure that all this information gathered is available, accessible and reusable to facilitate transparency and dissemination of knowledge. The vision of the EDP is a well-informed European Union, with access to reliable information and knowledge, exploiting to the maximum the opportunities provided by this context awareness. And finally, the EDP values are based on transparency, trustworthiness, accessibility and service orientation.

The portal is a metadata catalogue[2]. In other words, it is a portal in which all available datasets are presented through attributes and properties that describe the most important characteristics, such as title, access URL, supported formats or geographical scope. It uses a familiar format for the metadata representation through the DCAT-AP (Data Catalogue Vocabulary – Application Profile) specification, using RDF (Resource Description Framework) syntax.

Furthermore, it is important to highlight that the content offered by this open data portal allows consumers and society to generate value-added services. Likewise, the portal itself provides services for its consumption, such as its search engine, with geospatial search options, its SPARQL endpoint or its API endpoint, as well as its metadata quality dashboard[3], displaying the parameters and quality scores of each of the datasets to offer this extra informative value to the user.

2.1.2 CKAN

CKAN [4] is an open-source DMS (data management system) for the creation data hubs and data portals. Used by governments, research institutions and other organisations, CKAN provides a set of tools to manage and publish collections of data in a central endpoint. Additionally, CKAN provides users the possibility to browse through all the information until finding the collection they need according to certain criteria.

Within the CKAN context, data is published in units called *datasets*, bound to one or several organizations. A dataset is made of two ingredients:

- **Metadata:** extra information about the data. Attributes such as the title of the dataset, author and publisher, license or even in what formats the data is available in.
- **Resources:** data itself, regardless of its format (CSV, RDF, JSON…). CKAN is able to store the data internally (e.g. a file) or store it as a link pointing to another endpoint in the web containing the actual data.

In addition to the data management system, CKAN offers several useful features that add value to its organisation. The most representative features encompass the CKAN API, which provides full access to the core functionality; CKAN extensions, with the possibility to pick and

[2] https://data.europa.eu/es
[3] https://data.europa.eu/mqa/?locale=en

choose which characteristics and facilities for the CKAN instance, as well as design and development of custom plugins; metadata handling, including default fields and allowing the chance to add custom ones; and geospatial properties management.

More information on these and the remaining features can be found in the official CKAN homepage[4].

[4] https://ckan.org/features

3 USER TO CONTEXT BROKER CONNECTOR

3.1 ARCHITECTURE OVERVIEW

The first connector accessible in the overall architecture is the User to Context Broker connector, shown in Figure 2. The purpose of this connector is to register the desired datasets in the Federator Context Broker, along with customizable metadata.

The workflow followed is straightforward. The SALTED webpage hosts a form[5] that provides the SALTED partners the opportunity to customise the description of the Entity Types that will be federated into the Scorpio Context Broker and, later on, published in the EDP. As mentioned previously, EDP highly encourages the use of metadata for the granular and in-depth description of the datasets.

The submitted information is received in the Dataset Registry Module, which performs a mapping between the fields included in the webpage form and the available properties in the corresponding data model from the Smart Data Models initiative[6], using the NGSI-LD information model. Since in this phase it is only intended to make a record of the federated types or datasets to be generated, the Smart Data Models used are Dataset and CatalogueDCAT-AP, from the DCAT-AP subject[7]. Both will be explained at a later point in Section 3.2.1.

Once the module has the required NSGI-LD entities, it creates or updates the existing information in the Federator Scorpio Context Broker. using its own exported NGSI-LD API. These entities will be used as starting points for the next connector, the so-called Context Broker to CKAN connector.

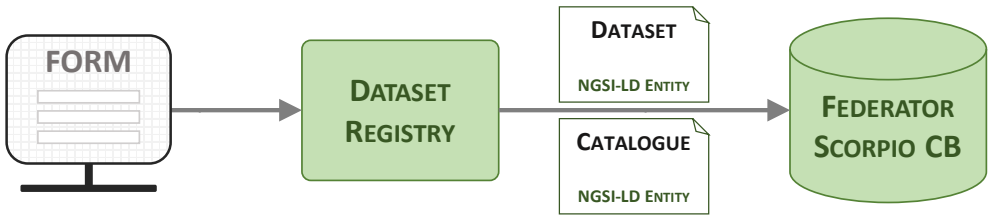

Figure 2. User to Context Broker connector architecture overview

3.2 IMPLEMENTATION

This section describes in detail all the elements involved in the User to Context Broker connector: the Smart Data Models used, an overview of the form template fields and, finally, the purpose of the Dataset Registry module and its inflows and outflows.

It is worth noting that the implementation of this connector is in its first version, and we are already working on the improvement and expansion of the functionality of each of the

[5] https://salted-project.eu/ckan-type-form/
[6] https://smartdatamodels.org/
[7] https://github.com/smart-data-models/dataModel.DCAT-AP

components. These forthcoming actions will be discussed under each of the elements and will be completely described in the next deliverable (D3.2).

3.2.1 NGSI-LD Data Models

Given that this connector's aim is focused on the registration of the Entity Types of the generated datasets, the two Smart Data Models Dataset and CatalogueDCAT-AP have been adopted. In this section, the properties used for both Smart Data Models in this first version of the connector are outlined and are likely to be extended in the ongoing work towards a new version.

Dataset

This data model can be found within the Smart Data Models[8].

The Dataset data model is used for the representation of datasets following the schema of the DCAT-AP 2.0 specification. Following the principles of Smart Data Models, the properties to be used may differ according to the needs of each use case. Based on the SALTED use cases, apart from the core properties of an NGSI-LD Entity (*id* and *type*), and due to the possibility of using the remaining properties defined in the specification according to the requirements as the development progresses, the properties currently used are:

- **datasetDescription:** It is an array containing free-text descriptions of the Dataset. It corresponds with the *description* mandatory property of DCAT-AP 2.0.1. It may take more than one value repeated for parallel language versions of the description.
- **title:** It is an array containing the name given to the Dataset. It corresponds with the *title* mandatory property of DCAT-AP 2.0.1. It may take more than one value repeated for parallel language versions of the name/title.
- **publisher:** It refers to the organisation responsible for making the Dataset available. It corresponds with the *foaf:Agent* property of DCAT-AP 2.0.1.
- **theme:** It refers to a category of the Dataset. It should take a value among those available[9] at the Publications Office of the European Union. A Dataset may be associated with multiple themes and it corresponds with the *dcat:theme* property of DCAT-AP 2.0.1.

An example of use of the Dataset entity is shown in Figure 3.

CatalogueDCAT-AP

This data model can be found within the Smart Data Models[10].

The CatalogueDCAT-AP data model is a representation of a catalogue of datasets compliant with DCAT-AP specification. Based on the SALTED use cases and apart from the core properties of an NGSI-LD Entity (*id* and *type*), the properties used are:

- **dataset:** It is an array that links the Dataset entities contained in the Catalogue. It corresponds with the *dcat:Dataset* mandatory property of DCAT-AP 2.0.1.

[8] https://github.com/smart-data-models/dataModel.DCAT-AP/tree/master/Dataset
[9] http://publications.europa.eu/resource/authority/data-theme
[10] https://github.com/smart-data-models/dataModel.DCAT-AP/tree/master/CatalogueDCAT-AP

- **description:** It is a free-text description of the Catalogue.
- **publisher:** It refers to the organisation responsible for making the Catalogue available. It corresponds with the *dct:publisher* property of DCAT-AP 2.0.1.
- **title:** It is an array containing the name given to the Catalogue. It corresponds with the *rdfs:Literal* property. It may take more than one value repeated for parallel language versions of the name/title.

An example of use of the CatalogueDCAT-AP entity is shown in Figure 4.


```
{
    "id": "urn:ngsi-ld:Dataset:Temperature",
    "type": "Dataset",
    "datasetDescription": {
        "type": "Property",
        "value": [
            "This is the description of the Dataset Temperature"
        ]
    },
    "title": {
        "type": "Property",
        "value": [
            "Temperature"
        ]
    },
    "publisher": {
        "type": "Property",
        "value": "salted-project"
    },
    "theme": {
        "type": "Property",
        "value": [
            "ENVI",
            "REGI"
        ]
    },
    "@context": [
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.6.jsonld",
        "https://raw.githubusercontent.com/SALTED-Project/contexts/main/context.jsonld",
        "https://raw.githubusercontent.com/smart-data-models/dataModel.DCAT-AP/master/context.jsonld"
    ]
}
```


Figure 3. Example of a Dataset entity


```
{
    "id": "urn:ngsi-ld:Catalogue:SALTED",
    "type": "CatalogueDCAT-AP",
    "dataset": {
        "type": "Relationship",
        "object": [
            "urn:ngsi-ld:Dataset:Temperature",
            "urn:ngsi-ld:Dataset:AirQualityObserved"
        ]
    },
    "description": {
        "type": "Property",
        "value": "This is the description of the SALTED Catalogue"
    },
    "publisher": {
        "type": "Property",
        "value": "salted-project"
    },
    "title": {
        "type": "Property",
        "value": [
            "SALTED"
        ]
    },
    "@context": [
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.6.jsonld",
        "https://raw.githubusercontent.com/SALTED-Project/contexts/main/context.jsonld",
        "https://raw.githubusercontent.com/SEMICeu/DCAT-AP/master/releases/1.1/dcat-ap_1.1.jsonld",
        "https://raw.githubusercontent.com/smart-data-models/dataModel.DCAT-AP/master/context.jsonld"
    ]
}
```


Figure 4. Example of a CatalogueDCAT-AP entity

3.2.2 Form

The form[11] hosted by the SALTED webpage enables the immediate interaction of the partner's user with the platform, by filling in the information required in a personal manner and, thus, allowing the datasets description customisation. The information submitted by this form will also be used to create and set up the federation registry through the *cSourceRegistration* entities.

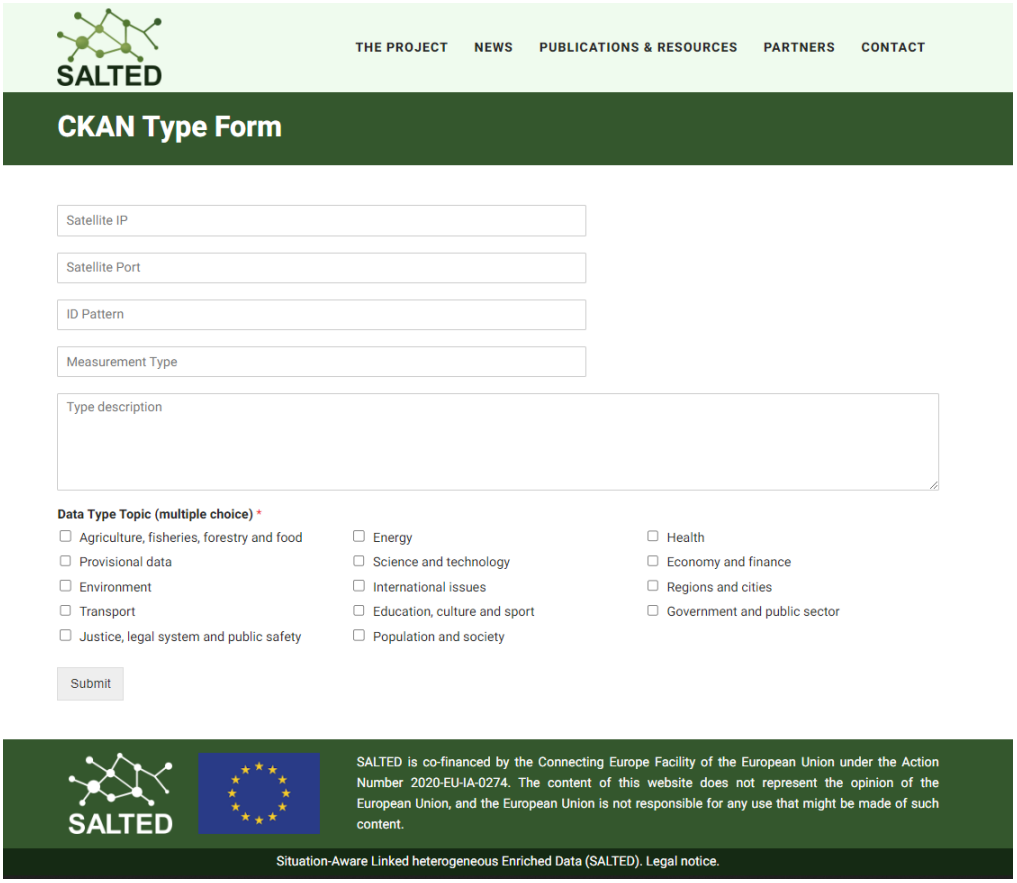

Figure 5. Form appearance

Figure 5 shows the form's current appearance and the fields that need to be filled in, which are described as follows:

- **Satellite IP:** IP address of the Satellite Scorpio Context Broker endpoint.
- **Satellite Port:** Port of the Satellite Scorpio Context Broker endpoint.
- **ID Pattern:** ID Pattern of the entities that will be federated.
- **Measurement Type:** Entity type.
- **Type description:** Entity type description. Customisation of the dataset description in the CKAN and EDP.
- **Data Type Topic:** Theme of the dataset (description available[12]).

[11] https://salted-project.eu/ckan-type-form/

[12] http://publications.europa.eu/resource/authority/data-theme

As previously mentioned, an improved version of the connector is being developed, enhancing the scope of available customisation options in the form. These changes will be collected in the next deliverable of Work Package 3 (D3.2).

3.2.3 Dataset Registry

The goal of the Dataset Registry module is to perform a mapping between the data submitted through the form and the chosen properties of the NGSI-LD data models. Thereby, two NGSI-LD Entities are generated and obtained at its output, both defining and describing the desired dataset to be recorded and the organization or catalogue associated with its publication.

The first step is to identify the correlation between the information collected through the website form and the attributes of the proposed data models.

Regarding the Dataset data model, the mapping of the fields is straightforward: The *Measurement Type* is mapped in the *title* property, the *Type description* is mapped in the *datasetDescription* property and, finally, the *Data Type Theme* is mapped in the *theme* property. It is important to note that the *publisher* property remains to be filled in because in our approach all data provided by the SALTED platform will be considered as SALTED data. Therefore, the *publisher* is the SALTED project itself. A realistic example can be seen in Figure 3.

According to this methodology on the SALTED data, a single CatalogueDCAT-AP entity is used to gather all the datasets that will be generated. That is to say, the *description*, the *publisher* and the *title* fields will be both static and specific defining the SALTED organization. The *dataset* property will be constantly updated each time a new dataset is recorded. A realistic example can be seen in Figure 4.

Once both entities have been generated, for Dataset and CatalogueDCAT-AP, an UPSERT request is made to the Federator to incorporate the information generated (either for the first time or by updating existing entities), thus making the registry of the desired datasets.

In the future improvements which will be carried out, it is clear that by increasing the number of customisation fields in the form and making a broader use of the Smart Data Models presented, the functionality of the Dataset Registry module is going to evolve. Its main objective will remain the same, to perform a conversion from the NGSI-LD format of the information stored in the Context Broker to the CKAN format in order to be able to import the datasets. But these changes will entail a greater automatization of the mapping between properties and attributes, allowing the user to have more decision-making power in the dataset metadata.

4 CONTEXT BROKER TO CKAN CONNECTOR

4.1 ARCHITECTURE OVERVIEW

The Context Broker to CKAN connector is the second connector in the overall architecture outlined previously in Section 2.1. It comprises all the modules and interfaces necessary to establish the communication between the Scorpio Federator Context Broker and the CKAN data management system. The purpose of this connector is to render the datasets accessible from CKAN via the description created with the first connector.

The workflow follows two paths: the injection of these dataset descriptions into the CKAN and the access to the required information in the Scorpio in several formats, as specified in the previous description. These two paths can be easily seen in Figure 6.

First, the *ckanext-ngsild-harvester* plugin, which will be described in detail in Section 4.2.2, subscribes to entities of type Dataset. Therefore, each time a new Dataset entity is posted to the Scorpio Federator Context Broker by the Dataset Registry, the *ckanext-ngsild-harvester* is notified of this new information. It will then generate the Dataset description in CKAN format and import it into the CKAN data management system.

One of the CKAN Dataset properties is the URL where the represented data is available. This is where the Information Retriever comes in. This URL points to this module, specifying the kind of information that is needed through the query parameters (e.g. real-time temperature in JSON format). Once this request is received, it will act as a reverse proxy, redirecting the request to the Scorpio and forwarding the response back to the CKAN.

As a result of this phase, the description of the data in CKAN units - Datasets and Resources - is made available as open data through this system.

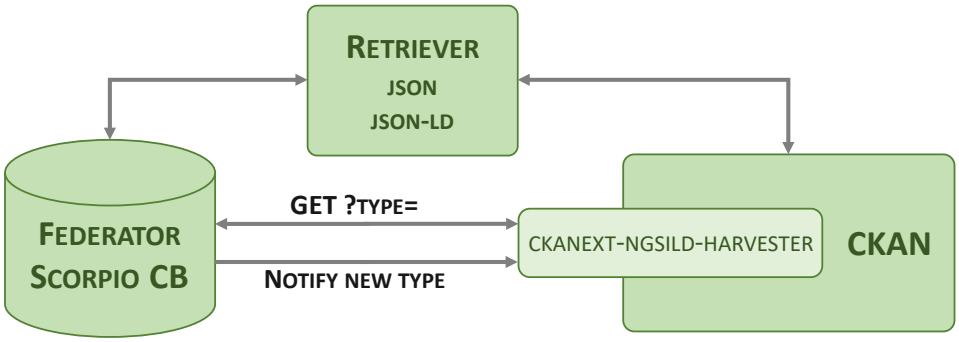

Figure 6. Context Broker to CKAN connector architecture

4.2 IMPLEMENTATION

In this section, all the elements involved in the Context Broker to CKAN connector are described in detail: the two plugins developed for the CKAN module, altering its appearance, and performing the import of Datasets, and the Information Retriever module behaving as a reverse proxy.

4.2.1 CKANEXT-SALTED_TEMPLATE

The purpose of this plugin is to modify the appearance of the CKAN site using the representative colours of SALTED.

The official CKAN documentation[13] has been followed for the creation of the extension. After its schema generation, two new directories have been created under the path `/ckanext/salted_template`: `/public` and `/templates`. The first one includes files that can be publicly accessed, e.g. images and stylesheets. The second directory contains all the HTML templates to be modified. A visual representation of the created directory tree can be found in Figure 7.

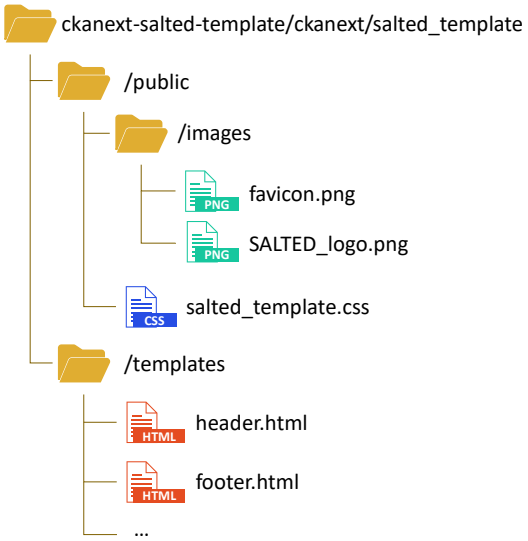

Figure 7. Directory tree of the ckanext-salted_template plugin

Beginning with the first directory, in the current version of the plugin, the main stylesheet has been altered, essentially changing the main colours to match the SALTED theme (green tones), as opposed to the default blue tones in the standard version of the CKAN. This CSS file will replace the native one that comes by default with the CKAN system installation. Additionally, the SALTED logo and the favicon have been included in the `/public/images` directory so they appear on the main page of the CKAN (`index.html`). These two files are accessed independently of the plugin, but since this is an extension for customising SALTED's CKAN theme, it seems appropriate to enclose them in the plugin's directory.

Regarding the second directory, `/templates`, the html files to be modified are added. Following the tutorial and the official documentation[14] provided by CKAN, the necessary adaptations were made. There are two types of amendments: extension and alteration of the default code. The first one includes use cases where only extra features are to be attached to the original template, using the line `{% ckan_extends %}`. The second option covers the remaining use cases, with the actual modification of the file itself, removing features or altering their order. In this case, the whole idea is to rewrite the original file, applying the necessary changes.

[13] https://docs.ckan.org/en/2.9/extensions/index.html
[14] https://docs.ckan.org/en/2.9/theming/index.html

Finally, following the same procedure as for open-source plugins already developed, the extension is installed and added to the list of plugins available for use in our instance of CKAN.

4.2.2 CKANEXT-NGSILD-HARVESTER

The purpose of this plugin is to transform the NGSI-LD data obtained through an accessible endpoint into CKAN format and afterwards import it into this data management system.

The official CKAN documentation[13] has been followed for the creation of the extension. This plugin provides a reachable endpoint that will receive information in NGSI-LD format, more specifically, NGSI-LD Entities of type Dataset. This endpoint will act as a callback to the Context Broker subscription and notification service. That is, once a subscription to the Context Broker is generated, it will trigger every time there is a new entity of type Dataset, sending this entity to the established callback.

Once the plugin receives an NGSI-LD Entity via the available endpoint, it transforms the data into the CKAN format in order to import it into the CKAN. This format corresponds to a JSON body with specific fields depending on whether it is an organization, a dataset, or a resource. Figure 8 and Figure 9 show the JSON bodies corresponding to an organization and a dataset, respectively. Both bodies are directly related to the NGSI-LD Entities of the Entity Type CatalogueDCAT-AP and Dataset presented in Section 3.2.1. In the current version of the plugin, resources are generated directly inside the body of the dataset, thus automatically creating a parent-child relationship between the dataset and each of its data representation formats.

Since this extension is intended to perform the import of organizations and datasets into CKAN, it makes use of its API to carry out these operations, which are described in the official CKAN API documentation[15]. To do this, the REST endpoint provided by the CKAN API is used through the POST method on the two resources shown below, the root of the URL being `https://{ckan-host}/`. The payloads of the requests are the corresponding bodies shown in Figure 8 and Figure 9.

- `/api/action/organization_create`: used to create a new organization.
- `/api/action/package_create`: used to create a new dataset.

```
{
    "name": "salted-project",
    "title": "salted-project",
    "description": "This is the description of the SALTED Catalogue",
    "state": "active"
}
```

Figure 8. Organization JSON body

It is important to note that, when making use of the enabled REST endpoint, the requests must include an authentication header with an API TOKEN that includes the necessary permissions for the creation of entities. This API TOKEN is obtained through the CKAN web interface or from the docker virtual environment where the CKAN is deployed. Clearly, this method is not very flexible, being one of the key points in the improvement proposed for the second version of the plugin.

[15] https://docs.ckan.org/en/2.10/api/index.html

```
{
    "name": "Temperature",
    "title": "Temperature",
    "private": "False",
    "author": "SALTED",
    "license_id": "odc-by",
    "notes": "This is the description of the Dataset Temperature",
    "url": "https://salted-project.eu",
    "state": "active",
    "type": "dataset",
    "resources": [
        {
            "name": "JSON: Temperature data in real time",
            "format": "JSON",
            "mimetype": "application/json",
            "url": "http://ckan.tlmat.salted- project.eu:5012/realtime/Temperature.json",
            "download_url": "http://ckan.tlmat.salted- project.eu:5012/realtime/Temperature.json",
            "access_url": "http://ckan.tlmat.salted- project.eu:5012/realtime/Temperature.json",
            "description": "This resource provides real-time information in JSON format.",
            "rights": "PUBLIC",
            "hash": "",
            "created": "21-03-2023"
        },
        {
            "name": "JSON-LD: Temperature data in real time",
            "format": "JSON_LD",
            "mimetype": "application/ld+json",
            "url": "http://ckan.tlmat.salted- project.eu:5012/realtime/Temperature.jsonld",
            "download_url": "http://ckan.tlmat.salted- project.eu:5012/realtime/Temperature.jsonld",
            "access_url": "http://ckan.tlmat.salted- project.eu:5012/realtime/Temperature.jsonld",
            "description": "This resource provides real-time information in JSON-LD format.",
            "rights": "PUBLIC",
            "hash": "",
            "created": "21-03-2023"
        }
    ],
    "tags": [
        {
            "name": "Temperature"
        },
        {
            "name": "real time"
        }
    ],
    "extras": [
        {
            "key": "access_rights",
            "value": "http://publications.europa.eu/resource/authority/access-right/PUBLIC"
        },
        {
            "key": "spatial",
            "value": "http://publications.europa.eu/resource/authority/continent/EUROPE"
        },
        {
            "key": "temporal_start",
            "value": "21-03-2023"
        },
        {
            "key": "theme",
            "value": [
                "http://publications.europa.eu/resource/authority/data-theme/ENVI",
                "http://publications.europa.eu/resource/authority/data-theme/REGI"
            ]
        }
    ],
    "owner_org": "salted-project"
}
```


Figure 9. Dataset JSON body

In line with the work in progress to improve the functional components of the architecture, the operation of this plugin will be affected by further automatization of the mapping of

attributes between both formats, as well as its interaction with the subscription and notification service of the Context Broker. Additionally, the interaction with the CKAN system is also enhanced, performing these operations of creation of organizations and datasets through the internal API, instead of using the available REST endpoint. In this way, authentication is implicit, and it is not necessary to add the HTTP header with the content of the specific TOKEN API. All these improvements and changes will be reflected in the next deliverable of WP3 (D3.2).

4.2.3 Information Retriever

The Information Retriever aims to provide the data and entities stored in the Context Broker in different formats. Hence, it does not only act as a reverse proxy but also transforms and maps the data into multiple representations.

As shown in Figure 6, this module establishes a communication interface between the CKAN and the Context Broker. First, the CKAN Dataset description with the `url` parameter in each resource points to the Retriever endpoint. This endpoint has two different routes:

- `/realtime/<entityType>.<format>`

  This first path is intended for real time data requests, i.e. the last instance recorded in the Scorpio Context Broker. The `<entityType>` field refers to the type of entity being requested and the `<format>` field refers to the format in which information has to be retrieved. For example, `/realtime/Temperature.json` will provide the latest values recorded for the entity type Temperature in JSON format.

- `/temporal/<entityType>.<format>?<temporal_unit>=<value>`
  - `temporal_unit = [“years”, “months”, “weeks”, “days”, “hours”]`

  This second path allows to perform requests with temporal context, i.e. making use of the temporal storage provided by the Context Broker. The `<entityType>` field remains the same as in the previous scenario, as well as the `<format>` field. However, in this case, there is the possibility of adding a query parameter, `<temporal_unit>`, which indicates which time unit has to be used, and, therefore, its value, `<value>`. The possibilities currently deployed for the time unit are those discussed above: years, months, weeks, days and hours. For example, `/temporal/Temperature.json?days=5` will provide the stored values of the last 5 days for the entity type Temperature in JSON format.

Once the Retriever receives the request, it transfers the necessary parameters to an NGSI-LD Query to obtain the values demanded in the first phase. For each of the Information Retriever routes, the transformation is:

- First Retriever route:

  ```
  GET /realtime/<entityType>.<format>
  ```

  Scorpio Request:

  ```
  GET /entities/?type=<entityType>

  Accept: application/<format>

  Link: ‘<“https://raw.githubusercontent.com/SALTED-
        Project/contexts/main/wrapped_contexts/<entityType>-
        context.jsonld”>;rel=“http://www.w3.org/ns/json-ld#context”’
  ```

- Second Retriever route:

```
GET /temporal/<entityType>.<format>?<temporal_unit>=<value>
```

Scorpio Request:

```
GET /temporal/entities/?type=<entityType>&timerel=after&timeAt=<date>
```

- `<date> = datetime.now()-<temporal_unit>`

```
Accept: application/<format>
```

```
Link: ‘<“https://raw.githubusercontent.com/SALTED-
      Project/contexts/main/wrapped_contexts/<entityType>-
      context.jsonld”>;rel=“http://www.w3.org/ns/json-ld#context”’
```

As can be seen in both cases, not only are the variables passed to the query parameters of the Scorpio route, but it is also necessary that the value of both headers presented (Accept and Link) become dynamic. The first header, Accept, specifies the format in which the information needs to be provided, either JSON or JSON-LD (due to limitations of the Context Broker itself), so it can be sent back to the client. On the other hand, the Link header allows the addition of the `@context` of the requested data, in order to retrieve the short names of the attributes, i.e. without the namespaces.

5 CKAN TO EDP CONNECTOR

5.1 ARCHITECTURE OVERVIEW

The last piece of the architecture is the CKAN to EDP connector, which enables the EDP to harvest SALTED data via the CKAN platform. The relevant components can be seen in more detail in Figure 10, including two plugins that allow the transformation of the data into a format compatible with the DCAT-AP specification[16].

The EDP requires an open and accessible endpoint on which to perform data harvesting requests. One of the options presented by the EDP documentation itself is the endpoint established by a server using the Open Archives Initiative - Protocol for Metadata Harvesting (OAI-PMH). This protocol handles the transmission of metadata over the Internet, which is ideal for EDP, as its harvesting is based on the description of the datasets, i.e. the metadata, and not the data itself. Therefore, the deployment of this OAI-PMH server as a CKAN extension (*ckanext-oai-pmh-server*) is a clear decision.

However, this plugin merely acts as a gateway to the dataset descriptions. In order for these metadata to be harvested by the EDP, it also requires them to be represented in a format compatible with the DCAT-AP specification. For this purpose, another extension is developed and deployed (*ckanext-dcat-ap-edp-mqa*), to translate and map the data descriptions from the CKAN format to any other compatible (such as Turtle, JSON-LD or RDF) with this standard and uniform format (DCAT-AP).

The result of this final phase is the harvesting of the SALTED datasets by the EDP, getting them imported, published and disseminated on this EDP.

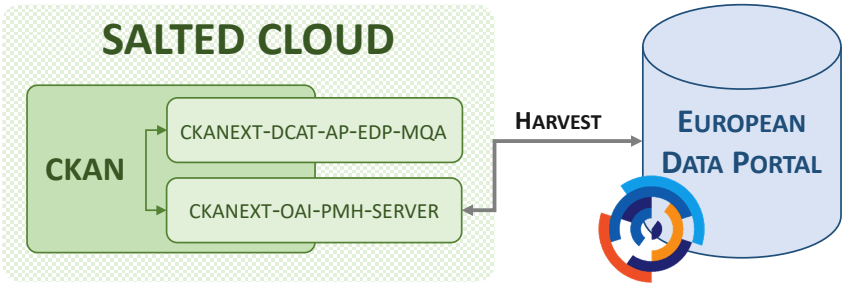

Figure 10. CKAN to EDP connector architecture

5.2 IMPLEMENTATION

In this section, all the elements involved in the CKAN to EDP connector are described in detail, focusing on the two CKAN plugins developed to provide an endpoint accessible by the EDP to make the description of the datasets available in formats compatible with the DCAT-AP specification.

[16] https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/dcat-application-profile-data-portals-europe/release/210

5.2.1 CKANEXT-DCAT-AP-EDP-MQA

The purpose of this plugin is to translate and map the CKAN dataset descriptions from the CKAN format to another one compatible with the DCAT-AP specification.

The official CKAN documentation13[13] has been followed for the creation of the extension. This plugin developed makes use of and requires the installation of the *ckanext-dcat*[17] extension as a starting point, as it is actually fully compatible with DCAT-AP v1.1, and partially compatible with DCAT-AP v2.1.0. Our approach is to extend the mapped vocabulary to make it fully compatible with DCAT-AP v2.1.0 too.

ckanext-dcat

The reference plugin, *ckanext-dcat,* aims to provide RDF representations of the metadata available in the CKAN instance. As it has been outlined above, this plugin version is fully compatible with DCAT-AP v1.1, but only partially compatible with DCAT-AP v2.1.0. Thus, it implements two different schema profiles for both the parsing and serialisation of these two specification versions:

- **euro_dcat_ap_2:** partially compatible with DCAT-AP v2.1.0 (default).
- **euro_dcat_ap:** fully compatible with DCAT-AP v1.1.1.

Moreover, the serialisation formats supported and deployed for their representation are:

- **xml/rdf:** RDF/XML format (`application/rdf+xml`).
- **ttl:** Turtle format (`text/turtle`).
- **n3:** Notation3 format (`text/n3`).
- **jsonld:** JSON-LD format (`application/ld+json`).

These four formats are compliance with the DCAT-AP specification, allowing the consumers to choose which one of them fits better for their purpose.

This extension provides two different endpoints, for both datasets and catalogue, in order to access to their RDF representations. It is important to note that the catalogue encloses the metadata description and compilation of the entire CKAN instance, listing the datasets, organizations, groups and resources available. These endpoints are:

- Dataset:

  ```
  https://{ckan-host}/dataset/{dataset-id}.{format}?profiles={profile}
  ```

- Catalogue:

  ```
  https://{ckan-host}/catalog.{format}?profiles={profile}
  ```

Both cases implement two types of personalisation of the request. First, the format in which to represent the data is indicated, with the `{format}` parameter, by choosing any of the four listed above. The second configuration refers to the schema of the DCAT-AP specification to be used for the representation of the metadata, with the query parameter `{profile}`. In this case, the standard options of the plugin are the ones listed before. However, this list can be extended as needed by defining new profiles, as will be seen later.

[17] https://github.com/ckan/ckanext-dcat

The endpoints' requests can be further configured with other parameters such as pagination or the application of search filters. For more information, see the extension's official documentation17[17].

ckanext-dcat-ap-edp-mqa

This plugin, developed within the SALTED activity, is focused on the extent of the original profile dedicated to the DCAT-AP v2.1.0 specification towards full compatibility. Therefore, using the *ckanext-dcat* extension as a base, a new profile was created which allows the serialisation and parsing of metadata. This profile is:

- **dcat_ap_edp_mqa:** fully compatible with DCAT-AP v2.1.0 and compliant with the EDP Metadata Quality Assurance process.

Using this profile, the metadata describing the datasets hosted in our CKAN instance can be accessed in an RDF data representation format compatible with the desired specification.

As stated already, more in-depth implementation and details of this plugin will be covered in Deliverable 3.2.

5.2.2 CKANEXT-OAI-PMH-SERVER

The OAI-PMH[18] provides an application-independent interoperability framework based on metadata harvesting, allowing the data providers to expose their metadata and the service providers to use that metadata harvested to build value-added services.

This last plugin aims to provide an open endpoint following the principles of OAI-PMH. This protocol enables the descriptions of the datasets hosted in the CKAN instance to be available and accessible for their harvesting. This endpoint will be the one used by the EDP to publish the datasets in its Open Data Portal. The official CKAN documentation13[13] has been followed for the creation of the extension.

Therefore, it is important to know which requests are allowed by a server implementing this protocol (deployed in the extension) and how they are used. The common root path established by this plugin is:

```
https://{ckan-host}/oai?verb=
```

This plugin enables an endpoint outside of the core CKAN context. Based on this URL, the further requests are constructed through the query parameter `verb`. The six requests and their corresponding responses available are defined in the official documentation[19]. They are briefly described below using CKAN terminology.

- **Identify:** used to retrieve information about the CKAN instance.

  `/oai?verb=Identify`

- **ListMetadataFormats:** used to retrieve the metadata formats available. It makes use of other additional and optional argument: *identifier*. This parameter specifies the unique identifier of the item for which the available formats are being requested. Some common responses are: `oai_dc` or `dcat`.

  `/oai?verb=ListMetadataFormats&identifier={item_id}`

[18] http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm
[19] http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm#ProtocolMessages

- **GetRecord:** used to retrieve the metadata from an individual dataset. It makes use of two other parameters: `identifier` and `metadataPrefix`. The first one, `identifier`, specifies the unique identifier of the CKAN dataset, while the second one, `metadataPrefix` determines the format in which the metadata should be represented. The available options for this last parameter can be obtained through the *ListMetadataFormats* request.

  ```
  /oai?verb=GetRecord&identifier={dataset_id}&metadataPrefix={metadata_prefix}
  ```

- **ListRecords:** used to harvest datasets from a CKAN instance. It makes use of several additional arguments: `from`, `until`, `set`, `resumptionToken` and `metadataPrefix`. The first three are optional and used for selective harvesting. The fourth one, `resumptionToken`, is used for pagination and, the last argument, `metadataPrefix`, has been described in the previous request.

  ```
  /oai?verb=ListRecords&metadataPrefix={metadata_prefix}
  ```

- **ListIdentifiers:** is an abbreviated form of *ListRecords*, retrieving only headers rather than datasets. It makes use of the same additional arguments as its extended version: `from`, `until`, `set`, `resumptionToken` and `metadataPrefix`.

  ```
  /oai?verb=ListIdentifiers&metadataPrefix={metadata_prefix}
  ```

- **ListSets:** used to retrieve the organizations structure of the CKAN instance. It makes use of one additional argument: `resumptionToken`. This parameter is used for pagination.

  ```
  /oai?verb=ListSets
  ```

Among all these requests, the EDP will use the *ListRecords* for harvesting, indicating the serialisation with `dcat` as the metadata prefix.

As with the previous plugin, Deliverable 3.2. will go into further detail on the use of this extension and its communication with the EDP.

6 CONCLUSIONS

With this report, we provide the D3.1 deliverable on the SALTED Platform to EDP connector, focusing on the architecture status and the components involved. First, we tackle the issue from a higher-level perspective, discussing the architecture in terms of the most fundamental components and highlighting the need among them to communicate, resulting in the creation of sub-connectors. We have followed a top-down approach while defining the overall architecture as well as the sub-connectors architecture. That is, defining the necessary components along the entire pipeline while keeping in mind what the result should be: publication of SALTED datasets in the EDP that meet certain criteria and requirements.

Upon examining the overall architecture and identifying the necessary sub-connectors, we then move into greater detail on each of these mid-connectors, discussing and describing the key elements required to carry out their development and implementation.

User interaction with the platform is defined through the form for customising the description of the datasets, the Smart Data Models used to register the necessary information in the Scorpio Context Broker are described, and an intermediate component, Information Retriever, is developed and deployed, which acts as a reverse proxy and enables the data stored in the Context Broker to be obtained in different formats (JSON, JSON-LD).

Furthermore, CKAN data management system is employed, on which 4 new extensions or plugins are generated to extend its set of features. These plugins contribute to the CKAN open-source initiative, by developing and implementing them in the most general way possible so that they can be used by the community. These plugins are: *ckanext-salted-template*, which is used to customise the CKAN template; *ckanext-ngsild-harvester*, which is responsible for importing datasets into the system, by mapping the information from NGSI-LD format to CKAN format; *ckanext-dcat-ap-edp-mqa*, which is involved in serialising and parsing the metadata of the datasets hosted in the CKAN in formats compatible with the DCAT-AP specification; and finally, *ckanext-oai-pmh-server*, which ensures that an endpoint is enabled to access the dataset descriptions in a format compatible with the harvesting of metadata required for communication with the EDP.

This deliverable provides a blueprint and establishes the pillars of the Work Package 3 progress. The next deliverable will emphasise the need to fulfil the quality requirements imposed by the EDP and will also discuss their impact on the components of the architecture presented in this document. That is, how these quality constraints affect the development of the elements and the choice of attributes and properties for the description of the information and datasets.

With both the compilation of both deliverables and the actual development and implementation involved, the aim imposed by Work Package 3 will be more than achieved.

The final implementation of the connectors specified (mainly the CKAN to EDP connector), and its interaction with the EDP, in addition to the quality requirements that the metadata needs to meet, will be further described in the next deliverable of this work package (Deliverable 3.2).

7 BIBLIOGRAPHY

[1] SALTED, “D2.2: Report on data modelling and linking,” 2023.

[2] Context Information Management (CIM) ETSI Industry Specification Group (ISG), “NGSI-LD API,” 08 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.06.01_60/gs_CIM009v010601p.pdf. [Accessed 13 03 2023].

[3] European Commission, “Documentation of data.europa.eu (DEU),” [Online]. Available: https://dataeuropa.gitlab.io/data-provider-manual/.

[4] CKAN, “User guide — CKAN 2.10.0 documentation,” 2023. [Online]. Available: https://docs.ckan.org/en/2.10/user-guide.html#what-is-ckan.