EOSC-hub Configuration Management Plan

Authors	Isabella Bierenbaum (KIT), Joao Pina (LIP), Pavel Weber (KIT)
Version:	1.0
Status:	Final
Dissemination Level:	Public
Document Link:	https://documents.egi.eu/document/3703

Abstract

This document provides a description of the configuration management plan applicable to the services of the European Open Science Cloud (EOSC). The plan provides a simple and modular model with clear guidelines and principles on how to establish the EOSC configuration management process. It starts by defining the major configuration items together with their attributes, relationships and other metadata associated with them. At the end, it provides implementation examples for some of the candidate services for EOSC-Core.

EOSC-HUB RECEIVES FUNDING FROM THE EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO. 777536.

COPYRIGHT NOTICE

DOCUMENT LOG

Issue	*Date*	*Comment*	*Author*
v. 0.1	30.09.2020	Main description of the model is ready	Isabella Bierenbaum, Joao Pina, Pavel Weber
v. 0.2	30.10.2020	Relationship types and attributes are defined	Pavel Weber
v. 0.5	15.11.2020	Examples and use cases added	Nicolas Liampotis, Kostas Koumantaros, Themis Zamani, Cyril L'Orphelin, Agnieszka Pulapa, Nadia Tonello
v. 0.9	23.11.2020	Initial version is ready for comments	Pavel Weber
v. 1.0	01.03.2020	Ready for publication	Pavel Weber, Joao Pina

TERMINOLOGY

https://wiki.eosc-hub.eu/display/EOSC/EOSC-hub+Glossary

Terminology/Acronym	*Definition*
CI	Configuration Item
CMDB	Configuration Management Database
CMP	Configuration Management Plan
CMS	Configuration Management System
DMTF	Data Management Task Force
EOSC	European Open Science Cloud
IT	Information Technology
IP	Internet Protocol
MDR	Management Data Repositories
OLA	Operation Level Agreement
SLA	Service Level Agreement
SMS	Service Management System
SPM	Service Portfolio Management
SLA	Service Level Agreement
UID	Unique IDentifier

Contents

1 Introduction

Configuration Management is a backbone process of a Service Management System (SMS) implemented in any advanced IT infrastructure. It aims at providing and maintaining a logical model of all resources included in the Service Management System, their status information, relationships and dependencies, to enable services to be tested, built, rebuilt and deployed consistently.

The Configuration Management Plan (CMP) described in this document provides guidelines to establish the EOSC configuration management process. It aims at defining the major Configuration Items (CIs) together with their attributes, relationships and other metadata associated with them in the context of the EOSC environment. The proposed model can be applied to any EOSC service or with some customisation to other resource types and be implemented in any configuration data repository contributing to the EOSC Configuration Management System (CMS). Nevertheless, the main objective of this document is to provide a CMP for a particular type of EOSC resources namely EOSC-Core services and to facilitate design and development of the core EOSC architecture.

An accurate implementation of the Configuration Management Plan also facilitates a number of tightly related operational processes like Change and Release Management, Incident and Service Request Management as well as Problem Management, and it improves the transparency and management of the IT infrastructure contributing to the technical and other layers of interoperability, defined by the EOSC FAIR Working Group [R1], and the stable delivery of the services in EOSC.

The proposed model considers the complexity of the multi-tenant EOSC federated live environment, its distributed architecture, which is being established by multiple institutions and service organizations from different countries, with rapidly changing pace. It follows a lightweight, modular approach instead of strict definitions for metadata schema of CIs and their relationships.

The goal is to provide simple guidelines and principles for the implementation of Configuration Management, using an unambiguous description of the resources and services, together with their components and relationships, resulting in a scalable Configuration Management System that can be easily adapted and used by all organisations, resource and service providers contributing to EOSC.

2 Background Terminology

The EOSC-hub project builds an IT Service Management System with the focus on a clear, lightweight and pragmatic approach. To achieve this, it relies on the FitSM standards family [R2] as guiding principles, which are adapted when required by the project. Therefore, the terminology used in this document is based on the corresponding FitSM-definitions [R3], which are modified and extended when necessary to describe the Configuration Management Plan. In addition, some definitions are taken from the recently created EOSC Glossary [R4] with the intent to make the Configuration Management Plan, proposed in this document, also applicable at the expected EOSC scale.

2.1 Application

An application is defined as a deployed and running program or module which provides a specific functionality for a user or another application. A typical example of an application is a running instance of a database, which can be used by multiple other applications.

2.2 Building Block

An essential, functional unit or part of the composite service. Services and service components contributing to the composite service are referred to as building blocks.

2.3 Configuration Item

Element that contributes to the delivery of one or more services or service components, and therefore requiring control of its configuration. [R3]

Note 1: CIs can vary widely, from technical components (e.g. computer hardware, network components, software) to non-technical items such as documents (e.g. service level agreements, manuals, policies, license documentation).

Note 2: The data necessary for effective control of a CI is stored in a CI record. Any record can have several versions, representing different states of the CI. In addition to attributes of the CI, the CI record likely includes information on relationships it has with other CIs, service components and services. CI records are stored in a configuration management system.

2.4 Configuration Item Attribute

Each configuration item is characterized by a list of attributes. These attributes contain information about the status (e.g. in the case of a service “beta version”, “deprecated”), contact persons, identifiers, etc, which are required for the description of the CI.

2.5 Configuration Data Store

A collection of CIs, which represents a single entity in a distributed Configuration Management System. There is no strong requirement for a Configuration Data Store to establish and keep relationships between CIs. Instead, in the Configuration Management Database (CMDB) the

preservation of the relationships between CIs becomes a strong requirement compared to the Configuration Data Store.

2.6 Configuration Management Database

Store for data about configuration items (CIs) and relationships between them.

Note: A CMDB is not necessarily a single database covering all configuration items. Instead, it may be composed of multiple physical configuration data stores. In this case of a federated CMDB often the term Configuration Management System is applied.

2.7 Configuration Management System

A set of tools, data, and information that is used to support service configuration management. Typically, a CMS consists of multiple Configuration Data Stores and Configuration Management Databases interconnected with each other in a federated infrastructure. The CMS can also include the operational information about changes, incidents, known errors etc. and it is used by all service management processes.

2.8 Configuration Baseline

The state of a specified set of configuration items at a given point in time [R3].

2.9 Downtime

Downtime is defined as the period or periods of time, when a service or service components are unavailable. A downtime can be of several types (scheduled, non-scheduled) with an associated severity level.

2.10 Manager/Contact

Resource or service manager/contact is a person who is responsible for day-to-day operations and delivery of the service. Typically, this is a technical role used in the description of complex composite services which consist of multiple building blocks. For each building block a manager/contact can be assigned to make a distinction between a general service owner for the whole service and persons responsible for the building blocks.

2.11 Owner

The owner is the responsible and accountable person for a specific resource offered by the provider.

Note: This definition includes the Service Owner definition, where the Service Owner is the responsible and accountable person for a specific IT service offered by a service provider.

2.12 Operations Level Agreement

Documented agreement between a service provider and another part of the service provider's organisation or a federation member to provide a service component or subsidiary service needed to allow provision of services to customers [R3].

2.13 Provider

A Provider is an organisation or federation (or part of an organisation or federation) that manages and delivers a service, service component or any other resource to customers.

Note: In the case of the provider of a service, the term Service Provider might be used. However, it should be noted that this is already included in the more general definition of a provider. Any service can have multiple service providers.

2.14 Resource

Any discrete actor, service, policy, data or infrastructure that can be considered as asset or a constituent part of the EOSC [R4]. In some cases, an owner who controls the access and usage of the resource can be assigned to the resource.

2.15 Release

Set of one or more changes to configuration items (CIs) that are grouped together and deployed as a logical unit [R3].

2.16 Role

Set of responsibilities and connected behaviours or actions collected into a logical unit that can be assigned to an individual or group.

Note: An individual may take over multiple roles.

2.17 Service

Means of delivering value for the end-user by facilitating outcomes the end-user wants to achieve [R4].

2.18 Service Component

Logical part of a service that provides a function enabling or enhancing a service.

Note 1: A service is usually composed of several service components.

Note 2: A service component is usually built from one or more configuration items (CIs).

Note 3: Although a service component underlies one or more services, it usually does not create value for a customer in isolation and is therefore not a service by itself. [R3]

Note 4: A Service is usually composed of several components providing a function or enhancing that service. Usually, service components are not part of the Service Catalogue and so cannot be ordered directly by the users, but they may be composed by several CIs. On the other hand, in the federated environment service components of one service can be integrated into another service following composability and integration use cases. Thus, the services could share the same service components, although for each service component the initial parent service must always be defined.

Under this data model, service components can only be assigned to existing services.

2.19 Service Level Agreement

Documented agreement between a customer and service provider that specifies the service to be provided and the service targets that define how it will be provided. [R3]

2.20 User

A user is a person interacting with the EOSC ecosystem.

3 Service Data Model

A Service Data Model is an essential framework of a Configuration Management Plan, which builds the dependencies between a service, a part of the Service Portfolio/Catalogue and all underlying service assets, which are required to deliver this service to the user. It provides an integrated view and description of any service in the IT infrastructure based on specification of configuration items, their attributes, and relationships between them.

In order to define the service data model, we follow these steps: gather the requirements from the use cases, introduce the general structure of the configuration items and their associated attributes and finally, define the relationships between the CIs and present the CI structure.

3.1 Use cases and requirements to EOSC Configuration Management

In this section, only the major use cases and requirements to the EOSC Configuration Management System will be described considering the initial narrow scope of the EOSC-hub SMS with focus on the EOSC federating layer or EOSC-Core.

Support of the EOSC-Core Configuration

All services included in the EOSC-Core should be accurately described and the information requested by the Service Portfolio Management (SPM) Process about them should be included and maintained in the CMDB, which is operated by the Configuration Management Process. The Configuration Management should facilitate a decision-making process, which could take place at SPM level or at any other governance level with respect to inclusion or exclusion of one or another service in the EOSC-Core. Thus, an interface of the CMS to the Service Portfolio Process should be defined, which will allow to keep the accurate picture of EOSC-Core services up to date.

Support of the EOSC Interoperability Framework

The CMS should support the EOSC Interoperability Framework by delivering accurate up-to-date configuration for EOSC-Core services, their components and relationships to facilitate design and definition of overall federated EOSC architecture, composability and integration of resources. The EOSC-Core service functionality for vertical integration should be accurately described by the CMS and provided by the EOSC-Core service offers for composability and integration with other EOSC resources. All relationships and dependencies established during the integration of the EOSC-Core services with other EOSC resources should be registered in the CMS in order to facilitate the EOSC architecture evaluation and evolution process.

Support of Change and Release Management

All changes managed by the Change Management Process should be reflected in the CMDB for the CIs involved in the changes. For each CI in the CMDB, a list of attributes, for which the history of changes is documented, should be defined. This feature would allow to track the history of the changes for particular attributes of the service like the history of releases or list of change requests, and to control a configuration baseline and CI life cycle.

Support of Incident and Problem Management

Any incoming incident or service request ticket should be assigned to a particular set of CIs associated with the service. The propagation of the incident states should be enabled for all CIs related to the service based on the type of their relationship to the service. For example, if the operational state of the major service, which is reported to have an incident, changes to the “Incident” state, also the states of child-CIs like service components should be automatically set to “Incident”.

It should be possible to link any article in the Knowledge or in the Known Error Database if needed to the corresponding CI.

Support of Service Level Management

Any SLA or OLA defined for the service under Configuration Management should be attached to the corresponding CI. In addition to this, monitoring reports should be, if possible, automatically collected and attached to the service in order to evaluate performance of the service against service targets specified in the SLA or OLA. This use case implies the integration of the CMS with the EOSC monitoring system.

3.2 Configuration Items and their Relationships

3.2.1 Configuration Item

A configuration item is a key component of a Configuration Management Plan, which needs to be accurately defined. It is important to note that depending on the main requirements to the Configuration Management Process the structure of the configuration items has to be agreed. This structure declares the entities that have to be defined as CIs and to be included in the CMP. These requirements depend on the general organisation of the SMS and its objectives at different levels of consideration. For example, at the EOSC federated level, the main focus of the SMS could include services, their functional components, software products with their versions (releases) and used technology. Thus, the service’s local infrastructure like network topology, cluster configuration etc is out of the scope.

Another important aspect is the distinction between a CI and its attributes. We consider an attribute as any single elementary property or key:value pair at the given level of consideration. For example an “IP address” is an attribute of the Service CI, as it doesn’t contain any additional information apart from the address value, while for example Provider is a separate CI since it has multiple attributes like, e.g. Location, Type, etc. (cf. also Section 3.2.3).

Furthermore, the introduction of the CI as modular entity gives flexibility to establish dynamic relationships between different CIs, e.g., Service CI and Provider CI allowing relationships like for example Service X “is provided by” Provider Y (Section 3.2.2).

3.2.2 Relationships between CIs

As was already mentioned in the previous section, the relationship between CIs is the essential property of the Service Data Model, which provides an effective and object-oriented approach to

description of service topology and service compositions. In this section, we define the main types of CI-relationships applicable in the model.

We consider the following CIs as main constituents of the service data model:

- Service
- Service Component (a group of CIs in the current model)
- Infrastructure
- Software
- User
- Provider

It has to be stressed that a limited number of CIs in this model has been chosen for the sake of simplicity. In the given model, the "Service Component" is defined as a **group of CIs**, which, according to the definition in Section 2, are parts of the service and enhance it, but don't create a value by themselves. For example, a service component could be a logical function or module of the main service. Thus, we can consider different types of service components depending on the concrete implementation of the EOSC service or resource.

The Infrastructure CI group could include server, storage, virtual systems, docker containers, any running applications. Depending on the chosen complexity of the CMDB implementation, this analogy can be applied to any of the above CIs, which can become a group or container of different CIs types (sometimes referred as CI classes), instead of a single CI.

By Software CI we refer to any software product or software package with a given release version associated with a service. Again, this definition can be extended to multiple CIs categories e.g. operating systems, databases, software frameworks depending on the requirements of the particular project.

A User CI as a group includes multiple CI types of users, who could be identified by their roles, association to any group, organisation, service etc., for example, a User CI could include researchers, students, resource owners, resource managers etc.

The exact specification of the relationship types should be done during the implementation phase of the CMP. Here, a few typical types of relationships between CIs are proposed:

- includes/part of
    - Strong relationship of **composition** type [R5], which implies the parent-child relationship. The typical example could be a relationship of this type between Service and Service Component CI, meaning that if parent Service is deleted all Service Components which are parts of this Service are also deleted.
- depends on/required for
    - Also strong relationship of **aggregation** type [R5], although it doesn't imply a parent-child relationship between CIs. For example, the CI A which is "required for" CI B could still exist when CI B is deleted. Although this dependency could also imply that the operational state of CI B could be affected, if CI A is degraded.
- installed on

    - A strong relationship type between Software CI and Server CI (a possible member of the group of Infrastructure CIs), meaning that software product with given version is deployed on Server CI.
- connected to
    - Weak relationship between CIs which doesn't imply any strong dependency. Typically, this relationship has **association** type [R5]. There could be some exchange of information between CIs in this relationship, but both CIs have independent lifecycles.
- associated with
    - Weak relationship, which is similar to the previous one and also has association type. Can be used instead of "connected to" and defined exactly during implementation if required.
- provides/provided by
    - Strong relationship of **ownership** type [R5], meaning that CI A provides CI B to the customer, and takes responsibility and accountability for CI B.
- owns/owned by
    - Similar to the previous relationship of **ownership** type, meaning the CI A, e.g. a Person owns a CI B, e.g. Service. This relationship introduces a responsibility and accountability of CI A with respect to CI B.
- uses/used by
    - Weak relationship of **usage** type [R5] which implies the dependency when one CI, e.g. User uses another CI, e.g. Service.

Depending on the topology, many CIs could have multiple relationships to other CIs. The governance rules have to be defined during the implementation in a CMDB, in order to prevent the selection of the relationships between CIs that have no logical meaning. For example, an Infrastructure CI cannot own a Provider CI, etc.

3.2.3 CI structure

A CI record must include all relevant data, such as name, contact information, location, relationships, etc, necessary to describe the CI. In order to properly build a CI record we need to define the structure of the CIs and how to use the relationships between them to obtain the records. We sketch here the general structure of a Service CI, providing the list of all attributes and relationships in order to illustrate the main ideas. The attributes are grouped into major categories:

- CI Attributes Basic Information: Name, UID, Provider
- CI location
- Contact Information
- CI state: Deployment state, Operational status etc.
- Any CI specific Information e.g.: Access type, IP and License
- CI relationship information: e.g. CI "parent/child of" another CI
- CI Attributes operational information: SLA, Incidents, Downtime

This approach, explained in more detail in the next paragraph, is used to build the Service Data Model (Section 3.3).

The CI record is constructed using unique attributes of the CI and reference attributes, which contain external information related to this CI, provided by linked CIs.

For example, a Service has a Provider (a Service Provider) related to it (see Fig.1). Thus, also the Service CI has a Provider CI related to it. This Provider CI contains all information, all attributes, necessary to describe the Provider, like Provider Name, Provider ID, Provider Contact Information, etc. in the context of the given service. These attributes, however, are provider-specific and unique to a Provider CI and are called **unique attributes** of the Provider CI. When a Provider CI is linked to a Service CI, it therefore contains all the provider information for this service, and the provider's attributes become **reference attributes** of the service. They are *not* included *explicitly* in the Service CI's attributes list, since they are not unique attributes of the Service CI, but rather associated to it by relationship.

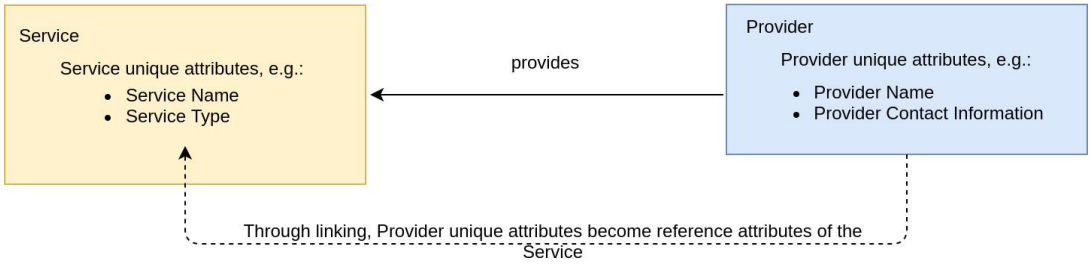


Fig. 1 - Illustration of unique and reference attributes considering the Service/Provider relationship.

Considering the Service CI example from the beginning of this section, we see that the category "CI Attributes Basic Information" for example, contains attributes from different CIs: Name and UID as unique attributes of the Service CI and Provider information as reference attributes from its related Provider CI. A detailed example of the Service CI with its unique and reference attributes is shown in the Annex A.

Note, finally, that within this model, the choice how to deal with this "linking of the CIs" (the building of the relationships), for example how to define which attribute belongs to the unique or reference group of the attributes, or how many CI types should be present in the database scheme, is deliberately left to the implementer of the model.

3.3 The Service Data Model

Figure 2 illustrates the Service Data Model we developed in this Configuration Management Plan. The figure only depicts the CIs which build the main constituents of this model, together with the possible associations (relationships) between these CIs, indicated by arrows[1]. In this figure the User CI has the attribute role of owner.

[1] For a general Reference how to construct such relationship models, see for example [R5].

Explicit choices of relationships between CIs can be taken from Section 3.2.2 and depend on the governance rules chosen for the implementation of the model. An exception has been made for the Owner CI and Provider CI, which have unique relationships towards the remaining CIs, shown by the text “owns” and “provides”. The numbers at both ends of any arrow define the multiplicity of the CI next to the respective number in the following sense: consider for example the pair Service / Owner and the arrow relating them. The Owner CI “Owner*” is marked with an asterisk *, to underline that it is the representation of a User CI for particular CI type “Owner”. A Service is owned by exactly one Owner, as marked by the number “1” at the beginning of the arrow next to the box with the entry “Owner*”. However, an Owner can own several services, shown by the expression “1..*” drawn next to the box of the Service CI at the end of the respective arrow. The symbol “*” represents an indefinite integer greater or equal to 1 and thus the expression “1..*” means “one to many”.

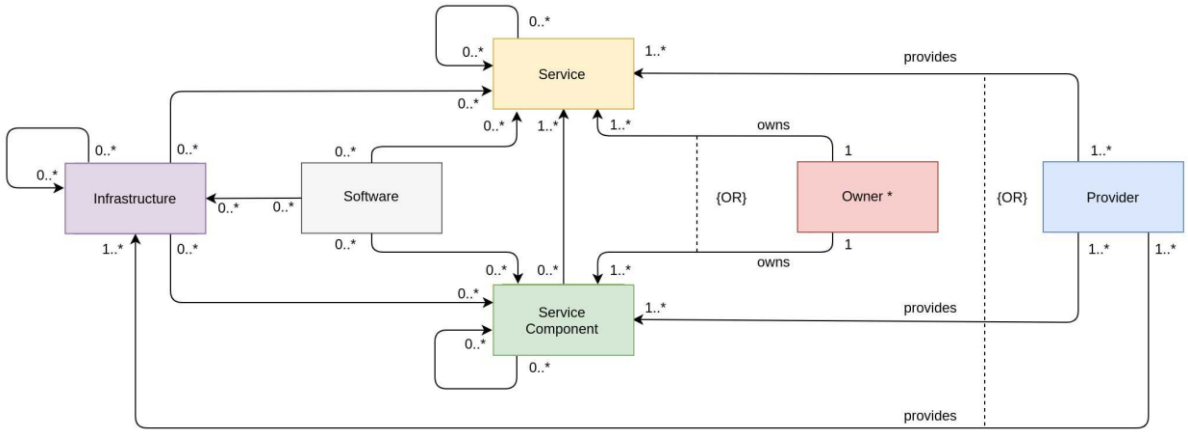


Fig. 2 - General Service Data Model. The arrows and multiplicities between the CIs show possible relationships of the corresponding CIs which have to be chosen appropriately from Section 3.2.2, according to the governance rules of the implementation. An exception is made for the Owner CI and Provider CI, with their unique relationships towards the remaining CIs. The Owner CI is marked with an asterisk *, since it is the representation of a User CI with the attribute role of owner.

Self-loops indicate that the respective CI can be related to other CIs of the same type. For example, a Service can be built of other Services, but it does not have to, since the respective multiplicities start with “0”. The dashed line with the logical expression “{OR}” implies that at least one of the relations, but not necessarily more than one, touched by the dashed line is true. As an example, a Provider is associated to the CIs Service, Service Component and Infrastructure, and it can provide 1..* of each. Already the existence of an association “provides” to one of these CIs (Service, Service Component or Infrastructure) is enough for any entity (organisation, federation etc.) to become a Provider. More than one is possible, in all possible combinations, but not necessary.

It’s important to note that a Service CI is not required to have Service Component CIs as depicted by multiplicity “0..*” meaning that for simple monolithic services it could be enough to have only a single Service CI which fully describes the service with associated Provider and Owner. On the other hand, Service Component CI, if present, must have at least one Service associated with it with a typically strong relationship like “includes/part of”. This requirement doesn’t exclude the situation when the Provider of Service Component is different from the one of the main Service. For simple monolithic services it is usual that the Provider of the Service and Service Components associated with it is the

same organisation. But, for complex distributed services it's typical that the Providers of some Service Components contributing to the Service are different organisations. The same considerations are also valid for Owner CI.

3.4 Service Offering

Although the service offering is not the subject of this configuration management plan, we consider this topic, as it is essential also for EOSC-Core services, which will support the integration of the onboarded community services and resources in the EOSC ecosystem. This integration based on the interoperability guidelines [R6] developed in the EOSC-hub project allows the researchers and research communities to enrich their services with new features, improve service interoperability using the capabilities provided by EOSC-Core. Typical examples of such capabilities are the Authentication and Authorisation Infrastructure, which enables seamless access to research data and services across multiple communities, service monitoring, which allows to improve the performance of the services, the helpdesk, which facilitates communication and support for users of the service etc.

A well-defined offer of the EOSC-Core services with a set of particular configuration parameters is an essential part of the service description and should be easily managed by the service providers, tracked in the SMS and visible for the communities engaged with EOSC.

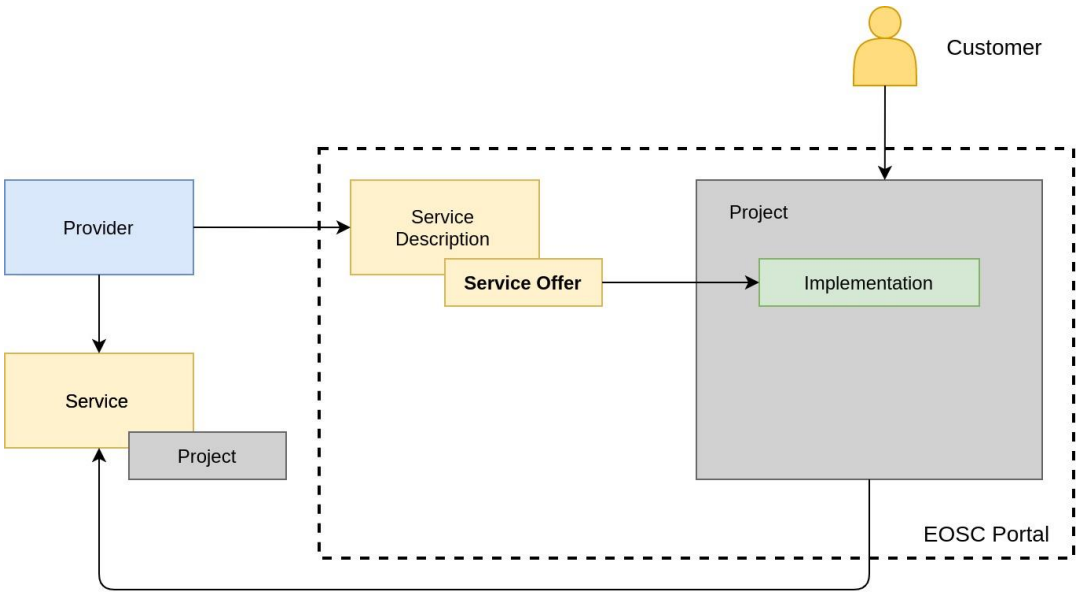


Fig. 3 - Relationship of the service offer to the implemented project in the EOSC Portal and associated service in CMS.

Figure 3 shows a proposed implementation of the relationship between the offer of the service to the project in the EOSC Portal Marketplace and a service, which will be created as the result of project fulfilment and assigned to the provider in CMDB. A provider publishes a description of the service together with a well-defined offer in the EOSC Portal. A customer accepts an offer and adds it to a project. A project is a concept developed in the EOSC Portal Marketplace, which allows to combine

multiple different offers and create orders in one container, namely the project. Upon implementation of the order based on the offer, e.g. the installation of a service for the customer, this new service is created in the CMS and assigned as a service, which is provided by the provider in the scope of the given project. Project information (stored typically in the order management system) is referenced to by the attribute “Projects” of the service. In this way, all services implemented via the EOSC ordering procedure can be easily tracked and integrated in the EOSC Service Management System. This integration will allow for example the assignment of the customer tickets related to the service, the management of SLAs etc.

In some cases, instead of offering a new service instance, a provider offers different capabilities, like for example the Helpdesk which offers a Support Unit for/to the customers. In this case, it is enough to define a new service component instead of new service, as a logical unit of the service, which is related to the implemented order.

Another possible scenario of tracking the acceptance and implementation of EOSC-Core service offers by new EOSC providers during the onboarding process is creation of the relationship directly in the Provider Dashboard of the EOSC Portal. For example, a new EOSC provider decides during the onboarding process to use EOSC Helpdesk taking one of the offers visible in the Provider Dashboard. After the new Provider has accepted the offer a new relationship has to be created in EOSC CMDB between EOSC Helpdesk and onboarded provider or onboarded services. This is more static approach compared to described above, which is based on Marketplace Projects. In this way the relationship between EOSC-Core services and EOSC Providers is established without use of the Projects and requires some development of the Provider Dashboard, which should list the offers provided by EOSC-Core services.

Although in the current document we focus on the offers of the EOSC-Core services, the proposed procedure can be extended to most of EOSC Resources like Data Storage, Compute etc. offered through the EOSC Portal.

3.5 Federated Configuration Management System

The aim of this section is to provide a few considerations which could help to make strategic decisions on implementation of the Configuration Management System. In practise, there are two main approaches for CMS implementation, which are taken into account during the CMS planning phase: federated approach and consolidation one.

The implementation of a Federated Configuration Management System according to federated approach implies a connection of multiple heterogeneous configuration data sources together in order to provide an aggregated view of a resource and its relationship information. A federated CMS allows programmatic (applications) and human access to the widely distributed data, as if they were provided by a single source. The benefits of this federated approach compared to the widely adopted data consolidation approach, where data are stored in a central repository, are:

- Minimisation of data movement to a central repository.
- An aggregated view on the Resource, even if the data are distributed across multiple external repositories.
- Fetching of the data in real-time from federated data sources.

- No need to perform synchronisation with central CMDB in case of changes to the data.

The conceptual architecture of the CMDB federation or so-called “Federating CMDB” published by the Distributed Management Task Force as DMTF standard [R8] defines the main building blocks and basic guidelines for a CMDB in federated scenarios. It describes an aggregated CMDB, which is built on top of multiple so-called Management Data Repositories (MDR) and CMDBs in the federation. The definition of MDR is close to the definition of the Configuration Data Store given in this document. According to [R8], a federated CMDB federates data from all MDRs connected to it, using pull and push modes.

Although this standard provides a fairly complete specification for a federated CMDB, after analysis of the standard, we tend to agree with conclusions provided in the paper [R9], that this specification neither defines an information or data model for a CMDB, nor addresses the question which MDRs to connect and which MDR should be the authoritative source for a particular piece of information and how to consolidate redundant data kept in the MDRs.

In addition, together with clear benefits of the federated approach listed above, also some complications one could meet during implementation have to be mentioned:

- The complexity to achieve a single configuration view taking into account many data sources.
- Difficulties to establish relationships across configuration data sources.
- Lack of tools which implement the DMTF standard, meaning complex integration scenarios across configuration repositories.

According to [R9], a preferable approach for CMS implementation is “top-down” one, which means a creation of an empty information data model, which is gradually filled with elements for the core services. A platform-independent information model is a starting point for setting up the CMDB. After that, the core management processes and procedures need to be analysed. Depending on the results of this analysis a decision process has to be established in order to answer the question which MDRs should be included in the centralized CMDB and which should be maintained according to the federation scenario. We assume this hybrid approach as a reasonable one, and also recommend it for implementation of Configuration Management System in EOSC. Although the difference of EOSC environment compared to one described in the paper has to be considered especially broader scope, heterogeneity of resources, organisations and customers.

We conclude this section with a general remark, that the consolidation approach to the CMDB, where the configuration data are accumulated in a central repository, is not a sustainable solution for EOSC, taking into account its distributed environment, but that also the implementation of pure federated scenarios could be a challenging task, which would require significant efforts and is also not always the best option. Thus, we consider a balanced iterative approach, which has to start with the definition of an information data model and an initial scope. After the model is approved and initial scope is defined, one can proceed with the CMS implementation, based on the use cases and community requirements, considering all benefits provided by both the federated and consolidation approaches.

4 General examples and EOSC Service examples

The main topic around which the Configuration Management Plan and Data Model is built, is the provisioning of a service in the EOSC ecosystem. The Data Model offers the possibility to cover a wide spectrum of service constellations in the EOSC environment with many different relationships, ranging from a service without any sub-services to complex service dependencies. In this section, we show two most basic service constellations and provide a few EOSC service examples.

4.1 General Examples for the Service Data Model

In general, one can imagine service constellations or composition of different complexities. One example might be a service, built by service components, which does not involve other services. Such a service with only *one* Service CI and no sub-services will in the following be called a Single Service. Figure 4 shows an example in analogy to our Service Data Model as depicted in Figure 2. According to the Service Data Model defined there, a Single Service is a service without self-loops.

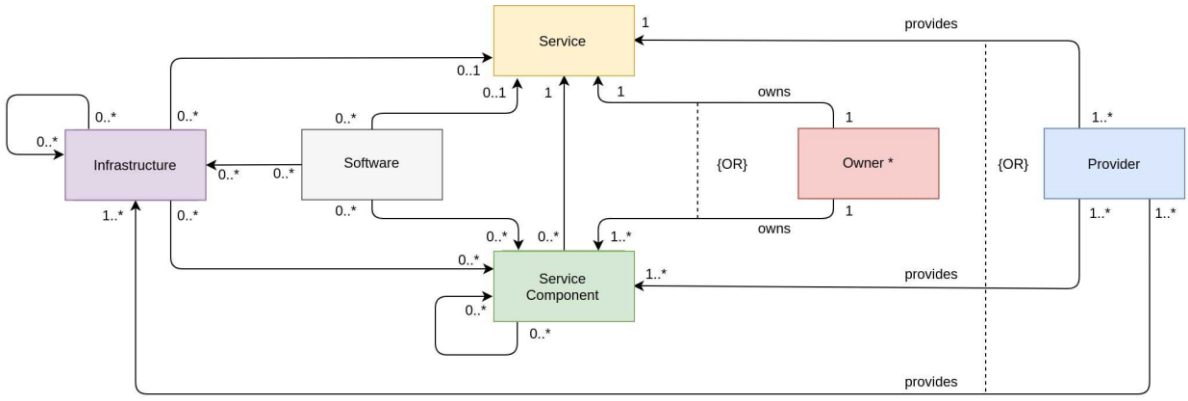


Fig. 4 - Simplified Service Data Model for a Single Service, which is a service, whose setup does not involve any other services.

In many cases the EOSC service providers will not be ready to implement the Service Data Model in full depth. Moreover, the implementation of the model in detail often will be unnecessary at EOSC level and could create more burden to support it and keep up to date, rather than benefits. Thus, the decision on the depth of details of the implementation of the model for one or another EOSC service should be taken by EOSC governance bodies. Due to its flexibility, the model allows us to describe a single service at different levels of complexity and can be easily adopted to the requirements defined by the use cases in the CMS. It is easy to identify a few description schemes:

- Service + Provider + Owner (the minimally required set of CIs, see Figure 5)
- Service + Provider + Owner + Infrastructure
- Service + Provider + Owner + Service Components
- Service + Provider + Owner + Infrastructure + Software
- Service + Provider + Owner + Service Components + Software

Any of these schemes can be applied to EOSC services. For example, for simple, monolithic services the Service Component can be skipped as most likely the description of the Service CI would be sufficient. In some other cases the Infrastructure CI can be omitted if it's not required to track any underlying infrastructure or applications running on the provider side at EOSC level. For the complex services it's recommended to include at least Service Component and Software CIs.

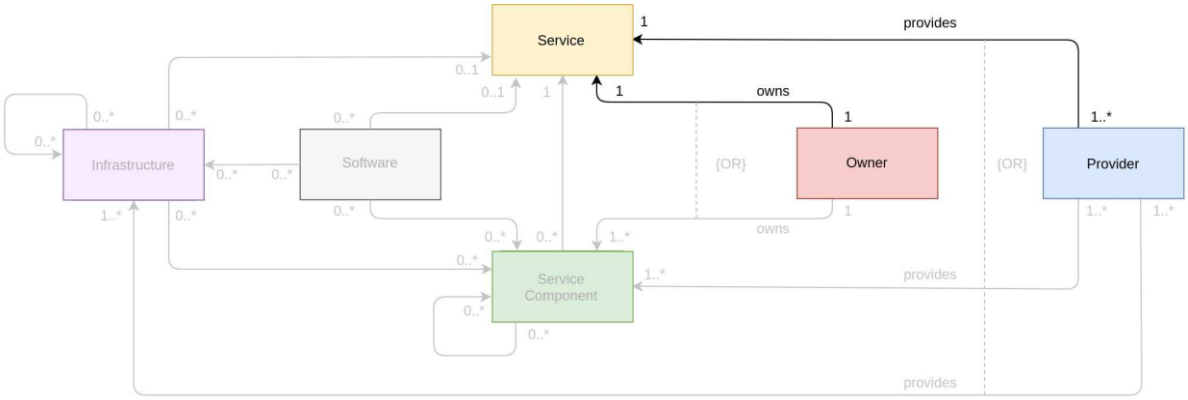


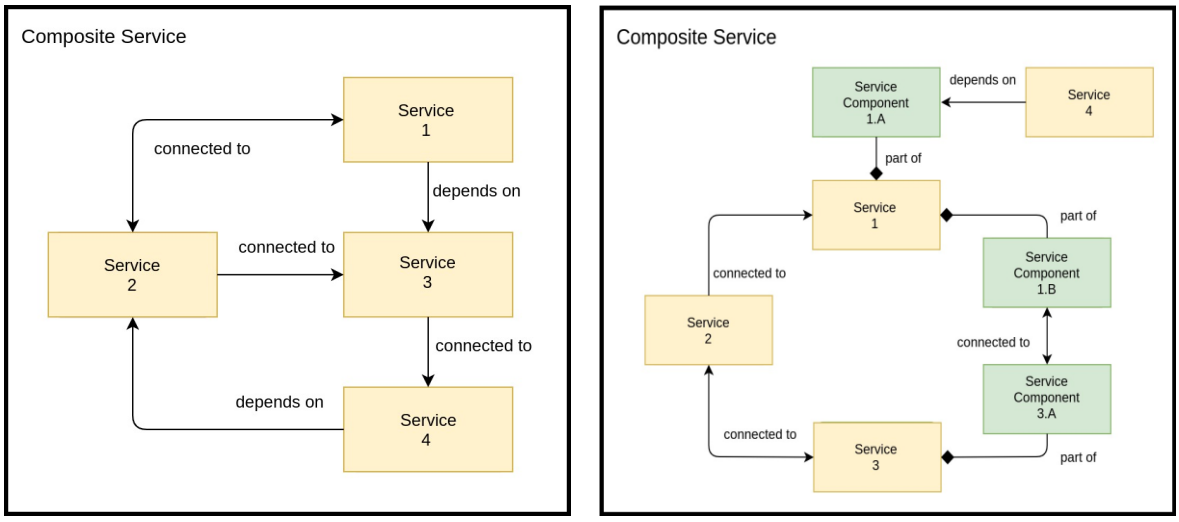
Fig. 5 - Example of a description scheme for a single service which only uses the minimally required set of CIs: the Service CI, Owner CI and Provider CI. The remaining CIs, CI groups respectively, are not used in this description scheme and are therefore only sketched in shaded colours.

Apart from Single Services, one can also encounter Composite Services. A Composite Service is, as the name already indicates, in a very broad sense a service which is a composition of other services and service components, which together serve as building blocks. The services and service components contributing as building blocks to a composite service can differ in their relationship towards EOSC, meaning that they could be in the wider group of EOSC services (e.g. EOSC-Core services, federated/federation services, thematic services at EOSC-Exchange layer etc). The explicit relationship and integration level depend on the particular use case one is looking at and might even change as the EOSC ecosystem evolves. Additionally, the building block services and service components can have all types of relationships between each other, to allow for and cover a broad range of implementation scenarios.

The following Figure 6 shows two general examples for such a Composite Service, only providing the Service and Service Component CIs. The remaining CIs (cf. Figure 2) have been omitted for simplicity, but can be included based on established relationships in a real-life implementation if needed. Figure 6a) shows a Composite Service, which only contains relationships between services. Figure 6b) shows a diagram of another composite service which contains Services and Service Components with possible dependencies between them. There can be service-to-service, service-to-service component and service component-to-service component relationships established. For example, as shown in Figure 6b) Service Component 1.A which is a part of Service 1 is required for Service 4 and Service Component 1.B which is also a part of Service 1 is connected to other Service Component 3.A which is a part of Service 3. Hereby, Service Components can be associated with several services via different relationships. Service 2 and Service 4 don't have any Service Components and have direct relationships to other services or Service Components. It's recommended that any Service

Component should have only one strong relationship of “parent-child” type like “includes/part of” to the service.

It’s also possible that only Service Component, which is a part of another service, but not another service itself, is contributing to the configuration of a Composite Service. In this case only Service Component will be included in the Composite Service leaving the service outside of the borders of Composite Service. It has to be noted that these are only generic examples of possible configuration of Composite Service and they can’t cover all possible scenarios of configuration of Composite Service. The exact configuration of Composite Service can only be defined during implementation.



a) An example of Composite Service which consists of a set of monolithic services and possible relationships between them.

b) An example of Composite Service which consists of a set of monolithic services and their service components and possible relationships between them.

Fig. 6 - Example of a Composite Service. Fig. 6a) shows the Composite Service on a high-level view, only providing its services as building blocks. Fig. 6b) offers a more refined view, where also the connected and connecting service components are depicted. The possible relationship types are described in Section 3.2.2 and no particular choice has been made here. Note, that although only the Service and Service Component CIs are shown, the other CIs according to the chosen description scheme are meant to be included.

4.2 Service Data Model applied to the EOSC-Core candidate services

In the following sections, we will show how the proposed Service Data Model (Section 3.3) describes some of the major candidates which may contribute to the EOSC-Core. Note, that we are not going to present all services, which are included in the candidate list of EOSC-Core. Instead, we demonstrate how complex and distributed services and systems like AAI or EOSC Portal can be described by the model. We focus on some main constituent CIs such as Service, Service Component, Provider for simplicity. We also provide a short description for each service and some service

components, but would like to point out that the functional description of the services is not a focus of the document and is provided for completeness.

The colours of the boxes in the following examples are taken in accordance with Figure 2 and colour-encode of the CIs given there. Frames around CIs group them together in the following sense: an association attached to a frame signifies that the attached association, displayed by an association arrow or by a composition diamond which points into the direction of the text, holds for all elements inside this frame. This is independent of the fact, whether the frame is at the origin or the end of the association. For example, in Fig.7, CCIN2P3 provides each of the Service Components, the Main Component, the Sombo Component and the EOSC Metrics Component, inside the green frame (the “provides'' arrow ends at the green frame). In the same manner, **each** Software CI in the grey frame **is required for each** Service Component inside the green frame, and each Service Component is part of the Operations Portal Service. Note also, that for practical reasons and since it should be clear from the context, we occasionally omit the association text for the owner and provider relationships, or group them into dashed frames (cf. Figures 9 and 11).

4.2.1 Operations Portal

The Operations Portal provides VO (Virtual Organisation) management functions and other capabilities which support the daily operations in the federated infrastructure. The Operations Portal is composed of three main components. A main component, which is the central portal for the EGI operations community, offers a bundle of different capabilities, such as the broadcast tool, VO management facilities, different dashboards that are used to display information about failing monitoring probes and to open tickets to the Resource Centres affected. The second component is the Service Order Management system for EOSC Portal (SOMBO) which tracks and manages orders coming from the EOSC Portal until their fulfilment and delivery of the requested services, and interfaces with third-party Order Management Systems to propagate orders to them. The third component EOSC Metrics component is a dashboard that shows statistics about service registrations in EOSC Portal and service orders.

Figure 7 shows the Service Data Model applied to Operations Portal service. Frames around CIs signify that the attached association holds for all elements of this frame. For example, each Software CI inside the grey frame is required for each Service Component inside the green frame, and each Service Component in the green frame is part of the Operations Portal Service.

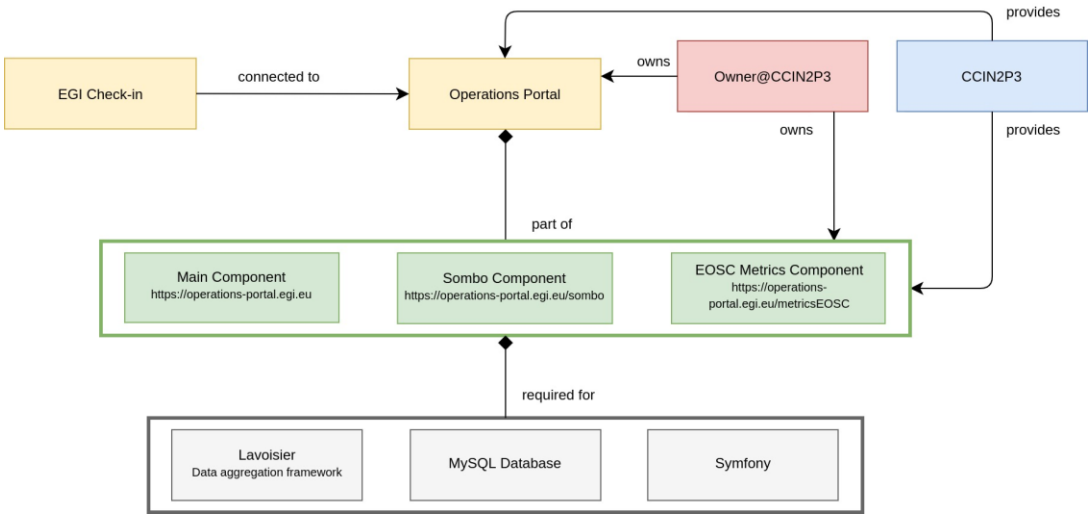


Fig. 7 - Operations Portal diagram.

4.2.2. EOSC Portal

The EOSC Portal is a user-facing platform where productional EOSC services can be onboarded, promoted, discovered, accessed and ordered. A set of functionalities implemented in the Portal supports efficient order management, onboarding of the new services and facilitates the interactions of users with e-infrastructures. The EOSC Portal is composed of three different elements:

- User Component (EOSC-hub Marketplace), which publishes EOSC service catalogue and facilitates the service ordering.
- Provider Component, which manages the onboarding of new EOSC providers and their services.
- Content Component, which is responsible for the management of the web layout of the EOSC Portal.

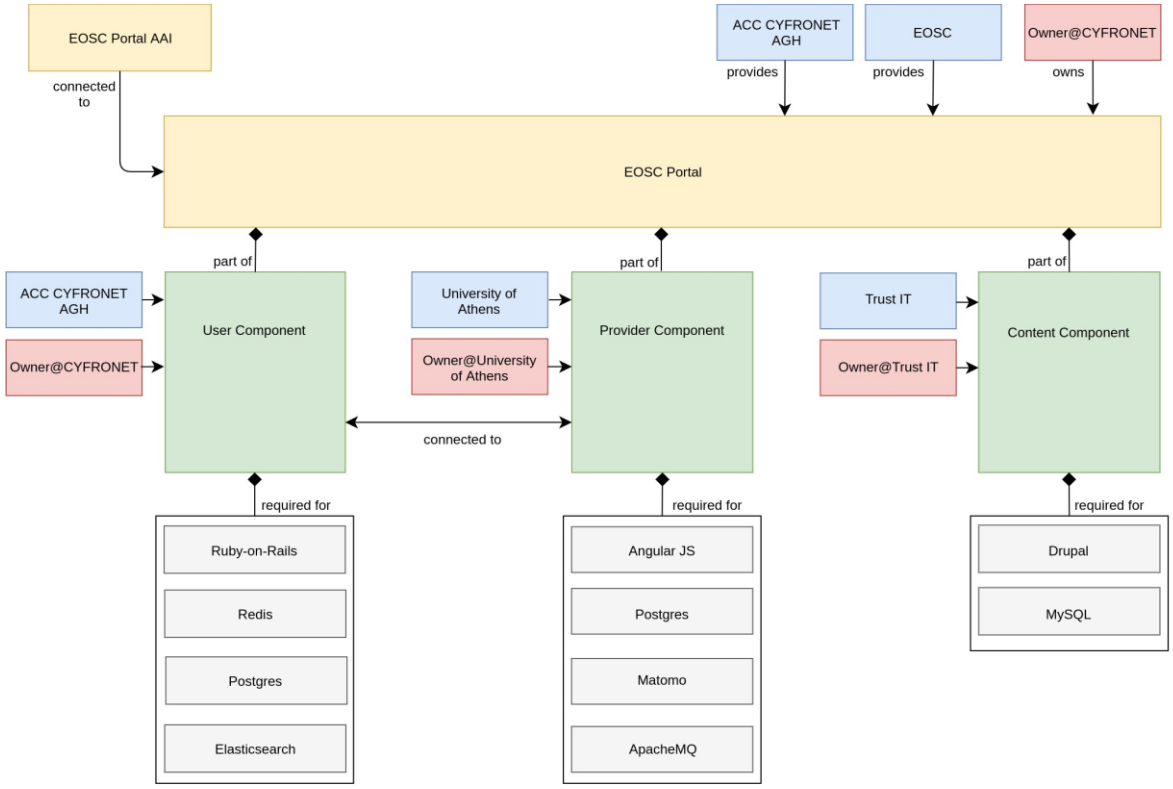


Fig. 8 - EOSC Portal diagram.

The Data Model applied to EOSC Portal is shown in Figure 8. Note, that we omitted the text for the owner and provider relationships for the 3 Service Components for practical reasons. The “connected to”-association between the User Component and Provider Component, which points in both directions, indicates that the two components are interconnected.

4.2.3 EOSC-hub Helpdesk

EOSC-hub helpdesk, based on xGUS, is a single point of contact for all EOSC users for requesting help or for fixing issues. The EOSC-hub helpdesk is integrated with both EUDAT and EGI Helpdesk systems, allowing to handle the management of tickets received on xGUS and assign them either to EUDAT or EGI/WLCG infrastructures.

Figure 9 shows how the Data Model can be applied for the description of the Helpdesk. The EOSC-hub Helpdesk is a composite service, which consists of three different helpdesk systems: EGI/WLCG GGUS, EOSC Helpdesk (candidate) and EUDAT Helpdesk. As shown in Figure 9 the EGI/WLCG GGUS and EOSC Helpdesk are sharing the same software stack. For better readability, we grouped the Owner and Provider CIs into a dashed frame and combined the relationship texts at the corresponding arrows. The “connected to”-association arrows between the EGI/WLCG GGUS and the EOSC Helpdesk, as well as the EOSC Helpdesk and the EUDAT Helpdesk run both ways, since tickets between the respective helpdesk ticketing systems are synchronized such that changes on tickets in one queue triggers the change of the corresponding ticket in the other, and vice versa.

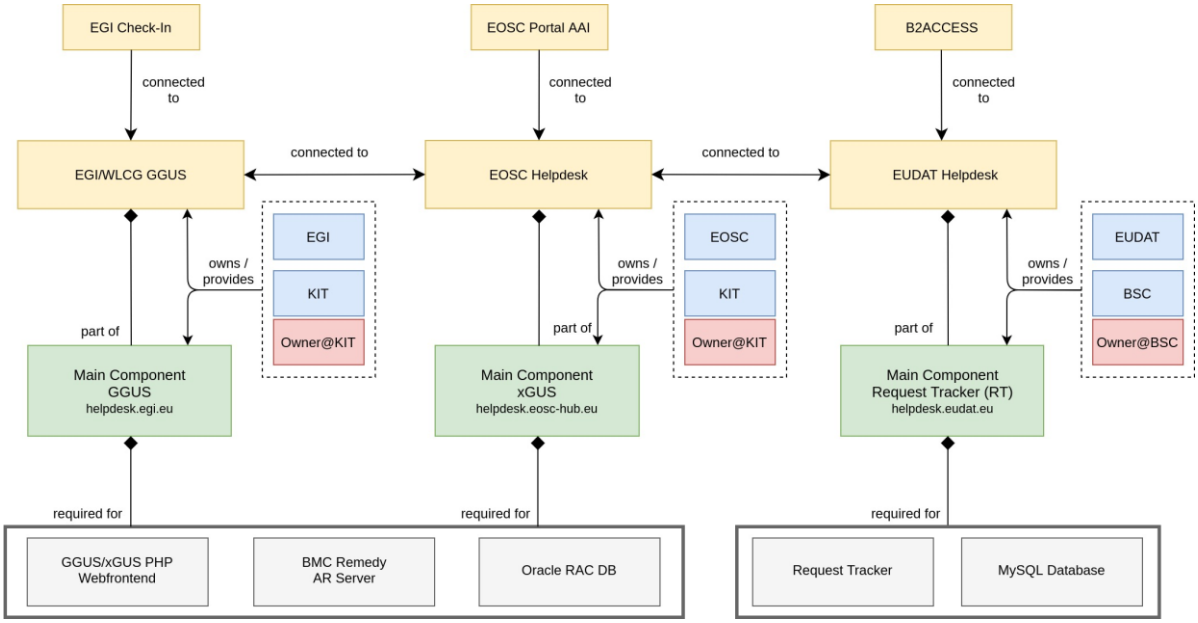


Fig. 9 - EOSC Helpdesk diagram.

4.2.4 EOSC-hub AAI

The EOSC-hub AAI enables seamless access to research data and services in EOSC-hub in a secure and transparent way. It comprises different AAI services acting as IdP-SP-Proxies, namely B2ACCESS, Check-in, eduTEAMS and INDIGO-IAM. These AAI services are connected to eduGAIN as service providers and, at the same time, act as identity providers from the services point of view, in order to allow users to use their credentials from their home organisations for accessing EOSC-hub resources. The suite of EOSC-hub AAI services also includes other services like Perun, an Identity and Access Management system, which can be used by Check-in and eduTEAMS for managing users within organisations and projects, as well as managing access rights to the services. There are also Token Translation Services such as WaTTS and MasterPortal, which provide mechanisms that enable translation between different protocols or technologies. The RCauth.eu service, in particular, is an Online CA that can identify entities on-the-fly based on federated credentials and issue them X.509 credentials in real-time, focussing on conversion from SAML to X.509 standard. Also, EOSC Portal AAI is a part of EOSC-hub AAI infrastructure, connected to multiple Community AAIs to allow researchers to access the underlying services and resources using their community identity, including eduGAIN, with their roles and other authorisation-related information managed by the community.

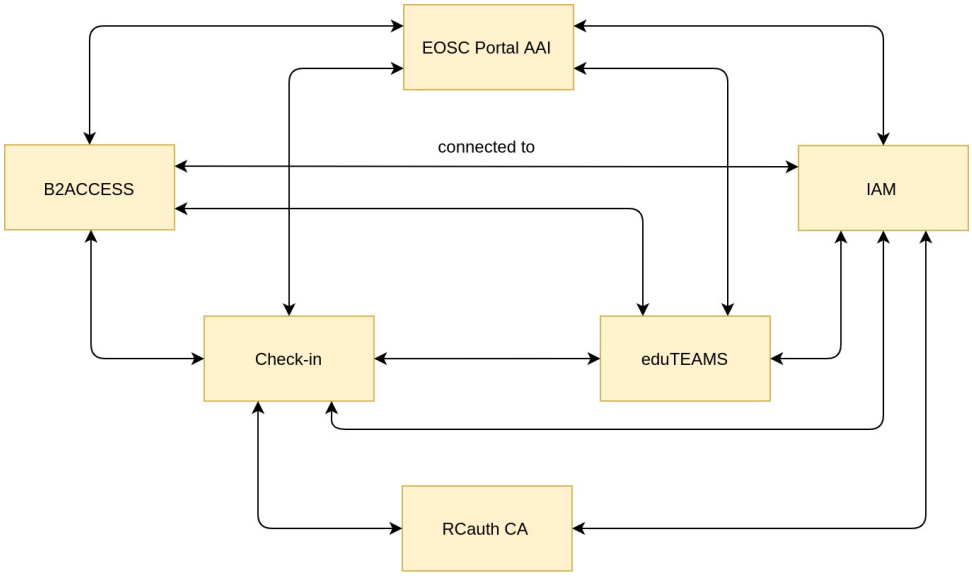


Fig. 10. - EOSC-hub AAI diagram. For practical reasons, the relationship type "connected to" has only been put once, but is applicable to all bi-directional associations' arrows between services.

Figure 10 shows EOSC-hub AAI configuration diagram with focus on the major AAI services and omitting many other AAI services for simplicity. Obviously, EOSC-hub AAI is a composite service with complex relationships and dependencies of different types, which should be accurately managed by EOSC CMDB. Considering the complexity of the AAI composite service we don't pursue the challenging task to depict the underlying structure of the single services even at the level of service components in this diagram and provide a few configuration diagrams of some AAI services in the next sections.

4.2.5 EOSC Portal AAI

The EOSC Portal AAI is part of the infrastructure layer of the EOSC-hub AAI. As an infrastructure proxy, it is connected to multiple Community AAIs to allow researchers to access the underlying services and resources using their community identity, including their roles and other authorisation-related information managed by the community. In addition to the Community AAIs, the EOSC Portal is connected to the upstream home organisation IdPs (e.g. from eduGAIN/social) to enable researchers to access services and resources as members of their home organisation. As shown in Figure 11 the EOSC Portal AAI consists of four main service components: Federation registry, High Availability & Balancing service, IdP/SP proxy and discovery service.

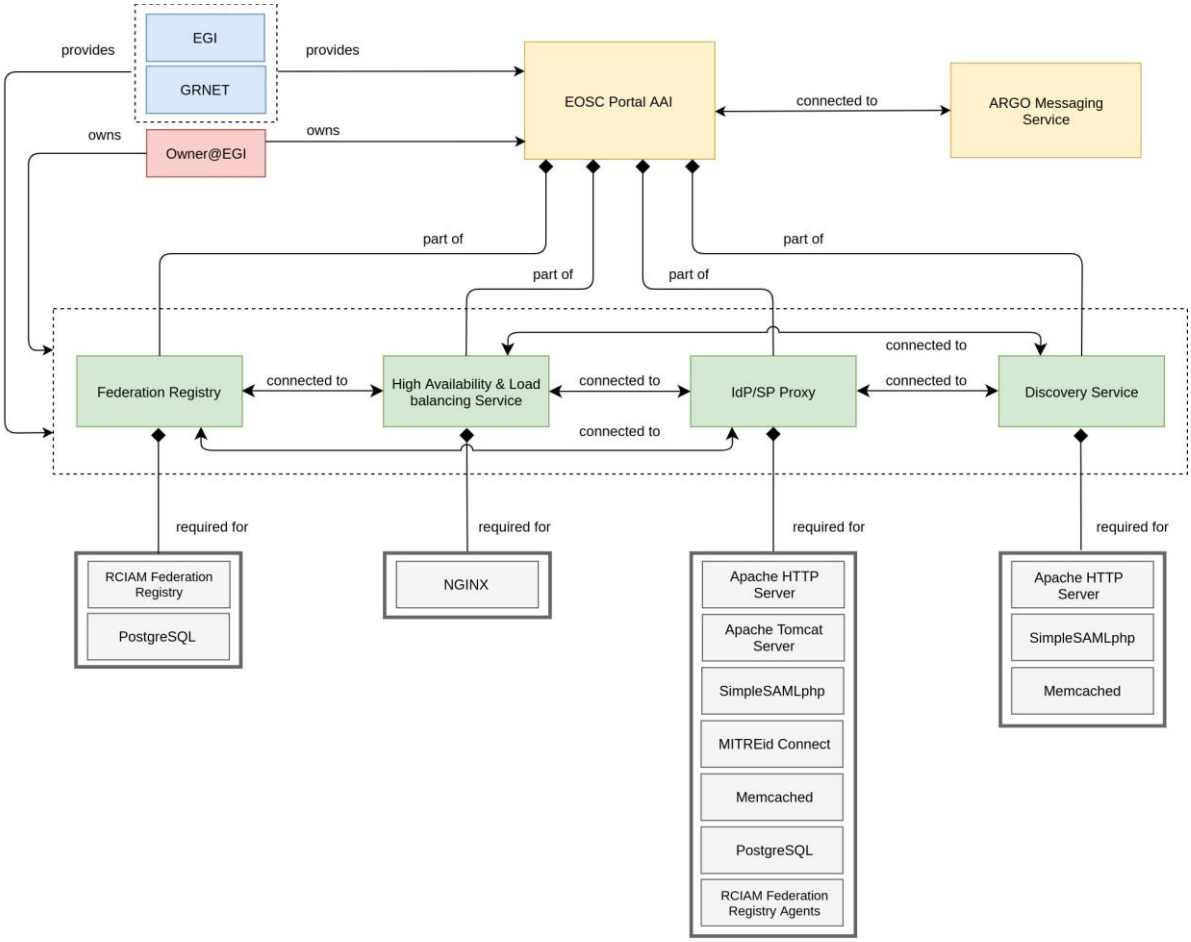


Fig. 11 - EOSC Portal AAI diagram.

4.2.6 RCauth CA

The RCauth CA service is a token translation service (TTS) that can on-the-fly identify entities based on federated credentials and issue to the PKIX credentials in real-time, focussing on converting SAML-to-PKIX. RCAuth CA is a part of the infrastructure layer of the EOSC-Hub AAI.

Without going into functional details (which is not the focus of this document) of the service, it has to be mentioned that RCauth CA is a distributed service provided by several providers as shown in Figure 12. As the infrastructure CIs also omitted for this configuration diagram for simplicity, it doesn't provide information on particular instances and applications running at different sites, instead it provides the information on main functional service components and underlying software required for their implementation.

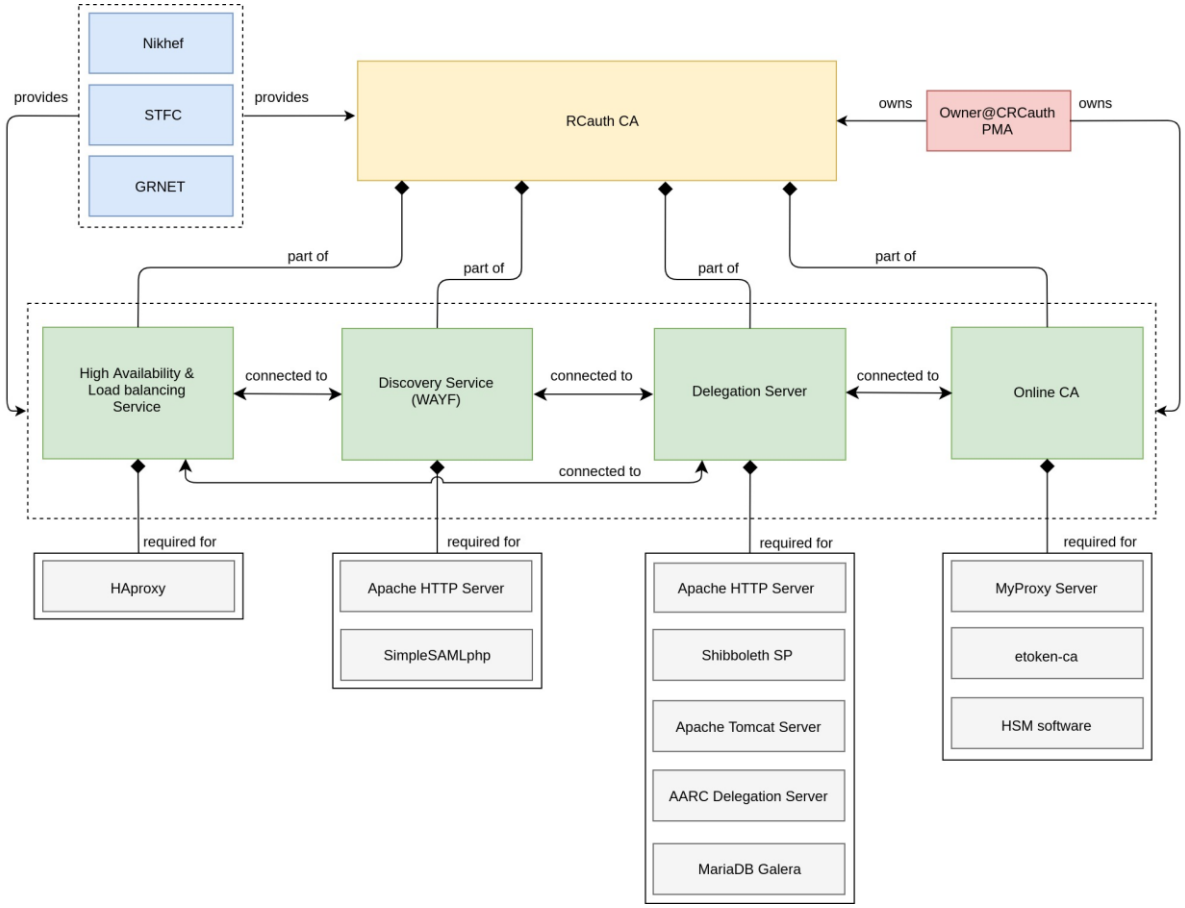


Fig. 12 - RCauth CA diagram.

4.2.7 EGI Check-in

The Check-in is the authentication and user management service for the EGI e-infrastructure. Through Check-in, users are able to authenticate with the credentials provided by the IdP of their Home Organisation (e.g. via eduGAIN), as well as using social identity providers, or other selected external identity providers. Check-in provides an intuitive interface for communities to manage their users and their respective groups, roles and access rights. Check-in is one of major services of the EOSC-hub AAI as shown in Figure 10.

Figure 13 demonstrates the complexity of the service even at the level of service components and the ability of the service data model to also describe complex services with multiple service components and interconnections between them.

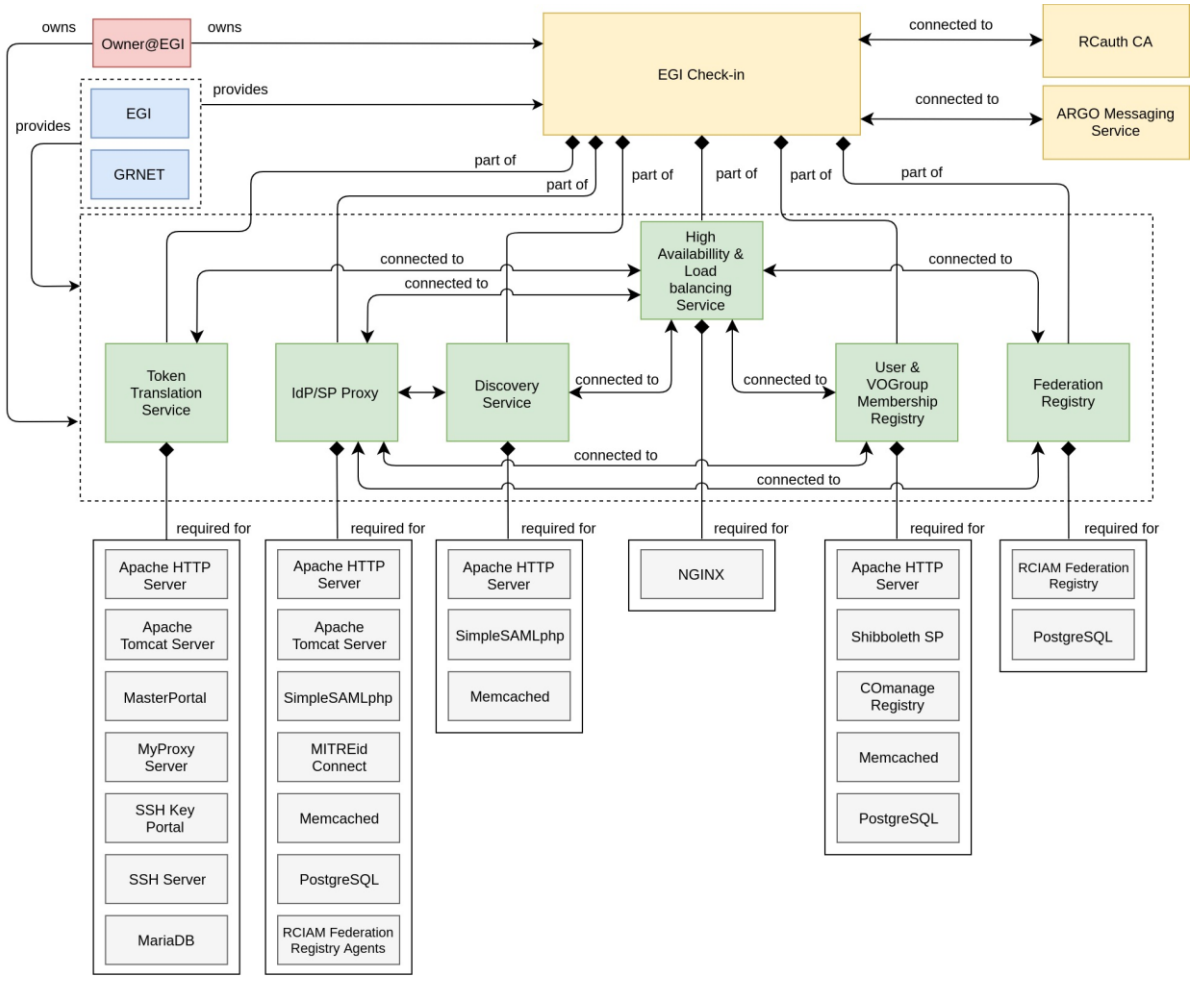


Fig. 13 - EGI Check-in diagram.

4.2.8 ARGO Messaging Service

The ARGO Messaging Service (AMS) is a real-time messaging service that allows you to send and receive messages between independent applications. It is a Publish/Subscribe Service, which implements the Google PubSub protocol. It supports both push and pull message delivery. In push delivery, the Pub/Sub initiates requests to your subscriber application to deliver messages. In pull delivery, the subscription application initiates requests to the Pub/Sub server to retrieve messages.

As shown in Figure 14 the ARGO Messaging Service comprises three service components:

- AUTH-n is an authentication service, which provides the ability for different services to use various authentication mechanisms without having to store additional user info or implement new functionalities.
- API enables Users/Systems to implement message-oriented service using the Publish/Subscribe Model over plain HTTPs
- AMS Metrics provides metrics about the service

Again, for simplicity the configuration diagram of the AMS doesn't include infrastructure CIs, but the description of the service in the CMDB can be extend also to that level of complexity following the propose service data model if needed.

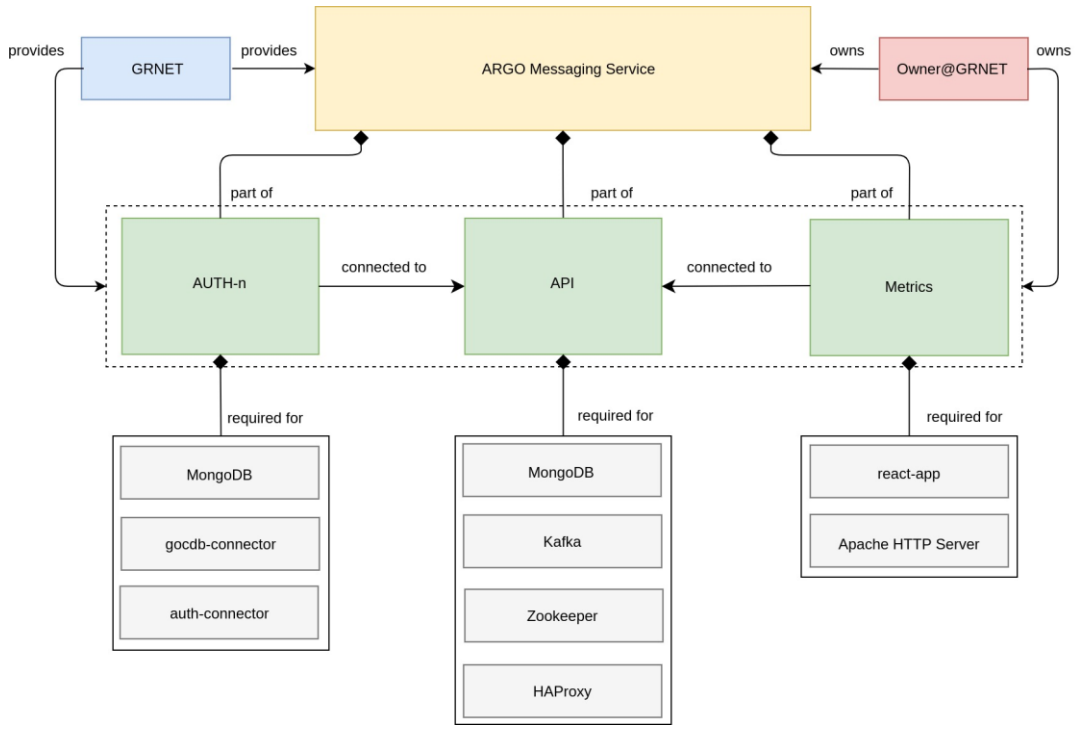


Fig. 14 - ARGO Messaging Service diagram.

4.2.9 ARGO Monitoring Service

ARGO is a flexible and scalable framework for monitoring status, availability and reliability. It provides monitoring of services, visualization of their status, dashboard interfacing, notification system and generation of availability and reliability reports. It consists of five main service components:

- Web UI enables easy access and visualisation of data for end-users. This component is used to store, consolidate and "feed" data into the web application. The global information from the primary and heterogeneous data sources is retrieved by means of the different plugins. The collected information is structured and organized within configuration files in the service and, finally, made available to the web application without the need for any further computations.
- Rest-like HTTP API which allows third parties to gather monitoring data from the system. It supports token-based authentication and authorization with established roles. Results are provided in JSON Format.

- POEM is an administrative component for managing packages, metrics, probes, reports and profiles.
- Analytics Engine component includes computational job definitions for ingesting data, calculates status and availability/reliability. It analyses the monitoring results and sends notifications based on a set of rules, to inform the users (operators, NGIs) about the status of their services.
- Monitoring Engine executes the service checks against the infrastructure and delivers the metric data (probe check results) to the Messaging Service.

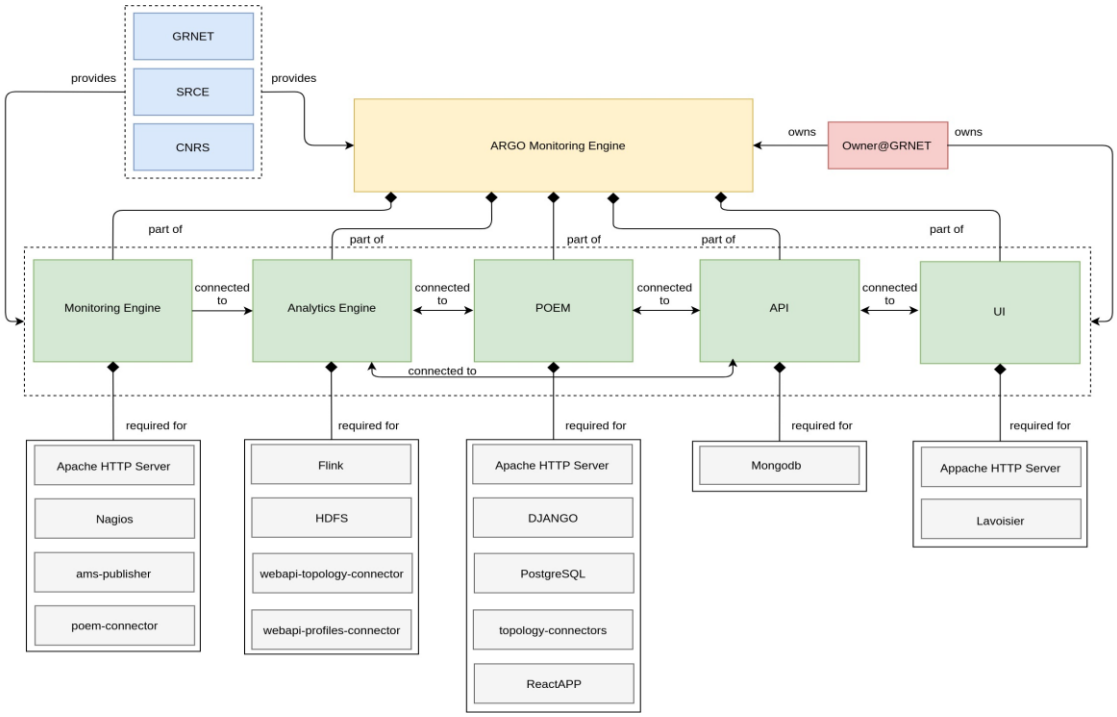


Fig. 15 - ARGO Monitoring Engine diagram.

4.2.10 Order Handling System

The EOSC-hub Order Handling System (OHS) allows customers to access the central EOSC-hub Marketplace, discover services and resources using the intelligent search and filtering mechanism, place an order and track it until its fulfilment and service or resource delivery.

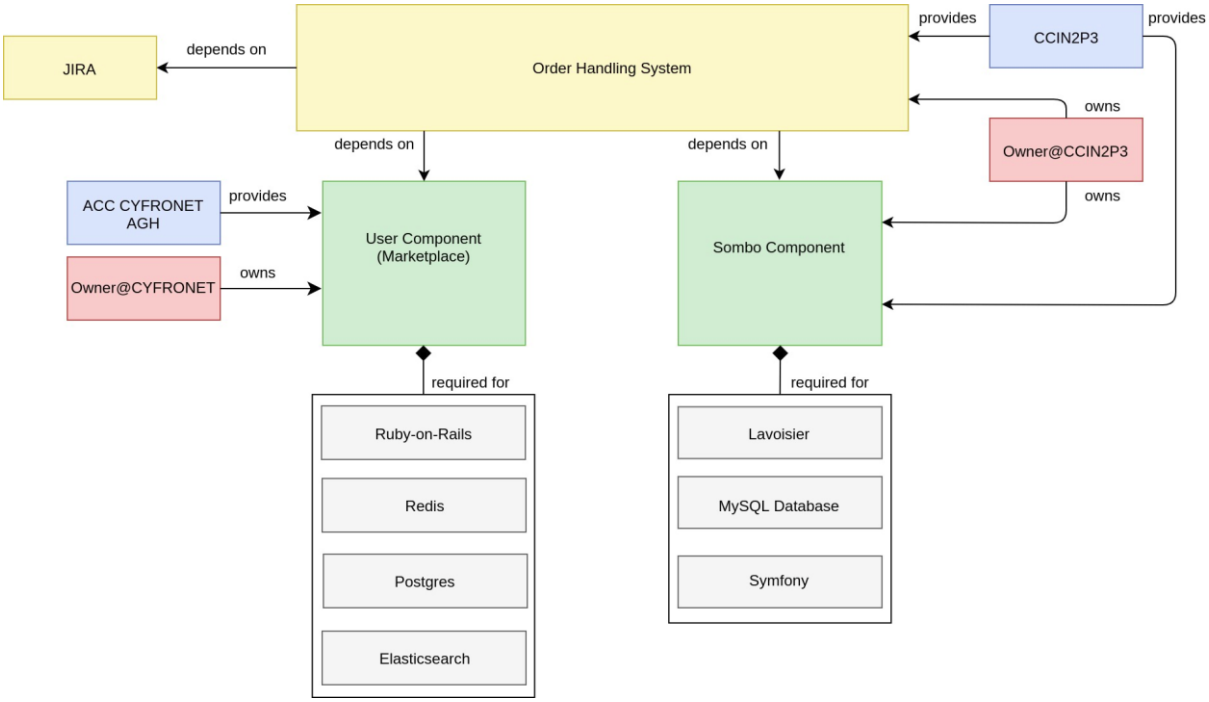


Fig. 16 - Order Handling System diagram.

As shown in Figure 16 the Order Handling System comprises two main components namely User Component (Marketplace) and Service Order Management Back Office (SOMBO) Component. The configuration diagram for OHS follows the configuration of composite service as shown in Figure 6b) and demonstrates the configuration case, for which the composite service is built on service components, which are initially the parts of Operations Portal and EOSC Portal. In this way, the service data model describes the fact of sharing of service components between different services stressing the composability and combined usage of interoperable services and their components.

5 Conclusions and Outlook

In this document, we introduce the configuration management plan which can be applied to the EOSC-Core services. In order to implement the plan, we developed a service data model which is worked out in detail in Chapter 3. The model allows a description of any service in the IT infrastructure based on specifications of configuration items, their attributes and relationships between them. We focused on modularity and flexibility of the model to provide a possibility of its application with some necessary customisation to the wide range of EOSC resources.

To establish the model we followed a few major steps: collection of the requirements from the use cases, introduction of the general structure of the configuration items and their associated attributes, and finally definition of the relationship types between the CIs. We have also provided a few considerations and recommendations for the implementation of a federated Configuration Management System in the EOSC ecosystem.

The examples of application of the proposed data model to some EOSC-Core candidate services are presented in Chapter 4. They demonstrate how the proposed Data Model is able to describe EOSC-Core services, including ones with complex architecture. In this, the configuration management plan is ready to serve as a basis for the further enhancement of a configuration management, which should provide a clear and comprehensive view to EOSC-Core services, facilitate the design of EOSC architecture, the composability of EOSC resources and successful EOSC operation.

Annex A Service record with a list of attributes

In the following section of this annex, a detailed list of the Service CI's record is provided.

The presented list should be taken as just recommendations and can be adapted and adjusted to the needs of the EOSC ecosystem. Of notice, that some of the attributes in the list are taken as reference from current EOSC Resource Profile definitions [R7].

Service

Following the definition given in the Chapter 2 we consider by term "Service" the means of delivering value for the end-user by facilitating outcomes the end-user wants to achieve [R4].

Service attributes

- **General Information**
  - **Name:** Generic Service Name as assigned by the Provider.
  - **ID:** ID of the Service, a unique reference to the Service in the context of the EOSC Portal, according to the EOSC Resource Profile if applicable [R7].
  - **Scope TAGS:** Generic tag/label used to tag this service to specific activities, infrastructures or communities. List of controlled values.
  - **Service category:** Classification according to EOSC or community standards (e.g. Operations & Infrastructure Management). List of controlled values.
  - **Service subcategory:** Classification according to EOSC or community standards (e.g. Monitoring)
  - **Provider [reference attribute] ->Provider CI (multiple)**
  - **EOSC Core Service:** yes/no
  - **Webpage:** Service Public URL
  - **Documentation page:** Public URL

- **Service status**
  - **Status of life cycle:** The status of the service in the overall infrastructure, e.g. planned, test, production, deprecated. List of controlled values.
  - **Scheduled status** (if defined): planned status change with date and new status. This status is typically trigger by creation of Change Request
  - **Operational state:** Incident, Operational, scheduled downtime. Predefined drop-down list.
  - **CI version number:** Version of the CI. On any change a new version of Service CI is created.

- **Geographical location and Language Availability**
  - **Geographical Location:** List of controlled values
  - **Language availability:** List of controlled values

- **Contact Information**
  - **Service Owner [reference attribute]->User CI**
  - **Service Manager [reference attribute]->User CI:** if applicable.
  - **Security Contact [reference attribute]->User CI**
  - **Helpdesk Contact Email:** Helpdesk contact email

- **Service operational information**
  - **Release [reference attribute]->Software CI:** release number of the service related to software stack.
  - **Software Repository [reference attribute]->Software CI**
  - **Monitored:** Is this service monitored by EOSC?
  - **Monitoring URL:** URL of the public monitoring of the service
  - **Registered incidents:** link to all incidents
  - **Change records:** link to all incidents
  - **Helpdesk tickets:** link to all helpdesk tickets (opened/closed)

- **Access and Order Information**
  - **Access Mode:** List of controlled values according to EOSC/community standards
  - **Service Orderable:** Yes/No
  - **Order Type:** Controlled list of values, e.g. (Request/Order required, Open access, Fully open access, other)
  - **Service offers:** List of service offers with offering parameters, e.g. Helpdesk support units for communities, monitoring of the external services
  - **Projects [reference attribute]->Project CI:** Entity which describes the project the service associated with

- **SMS information:**
  - **OLA:** URL for Operation Level Agreement (OLA)
  - **SLA**: URL for Service Level Agreement (OLA)
  - **UA:** URL for the Underpinning Agreement (UA)
  - **Availability Plan:** URL of the Availability Plan
  - **Continuity Plan:** URL of the Continuity Plan
  - **Capacity Plan:** URL of the Capacity Plan
  - **CMDB:** Is this service covered by EOSC SMS? (yes/no)

- **CI relationship information**
  - **Relationships**: e.g. CI “parent/child of”, “connected to” etc. another CI.

References

No	*Description/Link*
R1	EOSC Interoperability Framework https://www.eoscsecretariat.eu/sites/default/files/eosc-interoperability-framework-v1.0.pdf
R2	A free standard for lightweight ITSM https://www.fitsm.eu/
R3	FitSM-0 https://www.fitsm.eu/download/280/
R4	EOSC Glossary: https://docs.google.com/document/d/1zcF95LChshSCv1biqS-AWG12VRyRyzZ5SKyMF5cyk3k/edit#heading=h.xankr3awq6va
R5	https://www.uml-diagrams.org/
R6	https://www.eosc-hub.eu/technical-documentation
R7	https://www.eosc-portal.eu/sites/default/files/EOSC-Profiles-v3.00.pdf
R8	https://www.dmtf.org/sites/default/files/standards/documents/DSP0252_1.0.1_0.pdf
R9	Knittl, Silvia ,Brenner, Michael,Gillmeister, Markus 2010/06/01 Replace or Integrate? - Decision Support for Building a Federated Configuration Management Database https://www.researchgate.net/publication/280293790_Replace_or_Integrate_-_Decision_Support_for_Building_a_Federated_Configuration_Management_Database