Short Abstract
This deliverable is the functional specification of vf-OS. The functional specification is the primary document detailing the functionality provided by vf-OS. The document is structured following the components identified in the global architecture of vf-OS. This document will be the basis for the technical specification, along with the development efforts in the main technical workpackages.
Document Status

<table>
<thead>
<tr>
<th>Deliverable Lead</th>
<th>Raúl Poler, UPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Reviewer 1</td>
<td>Rebecca Campbell, ICE</td>
</tr>
<tr>
<td>Internal Reviewer 2</td>
<td>Stuart Campbell, ICE</td>
</tr>
<tr>
<td>Internal Reviewer 3</td>
<td>Stuart Campbell, ICE</td>
</tr>
<tr>
<td>Type</td>
<td>Deliverable</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP2: Virtual Factory Operating System Architecture</td>
</tr>
<tr>
<td>ID</td>
<td>D2.2: Functional Specifications &amp; Mockups</td>
</tr>
<tr>
<td>Due Date</td>
<td>2017-06</td>
</tr>
<tr>
<td>Delivery Date</td>
<td>2017-07</td>
</tr>
<tr>
<td>Status</td>
<td>EU Approved</td>
</tr>
</tbody>
</table>

History

See Annex B

Status

This deliverable is subject to final acceptance by the European Commission.

Further Information

www.vf-OS.eu

Disclaimer

The views represented in this document only reflect the views of the authors and not the views of the European Union. The European Union is not liable for any use that may be made of the information contained in this document.

Furthermore, the information is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user of the information uses it at its sole risk and liability.
Executive Summary

This document is the vf-OS functional specification and it describes how the vf-OS platform will work from the user’s perspective. Therefore, it is the goal of this document, grounded by the vf-OS D1.1 vision consensus, to explain how the vf-OS platform will provide functionality to the different vf-OS users. In order to do this, the document considers the components identified in D2.1 Global Architecture of vf-OS and analyses how every component offers functionality from the perspective of the subject using that component. The aggregation of the functionality provided by each component provides a rich functionality addressing the business and software requirements identified in D1.5 Requirements Specification.

The document explains, for each component of the vf-OS global architecture:

- The functionality and behaviour that the component offers to its users
- The flow of actions within a component that is carried out to satisfy its functionality
- The mockups of the user interfaces that will be the basis of the interactions with different vf-OS users (Software Developers, Manufacturing and Logistic Providers, and Manufacturing and Logistic Users)

Finally, the document addresses how the vf-OS platform with the current set of components and functionality, covers the functional requirements previously identified in the vf-OS Requirements Specification. An annexed Excel file covers the mapping of those functional requirements and how the user stories (ie fine-grained functionality of vf-OS) address them.
# Table of Contents

0 Introduction .................................................................................................................. 1
1 Context .......................................................................................................................... 5
2 Functional Specification Overview ............................................................................. 7
3 Environment Component ............................................................................................. 9
   3.1 vf-OS Platform ........................................................................................................ 9
      3.1.1 Behaviour and Functionality ...................................................................... 9
      3.1.2 UI mockups and Sequence Diagrams ....................................................... 14
4 Application Development (Design) Component ....................................................... 35
   4.1 OAK Toolkit ......................................................................................................... 35
      4.1.1 OAK SDK ..................................................................................................... 35
      4.1.2 OAK Studio ............................................................................................... 47
      4.1.3 Frontend Environment ............................................................................. 59
      4.1.4 Process Enabler Designer ................................................................. 73
      4.1.5 Data Mapping ......................................................................................... 89
   4.2 Engagement ......................................................................................................... 112
      4.2.1 Developer Engagement Hub ............................................................... 112
5 Application Services and Middleware (Runtime) Components ............................... 124
   5.1 Middleware ......................................................................................................... 124
      5.1.1 Process Enabler Execution ................................................................. 124
      5.1.2 Messaging .............................................................................................. 132
      5.1.3 Publish/Subscribe ............................................................................... 145
   5.2 Data Management ............................................................................................... 156
      5.2.1 Data Storage .......................................................................................... 156
      5.2.2 Data Transformation ......................................................................... 170
      5.2.3 Data Analytics ...................................................................................... 176
   5.3 I/O Toolkit .......................................................................................................... 215
      5.3.1 Enablers Framework ........................................................................... 215
      5.3.2 Drivers .................................................................................................... 232
      5.3.3 APIs ........................................................................................................ 250
      5.3.4 External Service Provision ............................................................... 270
   5.4 Control .................................................................................................................. 277
      5.4.1 System Dashboard ............................................................................... 277
6 Application-Deployment (Use) Components .............................................................. 292
   6.1 Marketplace Services ......................................................................................... 292
      6.1.1 vf-OS Marketplace ............................................................................. 292
   6.2 vf-OS Assets ....................................................................................................... 328
      6.2.1 vf-OS Enablers ...................................................................................... 328
      6.2.2 FI-WARE Manufacturing Enablers ..................................................... 336
      6.2.3 FI-WARE Generic Enablers ............................................................... 349
7 Compliance with software requirements .................................................................... 357
8 Potential Risks and Open Issues .............................................................................. 358
   8.1 Potential Risks .................................................................................................... 358
   8.2 Open Issues ....................................................................................................... 359
9 FAQs ........................................................................................................................... 360
10 Conclusions .............................................................................................................. 361
0 Introduction

0.1 vf-OS Project Overview

vf-OS – virtual factory Open Operating System – is a project funded by the H2020 Framework Programme of the European Commission under Grant Agreement 723710 and conducted in the period October 2016 until August 2019. It engages 14 partners (Users, Technology Providers, Consultants and Research Institutes) from 7 countries with a total budget of circa 7.5M€. Further information can be found at www.vf-OS.eu.

The World is facing the fourth industrial revolution based on ICT, specifically architectures and services, as key innovation drivers for manufacturing companies. Traditional factories will increasingly be transformed into smart digital manufacturing environments but currently the full potential for ICT in manufacturing is far from being fully exploited. Factories are complex systems of systems and there is a need to develop a platform on which future manufacturing applications can be built. Examples of platforms exist in some industrial sectors but there is a lack of cross cutting platforms based on open standards for creating an ecosystem for cooperative innovation. Innovative open platforms to attract talent from solution developers and to provide accessible manufacturing smart applications to European SMEs are examples of the kind of solutions being sought.

The goal of vf-OS is to develop an Open Operating System for Virtual Factories composed of a kernel, application programming interface, and middleware specifically designed for the factory of the future. An Open Applications Development Kit (OAK) will be provided to software developers for deploying Manufacturing Smart Applications for industrial users, using the vf-OS Manufacturing Applications Store all operated through a Virtual Factory Platform.

The Virtual Factory Platform is an economical multi-sided market platform with the aim of creating value by enabling interactions between four customer groups:

- **Software Developers** (independent or within individual manufacturers) which will build Manufacturing Apps either through innovation or from manufacturing user demand
- **Manufacturing and Logistics Users** which will explore the marketplace for already created solutions, ready to be run on the vf-OS
- **Manufacturing and Logistics Solution Providers** which will provide ICT interfaces and manufacturing connections
- **Service Providers** (vf-OS innovators and third parties) will make available services (hosting, storage, connected cloud services, etc.) including those based on developed solutions
The Virtual Factory Platform will provide a range of services to the connected factory of the future to integrate better manufacturing and logistics processes. The Manufacturing Applications Store will be open to software developers who, using the free Open Applications Development Kit provided, will be able to quickly develop and deploy smart applications to enable and optimise communication and collaboration among supply networks of all manufacturing sectors in all the stages manufacturing and logistic processes.

vf-OS aims to become the reference system software for managing factory related computer hardware and software resources and providing common services for factory computational programs. This operating system will be the component of the system software in a real factory system where all factory application programs will run.

0.2 Deliverable Purpose and Scope

The purpose of this document, D2.2 Functional Specifications and mockups, is to describe the functionality covered by the vf-OS platform through the vf-OS components. As such, the deliverable considers the components identified on the global architecture definition (D2.1) and describes for each component:

- Component definition with a clear description of the purpose and scope
- Diagram describing component’s internals work for satisfying functionality
- The mockups of the user interfaces
- Interaction of the component with other components

Finally, the document analyses’ how the requirements identified on (D1.5) are covered by different functionality identified on the different components, making sure that the functionality covers all the user demands.

0.3 Target Audience

This document is confidential and is aimed at the project partners, individuals in those organisations, the EU, and EU Reviewers to gain insight on the architecture of the vf-OS System and the technical RTD work.

0.4 Deliverable Context

This document is one of the cornerstones for establishing the research, and development baseline for the project. Its relationship to other documents is as follows noting that some are used as a basis and others will derive from this document:

- **Vision Consensus (D1.1):** Report providing the reference guide for vf-OS to be used by the partners to stay focused on the main ideas and goals of the project. Available now
- **Requirements Specification (D1.5):** Report documenting the requirements of vf-OS divided into strategic, high level and technical requirements. Available now
- **Global Architecture Definition (D2.1):** Report specifying the architecture of vf-OS. The document identifies the main components of the vf-OS platform and describes them. Available now
- **Technical Specification (D2.3):** Report providing concrete interfaces between vf-OS software components, protocols and class/packages structures, including definitions of methods and error handling together with the data models and data schemas.

- **Holistic Security and Privacy Concept (D2.4):** Report developing on the privacy and security handling. It will consider the components and details from D2.2 and D2.3 when providing a non-intrusive security solution for vf-OS. Derived in parallel with the Functional Specification

### 0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1:** Context: Positions the functional specification in the context of previous deliverables
- **Section 2:** Functional Specification Overview: Describes the functional specifications of vf-OS’ environment component
- **Section 3:** Environment Component: Describes the functional specifications of Design Time building block components
- **Section 4:** Application Development (Design) Component: Describes the functional specifications of Runtime building block components
- **Section 5:** Application Services and Middleware (Runtime) Components: Describes the functional specification of the middleware and services components
- **Section 6:** Application-Deployment (Use) Components: Describes the functional specifications of User building block components
- **Section 7:** Compliance with software requirements: Describes the coverage of user requirements by functionalities provided by vf-OS components
- **Section 8:** Potential Risks and Open Issues: Provides an overview of the potential risks, limitations and open issues of the functional specification and the choices made in this document
- **Section 9:** FAQs: A FAQ aiming to solve specific issues
- **Section 10:** Conclusions: Conclusions of the document

- Annexes:
  - Annex A: Document History
  - Annex B: Reference
  - Appendix I: Software requirements vs user stories mapping (excel)

### 0.6 Document Status

This document is listed in the Description of Action as “confidential” since it provides information for the development of the vf-OS system by the vf-OS partners.

### 0.7 Document Dependencies

This document has no preceding documents or further iterations.
0.8  **Glossary and Abbreviations**

A definition of common terms related to vf-OS, as well as a list of abbreviations, is available in the supplementary and separate document “vf-OS Glossary and Abbreviations”.

Further information can be found at [http://www.vf-OS.eu/glossary](http://www.vf-OS.eu/glossary)

0.9  **External Annexes and Supporting Documents**

Annexes and Supporting Documents:

- External Annex I: Software requirements vs user stories mapping (excel)

0.10  **Reading Notes**

None
1 Context

The vf-OS Project will develop an Open Operating System for Virtual Factories (vf-OS) (Figure 1) specifically designed for the factory of the future. Furthermore, an Open Applications Development Kit (vf-OAK) will be provided to software developers for developing vApps for industrial users, using the vf-Store at the vf-Platform to spread throughout the manufacturing domain.

Figure 1: vf-OS Big Picture

The following diagram depicts the context, the process, and the main influencing sources in the definition of the functional specification and its maturation process. Additionally, the diagram describes the next steps towards the implementation of vf-OS.
The process of defining the functional specification and the mockups begins with the Vision Consensus where the main components and ground of the vf-OS platform were established and with the global architecture definition where a first attempt to depict the main components of vf-OS was done. The vf-OS components must provide functionality satisfying the software requirements provided in requirements specification.

Figure 2: Context Diagram and Next Steps
## 2 Functional Specification Overview

The current document is the vf-OS functional specification and it describes how the vf-OS platform will work from the user’s perspective. As any functional specification, this document does not deal with the technical aspects on how the software is implemented. Instead, it explains the features provided by the software, specifying its features and interactions, including screens, menus, dialogs, etc.

Technical aspects associated with the implementation are included in the technical specification (D2.3).

This functional specification is divided into the specific functionality and interactions that the user has with each vf-OS component, which was identified in the global architecture definition (D2.1).

The functional analysis per each component is made from three perspectives (Figure 3):

- **Behaviour and Functionality**: Containing a story map with the features and functionality offered and the user stories that need to be developed to implement that functionality
- **UI mock-ups and sequence diagrams**: Describing, for each functionality, the interactions of the component with the user or with other vf-OS components
- **Interaction descriptions**: describing for each component the set of interactions that it has with other vf-OS components and users and describing the exchange of information flows that will be critical for a unified vf-OS platform

![Figure 3: OAK SDK Story Map](image)

In terms of the behaviour and functionality, story maps describe the functionality on different levels of aggregation / abstraction. In order to define the story maps interactively, the online tool “StoriesOnBoard” has been used in the preparation of this deliverable. The three main elements are:
- **Activities:** Activities provide a coarse definition of the behaviour of the component. Activities are organised from left to right in the order selected to present the functionality of the component. Normally, they are organised following the user workflow.

- **Tasks:** Activities are divided into tasks, which are features needed to complete an activity. Tasks are organised from left to right following a logical sequence to complete the activity. Tasks have related UML Sequence diagrams to support its description.

- **Subtasks (User stories):** Describe features of an application from the point of view of the subject who expects the new feature. The subject is not restricted to a vf-OS user (eg an operator or developer) and can be any entity with a behaviour, eg the component being described, another component, etc. User stories follow a standard format: as a *who*, I want *what* so that *why*. This way, user stories capture in a simple sentence who wants what and how will the subject benefit from the new feature. To force this format, user stories are written in a schematic way, just specifying the *who, what* and *why* syntactical functions. User tasks should include acceptance criteria – a checklist that determines when the user story is considered as done. The acceptance criteria are also expressed from the point of view of the subject that formulates the user story and provides a detailed description of the criteria by which user stories should be evaluated and validated. User Stories have a unique ID per story (US001) in each story map. User stories are organised in releases in an incremental development plan. Thus, there are releases defined for the software deliverables of each component (eg M18, M24 and M30).

This way, the functional specification of each component contains its story map, together with tables describing each user story.

As mentioned above, in the description of a feature, UML sequence diagrams are used to depict the interaction between the main classes and external components to the component under definition. Additionally, when a given functionality is initiated by a user, a UI mock-up has been provided so that a clear understating of the functionality is achieved. Thus, the functional specification of each component includes a subsection with the corresponding UI mockups and UML sequence diagrams.

Finally, the interactions of the component are explained using an architecture diagram and detailing the messages exchange through a UML class diagram.
3 Environment Component

3.1 vf-OS Platform

The vf-OS Platform is the outermost ‘container’ component of vf-OS. It integrates the services for application development, middleware and deployment into a coherent holistic Platform.

3.1.1 Behaviour and Functionality

The main functionalities of the platform are the following:

- **Portal Function**: Realising the holistic encapsulation of the web-based user interfaces of all relevant vf-OS components and vApps into a single coherent user interface
  - **Registration Functions**: Registration of new users and companies, and the management of existing registrations
  - **Entry-point Functions**: Provision and rendering of the platform home page, user login and logout
  - **User Activity Logging Functions**: Logging of all high-level user actions, inspection and management of user activity logs

- **Execution Function**: Providing centralised management of a distributed environment in which all vf-OS components and vApps (and potentially other Assets) are running
  - **Host Management Functions**: Management of the hosts for the deployment and execution of components and vApps
  - **Component and vApp Management Functions**: Installation and management of components and vApps onto hosts as well as management of user access rights
  - **Execution Services Logging Functions**: Logging of all high-level execution services actions, inspection and management of execution services logs

An overview of activities, tasks and stories related to the Platform is shown in Figure 237.
The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPUS001 register new user</td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td><strong>Who:</strong> Manufacturing and Logistics User - company admin</td>
</tr>
<tr>
<td></td>
<td><strong>What:</strong> register a new user on the platform, including all metadata needed about the user and settings applicable to the user.</td>
</tr>
<tr>
<td></td>
<td><strong>Why:</strong> in order for new users to start using the platform</td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

A new user enters valid metadata, after which it immediately gets access to the top-level container of the platform portal, in which they can access the Marketplace for downloading the vf-OS core components (Pub/sub, System).
<table>
<thead>
<tr>
<th>VPUS002</th>
<th>register new developer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Dashboard, Container core) for local installation (if needed).</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>Software Developer - company admin</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>register a new developer on the platform, including at least all data required from a regular user plus some extra fields.</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>in order for new developers to start using the platform</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VPUS003</th>
<th>manage users/developers within a company</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>A new developer registers in the same way as a user does, but a developer needs to specify a few more details: a list of predefined vApp categories the developer is willing to develop, bank and payment details for the billing of development activities, and a yes/no option to automatically receive requests from users who would like new vApps to be built.</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>Manufacturing and Logistics User or Software Developer - company admin</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>use basic management functions for user accounts: view the list of developers, update the metadata and settings of a user or developer, or remove existing user accounts</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>in order to be able to maintain the list of users registered at the platform</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VPUS004</th>
<th>register new company</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>A user with a valid account is able to create a new company entry, of which he is the primary admin (this role can be switched later).</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>Manufacturing and Logistics User or Software Developer - company admin</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>register a new company on the platform, including all metadata needed from the company and settings applicable to a company</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>in order to provide a group structure for coherently maintaining the users, activities, access control, accounting and billing for a company</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VPUS005</th>
<th>edit company settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>After alteration of the settings of the company, the settings immediately become active. Deletion of a company can only be done if there are no users associated with the company except the user who is currently deleting the company.</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>Manufacturing and Logistics User or Software Developer - company admin</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>edit the metadata and settings of a company, including the deletion of the company</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>in order to provide the possibility to alter the information about and settings for a company</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VPUS006</th>
<th>provide a top-level container</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The top-level container shows an overview of components and vApps installed and accessible for this user. At least the following components are always presented to the user: System Dashboard and Marketplace. Pub/sub and Container core are primarily operating &quot;under the hood&quot;, but their configuration dashboards are accessible via the System Dashboard. Components and vApps are only shown if they have been installed on registered hosts (cloud + on-premise machines) and if the user has the appropriate access rights for them and the hosts they are deployed at.</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>All Users</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>view and use the portal entry point for the platform</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>in order to access all functionality provided by the platform, its installed core and additional components and vApps, by means of a transparent single sign-on procedure</td>
</tr>
</tbody>
</table>

| VPUS007 | Description |
| **log in** | **Who:** All Users  
**What:** sign in to the platform  
**Why:** in order to use the functionality provided by the platform |
| **Acceptance Criteria** | After login, the user has access to the top-level container (See VPUS006). |
| **VPUS008 log out** | **Description** |
| **Acceptance Criteria** | After logout, the user has no longer access to the top-level container. |
| **log all user activity** | **Who:** Platform component  
**What:** log all actions of all users in the portal: login/logout, user management actions, and single sign-on to components and vApps  
**Why:** in order to track and trace user activities and to provide a means for due diligence |
| **Acceptance Criteria** | All actions of all users can be inspected by checking the user activity logs. |
| **view user activity logs** | **Who:** Manufacturing and Logistics User or Software Developer - company admin  
**What:** inspect the logs collected about the activities of users  
**Why:** in order for authorised users to be informed about the (potentially dishonest) actions of other users |
| **Acceptance Criteria** | The user activity logs contain all actions of all users: login/logout, user management actions, and single sign-on to components and vApps |
| **purge user activity logs** | **Who:** Manufacturing and Logistics User or Software Developer - company admin  
**What:** remove log entries collected about the activities of users  
**Why:** in order to clean up the logs for technical or legal reasons |
| **Acceptance Criteria** | After confirmation, the deleted logs are physically removed from the log store and can no longer be inspected in the user activity logs. |
| **Add host** | **Description** |
| **Acceptance Criteria** | After the host has been added, it is possible to host components and vApps at the host and to view the host status in the System Dashboard. |
| **Edit hosts** | **Who:** Manufacturing and Logistics User - IT Manager  
**What:** use basic management functions for hosts: view the list of hosts, update the settings of a host, or remove existing hosts  
**Why:** in order to be able to maintain the list of hosts registered at the platform |
| **Acceptance Criteria** | This functionality provides read, update and delete functionality for hosts. Next to edit the settings for a host, is it also possible to inspect the usage and utilisation statistics for a host. |
| **install a new component or vApp** | **Description** |
| **Acceptance Criteria** | in order to add new functionality for users to the platform |
Acceptance Criteria
After installation of the component or vApp, its configuration can be managed via the System Dashboard. Installation includes the option for the user to accept or refuse the installation of dependencies. If dependencies are refused, the installation is aborted. Furthermore, at each installation of a component or vApp, the user can choose at which host to deploy the component or vApp (the user can choose any available host registered in the platform and accessible to that user, either in-cloud or on-premise). In addition, the user can choose to install the component or vApp for a particular user or for company-wide use. Finally, newly installed components and vApps must be explicitly activated (see VPUS016) in order to be used.

<table>
<thead>
<tr>
<th>VPUS015</th>
<th>manage installed components and vApps</th>
</tr>
</thead>
</table>
| **Description** | Who: Manufacturing and Logistics User - IT Manager  
What: use basic management functions for installed components and vApps: view the list of installed components and vApps, update the settings of a component or vApp, or remove installed components or vApps  
Why: in order to be able to maintain the functionalities provided by the platform to users |
| **Acceptance Criteria** | After changes have been confirmed, they are immediately effectuated. Dependencies of components / vApps need to be removed explicitly and separately. |

<table>
<thead>
<tr>
<th>VPUS016</th>
<th>manage component or vApp settings</th>
</tr>
</thead>
</table>
| **Description** | Who: Manufacturing and Logistics User - IT Manager  
What: view and edit the settings for a component or vApp, including its access to other components and vApps, as well as its runtime state (activated/deactivated, start/stop)  
Why: in order to alter the configuration of a certain component or vApp and to enable or disable its usage |
| **Acceptance Criteria** | After changes in the settings have been confirmed, they are immediately effectuated. |

<table>
<thead>
<tr>
<th>VPUS017</th>
<th>manage user access to hosts, installed components and vApps</th>
</tr>
</thead>
</table>
| **Description** | Who: Manufacturing and Logistics User - IT Manager  
What: use basic management functions for access control to hosts, components and vApps: allow or deny users to use hosts and components / vApps, allow users to alter settings of hosts and components / vApps, allow users to add/delete hosts and components / vApps, etc.  
Why: in order to provide fine-grained access control with respect to the usage of the platform, its hosts and components / vApps by users |
| **Acceptance Criteria** | After changes in the settings have been confirmed, they are immediately effectuated. |

<table>
<thead>
<tr>
<th>VPUS018</th>
<th>log all execution services activity</th>
</tr>
</thead>
</table>
| **Description** | Who: Manufacturing and Logistics User - company admin  
What: log all administrative actions regarding hosts and installed components / vApps  
Why: in order to track and trace user activities and to provide a means for due diligence |
| **Acceptance Criteria** | All execution services activity can be inspected by checking these logs. |

<table>
<thead>
<tr>
<th>VPUS019</th>
<th>view execution services activity logs</th>
</tr>
</thead>
</table>
| **Description** | Who: Manufacturing and Logistics User - company admin  
What: inspect the logs collected about the administration activities regarding hosts and installed components / vApps  
Why: in order for authorised admins to be informed about the (potentially dishonest) actions of other admins |
3.1.2 UI mockups and Sequence Diagrams

The following sub-sections show sequence diagrams and UI mock-ups to clarify the stories sketched above and the vf-OS internal interactions related to them.

3.1.2.1 User Registration

Figure 5 shows the sequence diagram related to the registration of users, which primarily takes place via the platform Portal UI.
Figure 5: User Registration Sequence Diagram
The main steps/functionalities are:

- Register New User
- Register New Developer
- Manage Users / Developers within Company

The user interface for the registration of new users is shown as part of the Frontend component (a user registration form for FEUS005). It is a basic registration form in which first name, last name and email address are requested and the new user must agree with the platform terms and agreements. For the registration of developers, the form is followed by another form with more details, which is shown in Figure 245. These details can easily be changed later by means of an interface as shown in Figure 7. Note also the option here to add a company profile (VPUS003 / VPUS004). The UI Mock-up for the management of users within a company is integrated into the Mock-up for the registration of a company, as will be shown later in Figure 9.
Figure 6: Register New Developer UI Mock-up
3.1.2.2 Company Registration

Users can add a profile of their company (or, if they want to be added to an existing company profile, request the administrator of that company to arrange this). The sequence diagram for this task is shown in Figure 239.

The main steps/functionalities are:

- Register New Company
- Edit Company Settings
Figure 8: Company Registration Sequence Diagram
A user interface mock-up for the registration of a company (the editing of company details) and the management of users within this company (VPUS003) is shown in Figure 247.

![Company Registration UI Mock-up](image)

**Figure 9: Company Registration UI Mock-up**

### 3.1.2.3 Portal Entry point Functions

One of the key functions of the platform is to act as the first entry point and outward façade of the platform, as well as to provide login/logout and single sign-on functionality. Figure 10 shows the sequence diagram for these functions.

The main steps:functionalities are:

- Provide Top-level Container
- Log in
- Log out
The UI mock-up for the platform portal home page (after login) is shown in Figure 11. The home page simply provides an overview of all installed components and vApps that can be accessed by the user. The login UI mock-up is not shown here because it will be shown as part of the Frontend component (FEUS006-008).
3.1.2.4 User Activity Log Management

The sequence diagram for creating, monitoring and managing user activity logs is shown in Figure 12.

The main steps/functionalities are:

- Log all User Activity
- View User Activity Logs
- Purge User Activity Logs
Note that logging (VPUS009) takes place before and after all actions taken by the portal server and happens without any user intervention and hence without any user interface. The logs can be accessed via the company settings and users page in the platform portal. The UI mock-up for this functionality is shown in Figure 13.
3.1.2.5 Manage Platform Hosts

Execution services of the platform are all managed via the System Dashboard. The sequence diagram for adding, updating and deleting hosts attached to the platform is shown in Figure 14.

The main steps/functionalities are:

- Add Hosts
- Edit Hosts
Figure 14: Manage Platform Hosts Sequence Diagram

A user interface mock-up for the management of hosts is shown in Figure 15. All configured hosts are depicted both as a connected graph and as an editable list.
3.1.2.6 Manage Installed Components and vApps

The sequence diagram for managing installed assets (components and vApps) is shown in Figure 16.

The main steps/functionalities are:

- Install new Component or vApp
- Manage Installed Components and vApps
- Manage Component or vApp Settings
- Manage User Access to Hosts, Installed Components and vApps
A user interface mock-up for the installation of new components or vApps is shown in Figure 17. This interface is assumed to be triggered by the downloading / installation of a new component or vApp from the Marketplace and approval of the component or vApp through Security (SCUS010).

A mock-up for the management of components and vApps is shown in Figure 18. Usage and resource utilisation statistics, as well as an editable list, is shown as part of the platform dashboard inside the System Dashboard.

Finally, Figure 19 shows a UI mock-up for the management of user access to hosts, components and vApps, and general user modification rights.
Figure 17: Install new Components / vApp UI Mock-up
Figure 18: Manage Installed Components and vApps UI Mock-up
3.1.2.7 Execution Services Log Management

Log management for the execution services of the platform is managed via the System Dashboard as well. The sequence diagram for these log services is shown in Figure 20. The main steps/functionalities are:

- Log all Execution Services Activity
- View Execution Services Activity Logs
- Purge Execution Services Activity Logs

The execution services activity logs are updated before and after each action taken by the execution manager. These updates do not require any user intervention.
Figure 20: Execution Services Log Management Sequence Diagram
A user interface mock-up for the management of the execution services logs is shown in Figure 21.

![System Dashboard UI](http://dashboard.vf-os.eu)

**Figure 21: Execution Services Log Management UI Mock-up**

Interaction description

An Interaction Diagram for the Platform is shown in Figure 22. The interactions with other components can be categorised as follows:

- Interaction with the System Dashboard: Exchange of metadata and status information about hosts and deployed containerised assets, in order for the System Dashboard to provide the full platform dashboard functionality on behalf of the platform.
- Interaction with the user interfaces of other vf-OS components: This mainly takes place via integration-ware provided by the platform and is about the rendering of these user interfaces as part of the encapsulating platform portal.
• Interaction with virtualised/containerised assets (components or vApps): The retrieval of status information and the sending of actions to be performed by the asset (such as starting, stopping, deactivation or sending specific status information upon request)

• Interaction with vf-OS components for authentication/authorisation purposes: vf-OS components were supposed to make use of the authentication/authorisation service of the platform. This functionality is now considered deprecated since the Frontend environment handles this functionality together with the Security component (see Frontend)

• Interaction with Security: Mainly for the purpose of the Security component. This functionality is now considered deprecated since Security does no longer need this information (see Security)
Figure 22: Platform Interaction Diagram
4 Application Development (Design) Component

4.1 OAK Toolkit

4.1.1 OAK SDK

4.1.1.1 Behaviour and Functionality

The vf-OAK Software Development Kit (SDK) is a central environment for the development of applications and, generically, for the centralised access of the vf-OS assets and functionalities. The SDK itself will not have a user interface per se, instead, it will be accessed as a set of APIs to access the main development resources. That way, the SDK will be able to provide to the Studio and to other Application-Development components, the resources, and services that they require, as well as to access design orientated data stored in vf-OS (models, patterns, and behaviours). In general, data and models will be able to be retrieved from multiple sources: internal vf-OS Storage, vf-Store, and vf-OS Assets including Generic Enablers, external services and tools through the External Service Provision, and access to the real manufacturing devices, sensors and other mechanisms via vf-IO, within the I/O Toolkit. Pre-defined code snippets and patterns will additionally be able to be reused coming from the Development Engagement Hub.
The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDUS001 Get vApps</td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td><strong>What:</strong> will list existing vApps (stored on the vf-Store) getting their Names, detail on functionality, version, etc</td>
</tr>
<tr>
<td></td>
<td><strong>Why:</strong> browse the existing vApps to reuse them or to understand how to interact with them</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td>Invoker got a structured list of the vApps stored on vf-Store</td>
</tr>
<tr>
<td>SDUS002 Get vf-OS Assets</td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td><strong>What:</strong> will list existing vf-OS assets (components, services) getting their Names, detail on functionality, version, etc</td>
</tr>
<tr>
<td></td>
<td><strong>Why:</strong> browse the existing services to reuse them or to understand how to interact with them</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td>Invoker got a structured list of the vf-OS components and services</td>
</tr>
<tr>
<td>SDUS003 Get Data Analytics Services</td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td><strong>What:</strong> will list existing Data Analytics services, getting their names, detail on functionality, version, etc</td>
</tr>
<tr>
<td></td>
<td><strong>Why:</strong> browse the existing services to understand how to use them</td>
</tr>
</tbody>
</table>
### Acceptance Criteria
Invoker got a structured list of the Data Analytics services

<table>
<thead>
<tr>
<th>SDUS004</th>
<th>Get Enablers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will list existing vf-OS Enablers, getting their names, detail on functionality, version, etc</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> browse the stored Enablers to understand how to use them</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got a structured list of the stored Enablers</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS005</th>
<th>Get Data Models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will list existing Data Models and stored patterns, getting their names, detail on functionality, version, etc</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> browse the existing models and patterns to understand how to use them</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got a structured list of the stored Data Models and patterns</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS006</th>
<th>Get Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will list existing Drivers, getting their names, detail on functionality, version, etc</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> browse the existing Drivers to understand how to interact with them</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got a structured list of the stored Drivers</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS007</th>
<th>Get List of Stored Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will list existing SDK Configuration files, getting their names, detail on functionality, version, etc</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> browse the existing configurations to model the SDK</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got a structured list of configurations</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS008</th>
<th>Get Studio Manifests</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will list existing Studio Configuration Manifests</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to build the Composer manifest that will be used in the compilation and building of the final solution</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got a structured list of manifests</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS011</th>
<th>Get vApp APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve the APIs for the selected vApp</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> make use of the APIs</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got APIs definition and description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS012</th>
<th>Get vf-OS Asset APIs and Manifests</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve the API and manifest files for the selected vf-OS asset</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> make use of the APIs and manifests</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got API and manifest definition and description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS013</th>
<th>Get Data Analytics service API</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve the API and manifest files for the selected Data Analytics service</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> make use of the APIs and manifests</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>SDUS014</td>
<td>Get Enabler Manifests</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve the manifest files for the selected vf-OS Enabler</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> make use of the manifests</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got manifest definition and description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS015</th>
<th>Get Data Model definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve the APIs and manifest files for the selected data model or design pattern</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> make use of the APIs and manifests</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got API and manifest definition and description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS016</th>
<th>Get Driver APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve the API and manifest files for the selected Driver</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> make use of the APIs and manifests</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got API and manifest definition and description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS017</th>
<th>Get Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve the data for the selected configuration file stored in the Data Storage</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> use the data for configuration of the SDK, Process Designer or Studio</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got the selected Configuration information and its description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS018</th>
<th>Get Studio Manifest</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve the data for the selected Studio Manifest</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> use the manifest for building the final solution to be deployed</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker got the selected Manifest</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS101</th>
<th>Retrieve Composition data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will submit the set of APIs and Manifests describing the various services, and particularly a manifest that describes how the services should be composed</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to provide all information needed for the vApp build</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>vApp Composer received the set of APIs and Manifests corresponding to the vApp being developed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS102</th>
<th>Validate Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApp Composer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will analyse the received data corresponding to the development of a vApp and check the dependencies of the various involved modules</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to ensure all information needed for performing the vApp build is available</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>All dependencies of the received modules are included and accessible, whether using local vf-OS repositories (Data Storage, vf-Store, Developer Engagement Hub) or remote (internet, local machine uploading)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDUS103</th>
<th>Validate APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApp Composer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will analyse the interfaces and API specification and ensure that the invocation data complies to that specification</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to ensure that the defined APIs are being respected</td>
<td></td>
</tr>
</tbody>
</table>
## Acceptance Criteria

All invocations of internal and external interfaces comply with the APIs specifications.

### SDUS104
#### Structure the Build Manifest

**Description**

**Who:** vApp Composer  
**What:** ensure that the Build manifest complies to a set of process steps that are meaningful  
**Why:** to ensure the process steps needed for performing the build make sense and are listed in an appropriate way

**Acceptance Criteria**

vApp Composer returns success on the analysis of the vApp Build Manifest.

### SDUS201
#### Build

**Description**

**Who:** vApps Developer  
**What:** will submit a set of APIs and Manifests describing the new vApp, together with the modules of code for building, and the response should be whether the build was successful or not.  
**Why:** validate the Build of the new vApp

**Acceptance Criteria**

Invoker got response regarding the Build invoking after submitting a set of information.

### SDUS301
#### Deploy

**Description**

**Who:** vApps Developer  
**What:** will submit a set of APIs and Manifests describing the new vApp, and its code/built executable  
**Why:** store the new vApp in the vf-Store

**Acceptance Criteria**

Invoker got response about deploying the new vApp in the vf-Store.

### SDUS501
#### Store Configuration

**Description**

**Who:** vApps Developer  
**What:** will store the vApp project configuration file to the Data Storage  
**Why:** store the new vApp configuration to the Data Storage, as well as the configuration of the SDK and Studio that were used for building it

**Acceptance Criteria**

Invoker got response about storing the configurations in the Data Storage.

### SDUS401
#### List Hub service APIs

**Description**

**Who:** vApps Developer  
**What:** will list the various APIs available for the Developer Engagement Hub  
**Why:** to be able to invoke one or more of these services

**Acceptance Criteria**

Invoker got a list of APIs available (and their description) to the Developer Engagement Hub services.

### SDUS402
#### Configure Hub service

**Description**

**Who:** vApps Developer  
**What:** will send to the Developer Engagement Hub an updated configuration of the DE project associated with the vApp being developed  
**Why:** to change permissions, update or submit comments

**Acceptance Criteria**

Invoker got confirmation from the Developer Engagement Hub services.

### SDUS403
#### Add Code to Version Control

**Description**

**Who:** vApps Developer  
**What:** will send to the Developer Engagement Hub the definition of one or more modules of code of the vApp being developed  
**Why:** to prepare the commit of code in the version controlled repository

**Acceptance Criteria**

Invoker got confirmation from the Developer Engagement Hub services.
4.1.1.2 UI mockups and Sequence Diagrams

The following sub-sections describe the sequence diagrams describing the interactions between the vf-OAK SDK and its surrounding modules. As the SDK is an API it has no user interface (better saying, the Studio is the User Interface of the SDK).

4.1.1.2.1 Browse vApps and select a vApp

The SDK shall allow a developer (via API), or any other client service (eg Studio) to retrieve the set of vApps from the vf-OS Marketplace, as can be seen in Figure 24.
4.1.1.2.2 Retrieve Assets from Data Storage

Similarly to the vApps retrieval, also the multiple assets that are available on vf-OS can be retrieved from the Data Storage, using the flow depicted on Figure 25.

The main steps / functionalities are:

- Query the existing vAssets and select a set of vAssets
- Apply a desired filter and Select the desired vAsset

This sequence diagram, therefore, is similar for all the pair scenarios SDUS002 and SDUS012, SDUS003 and SDUS013, SDUS004 and SDUS014, SDUS005 and SDUS015, SDUS006 and SDUS016, and SDUS007 and SDUS017.
4.1.1.2.3 Retrieve the Project Manifest from the Studio
Similarly to the vApps retrieval, also the manifest that is being developed in the vf-OAK Studio can be retrieved, using the flow depicted on Figure 26.

The main steps / functionalities are:
- Query the Studio for the corresponding Manifest
- Retrieve the Studio Manifest

4.1.1.2.4 Invoke Plugin API
The SDK will include the possibility of working with different modules such as application builders, composers, and will be able to be extended to reuse any generic type of component, simply exposing its API to the SDK clients. These are plug-in modules that can be invoked by the SDK. Hence, these plugins also need to be connected to the SDK and requested to be accessed by their APIs, as shown in Figure 27. This scenario is one that can be seen in stories SDUS011, SDUS012, SDUS013, and SDUS016.
The main steps / functionalities are:

- Invoking the SDK for calling the desired functionality
- The SDK determines if there is any configuration needed for the execution
- The SDK completes the component configuration and invokes it to retrieve its API list
- The SDK returns to the caller the API of the plugged component

![Diagram](image)

**Figure 27: Invoking Plugin API Functionalities**

### 4.1.1.2.5 Invoke the Service Composition Services

The SDK will expose the API for the developer (or Studio) to compose a vf-OS Application. The supporting applications shall themselves be available at the vf-OS Platform (vf-P), and the service shall be invoked through the SDK, as can be seen in Figure 29.

The main steps / functionalities are, when invoking the SDK for Composing the Application:

- The SDK retrieves the vApp Configuration
- The SDK invokes the Dependency checker, to see if all needed sources/libraries are available in the vf-OS Repository
- The SDK invokes the API checker, to see if all Interfaces are being compliant
- The SDK invokes an application to define the build manifest with the outcomes of the previous calls
- The SDK returns the Build Manifest to the caller. This is the needed input for making the vApp build in the future
4.1.1.2.6 Build vApp

The SDK will expose the API for the developer (or Studio) to build a vf-OS Application. The builders shall themselves be available at the vf-OS Platform (vf-P), and the service shall be invoked through the SDK, as can be seen in Figure 29.

The main steps / functionalities are:

- Invoking the SDK for Building the Application
- The SDK retrieves and configures the appropriate Builder on the vf-P
- The SDK invokes the Build process
- The SDK returns to the caller the building report

4.1.1.2.7 Deploy a vApp

The SDK will expose the API for the developer (or Studio) to deploy a vf-OS Application. The creation of the deployment instance container shall itself be available at the vf-OS
Platform (vf-P), and the service shall be invoked through the SDK, as can be seen in Figure 30.

The main steps / functionalities are:

- Invoking the SDK for Deploying the Application
- The SDK configures the deployment environment on the vf-P
- The SDK invokes the Deploy process
- The SDK returns to the caller an instance of the deployed vApp

![Figure 30: Deploying a vApp](image)

4.1.1.2.8 Invoke the Developer Engagement Hub APIs

The SDK will include the possibility of working with the APIs of the Developer Engagement Hub. No interactions are foreseen here as the SDK will simply expose the Hub's APIs, as seen in Figure 31. This will cover the scenarios from SDUS401 to SDUS407.

The main steps / functionalities are:

- Invoking the SDK for getting the appropriate Engagement Hub API for the current project
4.1.1.3 Interaction description

The following diagram (Figure 32) was taken from the global architecture definition presented in D2.1, and the subsequent text focuses on the interactions and data exchange between the SDK and other vf-OS components.

In order to clarify the interactions between components, the main interactions of the SDK component with other components are:

- Usage API: Provides access to most of the vf-OS assets such as:
  - It retrieves configuration files and other stored items from the vf-OS Data Storage
- It retrieves services for data transformation, security and other services provided by the vf-OS framework
- It provides access to the vf-OS available development Enablers
- It provides access to the OAK Developers Engagement Hub functionalities, allowing the callers to be able to retrieve documentation, tutorials, issues and different versions of stored code for a developing unit

- vApp Composer: This component deals with the development of a new vApp, whether invoked from the Studio or from a Developer (using the SDK API). Its main information flows are:
  - It retrieves configuration files and others, needed for the configuration and build of the vApp
  - It is able to access all vf-OS Assets for supporting the build of the application
  - It is able to query the Marketplace for vApps that are needed for reuse or composition of the new vApp
  - It can query the Studio for the associated Studio Manifest or Build Manifest
  - It can invoke the vf-P for executing the needed services for Dependency checks, Build and Deployment of the vApp
  - It can hand to the vf-Store the new vApp to be registered in the

4.1.2 OAK Studio

4.1.2.1 Behaviour and Functionality

The vf-OAK Studio is a central piece in the vf-OS strategy as it is the Integrated Development Environment that will be used for developing the vApps. It is mainly a GUI that invokes many of the functionalities available by the vf-OAK SDK, but it also includes a rich set of features, such as a code editor with drag and drop of elements, builds automation, code completion, compilers, testing debugger and integration with the Developer Engagement Hub.
### Subtask Details

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STUS001</strong></td>
<td><strong>Log in to Studio Frontend UI</strong></td>
</tr>
</tbody>
</table>
| **Description** | Who: vApps Developer  
What: log in the Studio Frontend UI  
Why: to access the Studio services |
| **Acceptance Criteria** | Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment "Login User") |
| **STUS002**     | **Get Configuration**                                                                 |
| **Description** | Who: vApps Developer  
What: will list existing configurations for the Studio  
Why: select the most suitable Studio configuration for building the vApp. This may include access to different suites of tools, look & feel customisation, or other types of configuration |
| **Acceptance Criteria** | Invoker got a structured list of the Studio configurations |
| **STUS003**     | **Set Configuration**                                                                 |
| **Description** | Who: vApps Developer  
What: will select an existing configuration for the Studio  
Why: select one Studio configuration for building the vApp. This may include access to different suites of tools, look & feel customisation, or other types of configuration |
| **Acceptance Criteria** | The Studio will be customised in accordance with the selected configuration. A file with the Studio configuration will be stored in the platform connected with the user session. |
| **STUS005**     | **Log in to Studio Frontend UI**                                                       |
| **Description** | Who: vApps Developer  
What: log in the Studio Frontend UI  
Why: to access the Studio services |
| **Acceptance Criteria** | Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment "Login User") |
| **STUS006**     | **Edit Configuration**                                                                 |
| **Description** | Who: vApps Developer  
What: edit and update the configuration of the Studio  
Why: customise the most suitable Studio configuration for building the vApp. This may include access to different suites of tools, look & feel customisation, or other types of configuration |
| **Acceptance Criteria** | Invoker accesses a UI frontend with the current Studio configuration and can change it. The Studio will be customised to reflect the updated configuration. A file with the Studio configuration will be stored in the platform connected with the user session. |
| **STUS007**     | **Store Configuration**                                                                 |
| **Description** | Who: vApps Developer  
What: will store the existing configurations for the Studio  
Why: for reuse purposes on later executions |
| **Acceptance Criteria** | Invoker got a confirmation that the configuration was stored in the vf-OS Data Storage |
| **STUS051**     | **Description**                                                                        |

---

**Figure 33: vf-OAK Studio Story Map**

The textual description of each user story is as follows:
<table>
<thead>
<tr>
<th><strong>STUS052</strong></th>
<th><strong>Browse vf-OS Assets</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Invoker received a list of vf-OS Assets (with documentation) coming from the SDK and from the Marketplace. A file with the selected list of vf-OS Assets will be stored in the platform connected with the user session.</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>STUS101</strong></th>
<th><strong>Log in to Studio Frontend UI</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>STUS102</strong></th>
<th><strong>Define Project Name</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Studio creates a blank workspace for the new project in the vf-OS Platform (vf-P). A file with the Project configuration will be stored in the platform connected with the user session.</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>STUS103</strong></th>
<th><strong>Configure Project</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Project behaviour and Project definition files are updated to use the selected parameters.</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Project behaviour and Project definition files are updated to use the selected parameters.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>STUS104</strong></th>
<th><strong>Add Assets</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Project configuration file on the vf-OS Platform stores the set of assets. If the assets are coming from the Marketplace, invoke the Marketplace &quot;Place new order&quot;</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Project configuration file on the vf-OS Platform stores the set of assets. If the assets are coming from the Marketplace, invoke the Marketplace &quot;Place new order&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>STUS105</strong></th>
<th><strong>Add Dependencies</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Invoker accesses the Studio Frontend UI and is able to define dependencies for the current vApp</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI and is able to define dependencies for the current vApp</td>
</tr>
<tr>
<td>STUS151</td>
<td>Log in to Studio Frontend UI</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A file with the project dependencies configuration will be stored in the platform connected with the current project</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>log in the Studio Frontend UI</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to access the Studio services</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS152</th>
<th>Open Code Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>invoke the selected Code Editor’s Frontend UI</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to edit the code being developed for the vApp</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI for the determined purpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS153</th>
<th>Manage Code Files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>organises the file structure corresponding to the project structure. This may include actions like creating new files, removing files, renaming files, creating a hierarchical list of files, with drag and drop ability</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>define the project structure</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Studio stores the project structure as a set of organised files in the vf-OS Platform (vf-P) connected with the current project A file with the Project configuration will be updated and stored in the platform connected with the user session Saved files are kept in the vf-OS Platform connected with the current project</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS201</th>
<th>Log in to Studio Frontend UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>log in the Studio Frontend UI</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to access the Studio services</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS202</th>
<th>Invoke Code Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>invoke the selected Code Compiler</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to compile the code and resources developed for the vApp</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI for the determined purpose If configured to do so, this may be triggered upon saving code files The resulting compiled results will be stored in the project structure, kept in the vf-OS Platform (vf-P)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS203</th>
<th>Browse List of Compiler Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>browse the results of the Code Compiler</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to analyse eventual compilation errors and warnings</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Studio Frontend UI for the determined purpose The resulting compilation report will be stored in the project structure, kept in the vf-OS Platform (vf-P)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS251</th>
<th>Log in to Studio Frontend UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>log in the Studio Frontend UI</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to access the Studio services</td>
</tr>
<tr>
<td><strong>STUS252</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| **Invoke Dependency Checker** | **Who:** vApps Developer  
  **What:** invoke the selected Dependency Checker  
  **Why:** to compile the code and resources developed for the vApp |

| **Acceptance Criteria** | Invoker accesses the Studio Frontend UI for the determined purpose  
  The resulting list of dependencies report will be stored in the project structure, kept in the vf-OS Platform (vf-P) |

<table>
<thead>
<tr>
<th><strong>STUS253</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Browse Dependency Checker Results** | **Who:** vApps Developer  
  **What:** browse the results of the Dependency Checker  
  **Why:** to analyse eventual dependency errors and warnings before building the vApp |

| **Acceptance Criteria** | Invoker accesses the Studio Frontend UI for the determined purpose  
  The resulting dependencies report will be stored in the project structure, kept in the vf-OS Platform (vf-P) |

<table>
<thead>
<tr>
<th><strong>STUS301</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Log in to Studio Frontend UI** | **Who:** vApps Developer  
  **What:** log in the Studio Frontend UI  
  **Why:** to access the Studio services |

| **Acceptance Criteria** | Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment "Login User") |

<table>
<thead>
<tr>
<th><strong>STUS302</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Invoke Code Builder** | **Who:** vApps Developer  
  **What:** invoke the selected Code Builder’s Frontend UI  
  **Why:** to build the code being developed for the vApp |

| **Acceptance Criteria** | Invoker accesses the Studio Frontend UI for the determined purpose  
  The resulting built results will be stored in the project structure, kept in the vf-OS Platform (vf-P) |

<table>
<thead>
<tr>
<th><strong>STUS303</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Browse List of Build Errors** | **Who:** vApps Developer  
  **What:** browse the results of the Code Builder  
  **Why:** to analyse eventual build errors and warnings |

| **Acceptance Criteria** | Invoker accesses the Studio Frontend UI for the determined purpose  
  The resulting build report will be stored in the project structure, kept in the vf-OS Platform (vf-P) |

<table>
<thead>
<tr>
<th><strong>STUS351</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Log in to Studio Frontend UI** | **Who:** vApps Developer  
  **What:** log in the Studio Frontend UI  
  **Why:** to access the Studio services |

| **Acceptance Criteria** | Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment "Login User") |

<table>
<thead>
<tr>
<th><strong>STUS352</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Invoke Code Test Debugger** | **Who:** vApps Developer  
  **What:** invoke the selected Code Test Debugger’s Frontend UI  
  **Why:** to make a step-by-step debugging test over the vApp |

<p>| <strong>Acceptance Criteria</strong> | Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;) |</p>
<table>
<thead>
<tr>
<th>STUS401</th>
<th>Log in to Studio Frontend UI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invoker accesses the Studio Frontend UI for the determined purpose</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>log in the Studio Frontend UI</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>to access the Studio services</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS402</th>
<th>Invoke the System Dashboard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>invoke the System Dashboard's Frontend UI</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>to analyse the System Dashboard</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS451</th>
<th>Log in to Studio Frontend UI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>log in the Studio Frontend UI</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>to access the Studio services</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS452</th>
<th>Log in to Developers Engagement Hub</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>log in the Developer's Engagement Hub's Frontend UI</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>to access the Hub services. Note that the DE credentials may be different than the Studio credentials</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS453</th>
<th>Browse Developer Engagement Hub's Tickets</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invoker receives a list of tickets related with the current vApp.</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>invoke the Developers' Engagement Hub Frontend UI</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>to browse all Developers' Engagement Hub for tickets associated with the current vApp</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS501</th>
<th>Log in to Studio Frontend UI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>log in the Studio Frontend UI</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>to access the Studio services</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS502</th>
<th>Log in to Developers Engagement Hub</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>log in the Developer's Engagement Hub's Frontend UI</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>to access the Hub services. Note that the DE credentials may be different than the Studio credentials</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS503</th>
<th>Select Hub Project</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
<tr>
<td>Who</td>
<td>vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What</td>
<td>browse the Developer Engagement Hub's list of projects and select one</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>to make the following actions in the scope of a specific DE project</td>
<td></td>
</tr>
<tr>
<td>STUS504</td>
<td>Create Ticket</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>What:</strong> Use the Developer Engagement Hub's UI to create a ticket</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Why:</strong> create a new ticket in a specific DE project</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Invoker accesses the Developer Engagement Hub Frontend UI for the selected purpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS551</th>
<th>Log in to Studio Frontend UI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>What:</strong> log in the Studio Frontend UI</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Why:</strong> to access the Studio services</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS552</th>
<th>Log in to Developers Engagement Hub</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>What:</strong> log in the Developer's Engagement Hub's Frontend UI</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Why:</strong> to access the Hub services. Note that the DE credentials may be different than the Studio credentials</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Invoker accesses the Developer's Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS553</th>
<th>Select Hub Project</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>What:</strong> browse the Developer Engagement Hub's list of projects and select one</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Why:</strong> to make the following actions in the scope of a specific DE project</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Invoker accesses the Developer Engagement Hub Frontend UI for the selected purpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS554</th>
<th>Prepare to Commit Code files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>What:</strong> Use the Studio's UI to select a set of code files to be committed in the selected DE project repository.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Why:</strong> store the selected files under version control</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Invoker accesses the Studio's Frontend UI for the selected purpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS555</th>
<th>Commit Code files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>What:</strong> Use the Developer's Engagement Hub's UI to add the files to staged commit</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Why:</strong> store files under version control in a specific DE project</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Invoker accesses the Developer's Engagement Hub Frontend UI for the selected purpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS556</th>
<th>Solve Ticket</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Who:</strong> vApps Developer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>What:</strong> Use the Developer's Engagement Hub's UI to solve a ticket when committing the file, placing comments if needed</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Why:</strong> Resolve a ticket with a clearly identified version commit</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Invoker accesses the Developer's Engagement Hub Frontend UI for the selected purpose</td>
</tr>
<tr>
<td>STUS901</td>
<td>Log in to Studio Frontend UI</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: vApps Developer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What: log in the Studio Frontend UI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: to access the Studio services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invoker accesses the Studio Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS902</th>
<th>Prepare vApp for Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Who: vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What: will select a set of files that comprise the productised vApp (eg binaries, documentation, description, configuration, supporting files)</td>
<td></td>
</tr>
<tr>
<td>Why: for building the vApp package that will be published in the Marketplace</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td>Developer was able to select all files and include them in a set</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS903</th>
<th>Create/Update vApp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Who: vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What: will submit the vApp for creation (invoke Marketplace &quot;VMUS034: Create new Asset&quot;) or update (invoke Marketplace VMUS031 and &quot;VMUS032: Upload new version of vf-OS asset&quot;)</td>
<td></td>
</tr>
<tr>
<td>Why: storage of the developed vApp in the marketplace</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td>Invoker uploaded the developed vApp, and updated the information about the vApp, and the Studio displays the result from the Marketplace</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS904</th>
<th>Pricing Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Who: vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What: will enter/update the pricing information (invoke Marketplace &quot;VMUS038 Enter pricing information&quot;)</td>
<td></td>
</tr>
<tr>
<td>Why: determine the pricing information for the developed vApp</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td>Invoker used the UI and got a response from the Marketplace</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUS905</th>
<th>Check Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Who: vApps Developer</td>
<td></td>
</tr>
<tr>
<td>What: will check the Marketplace's error information about the uploaded vApp (invoke Marketplace VMUS040 and &quot;VMUS041: Enter pricing information&quot;)</td>
<td></td>
</tr>
<tr>
<td>Why: determine any errors on the publication of the developed vApp</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td>Invoker used the UI and got a response from the Marketplace</td>
<td></td>
</tr>
</tbody>
</table>

### 4.1.2.2 UI mockups and Sequence Diagrams

The following sub-sections describe the UI mock-ups and sequence diagrams describing the interactions between the vf-OAK Studio, the Developer, and the SDK.

#### 4.1.2.2.1 Authorisation Scenarios

The Studio will always require authentication and authorisation for all its actions, hence one initial step that is performed at every action is the check if the developer has logged in or has valid credentials, and if not, pop-up the login page. The interaction that is depicted in Figure 34 is valid for all scenarios STUS001, STUS005, STUS051, STUS101, STUS151, STUS201, STUS251, STUS301, STUS351, STUS401, STUS451, STUS501, STUS551, and STU901. Moreover, in the cases where the Developer Engagement Hub is also used, and as the user in the Studio can be different than in the DE, this interaction is also needed for the scenario steps STUS452, STUS502, and STUS552.
4.1.2.2 Configure Studio

In order to be configured, the Studio will request the SDK for the set of configurations available on the Data Storage. Upon receiving this list, the Developer will select one configuration and retrieve that information by name, as shown in Figure 35 (User Stories STUS002 and STUS003).

The retrieved information will then be used to configure the Studio look & feel and other environment customisations. These can also be changed in the studio itself, and a similar procedure can be found for the update and storage of the configurations, as shown in Figure 36 (User Stories STUS006 and STUS007).
4.1.2.2.3 **Browse Resources**

During the development of a vApp there are numerous opportunities where a developer will want to browse what is available in the development environment, namely other vApps and other resources. This will notably happen at the start of a new vApp project, as the developer will check to see if the existing resources can be reused or combined. Figure shows then the retrieval of vf-OS assets. The references for these assets will then be persisted in the local storage (platform).

4.1.2.2.4 **Invoke Plugin Functionalities**

The Studio will include the possibility of working with different editors, but also different compilers, builders, dependency checkers, etc as shown in Figure 38 and in Figure 39. These are plug-in modules that can be invoked by the Studio for performing UI tasks such as editing code, syntax highlighting, auto-completion of code, runtime error checking, dynamic compilation etc. Hence, these plugins also need to be connected to the Studio and requested to be rendered by the Frontend module, as shown in Figure 39. This scenario is one that can be seen in stories STUS152, STUS202, STUS252, STUS302, and STUS352.
Figure 38: Mockup of the vf-OS Studio
4.1.2.3 Interaction description

The following diagram (Figure 40) was taken from the global architecture definition presented in D2.1, and the subsequent text focuses on the interactions and data exchange between the Studio and other vf-OS components.

In order to clarify the interactions between components, the main interactions of the Studio component with other components are:

- **Studio API**: Provides access of the Studio to most of the vf-OS assets, through the vf-OAK SDK, such as:
  - It retrieves configuration files and other stored items from the vf-OS Data Storage
• It retrieves services for data transformation, security and other services provided by the vf-OS framework
• It provides access to the vf-OS available development Enablers
• It provides access to the SDK Builder and Deployment services, allowing the Build of the current vApp
• It provides access to the OAK Developers Engagement Hub functionalities, allowing the callers to be able to retrieve documentation, tutorials, issues and different versions of stored code for a developing unit

• Studio UI: Provides access of the Studio FrontEnd functionalities, which are provided by third-party applications registered to the vf-P as Plugin’s, allowing operations such as:
  • Code Editing
  • Syntax Highlight
  • Code Parsing
  • Debugger
  • Dependency Checks

4.1.3 Frontend Environment

The Frontend Environment provides a set of classes for developers, which are integrated into the OAK Studio. These classes are made to support developers for different use cases: On one hand, UI templates, which use predefined vf-OS styles which are also customisable, on the other hand, behaviour templates that process various operations successively.

4.1.3.1 Behaviour and Functionality

The Frontend Environment provides a set of functionalities that could be grouped around the following features:

• **Error Reporting:** Where errors in vf-OS Assets are recognised and for troubleshooting reasons reports are forwarded to the administration area of the Marketplace. There, developers can examine the error reports to react quickly with bug fixes. Error Reports consist of Information about the sender, client operating system, application, time and the error itself (eg exception stacktrace).

• **User Authorisation and Authentication:** Where the user can register or login to vf-OS easily. A UI is provided with a connection to the security, which handles the registration and login process. For the registration, the user has to enter a valid email address, first name, and last name. The security component then takes over the rest, and replies with a confirmation mail to the entered email address. In the case of a login, only the user credentials (username and password) are necessary. Again, the security component takes over for the verification of the credentials and responds with granted or denied access.

• **Multi Language Support:** Where the user is able to switch and add a language in any vf-OS Asset. Therefore, a resource editor will be provided, which shows all strings in a default language (English) that can then be translated into any language the user speaks or be switched to any language that is already provided. After translation has been performed, the user can request to add the translation to the official application. Then it depends on the developer if the new translation is accepted or declined.
• **Application Logging:** Where internal processes are logged and saved to an additional logfile. This file is then able to give the developer hints about possible error messages and eases troubleshooting.

• **Provide UI Elements:** Where different various UI elements are provided in the vf-Studio to be used by developers. These elements come with its default properties but are also customisable for its respective purpose.

• **Notification Template:** Where developers can provide the functionality to raise notifications, to make users aware of relevant information.

Below is a story map where the primary features, epics and user stories for the Frontend Environment have been identified (see Figure 41).

**Figure 41: Frontend Environment Story Map**

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEUS001 Send Error Report Automatically</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>An error report is sent to the Marketplace backend and can be viewed there</td>
</tr>
<tr>
<td><strong>Who:</strong> Frontend Environment, Marketplace</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Errors are submitted to the Marketplace Backend in order to inform the developer about such errors</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> To increase the quality of vApps in the Marketplace</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>FEUS002 Send Error Report Manually</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Frontend Environment, Marketplace</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Errors are submitted to the Marketplace Backend containing the email address of the user as a contact person in order to inform the developer about such errors</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> To increase the quality of vApps in the Marketplace</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>FEUS003</td>
<td>Create Error Report</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment, vf-OS Asset</td>
<td></td>
</tr>
<tr>
<td>What: Uncaught exceptions are intercepted and afterwards transformed in a readable format</td>
<td></td>
</tr>
<tr>
<td>Why: To increase readability of error reports</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The result of the creation should be a valid report model, that can be parsed into a valid JSON format</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS004</th>
<th>Forward Error Report</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment, Marketplace</td>
<td></td>
</tr>
<tr>
<td>What: Errors are forwarded to the Marketplace Backend containing the email address of the user as a contact person in order to inform the developer about such errors</td>
<td></td>
</tr>
<tr>
<td>Why: To increase the quality of vApps in the Marketplace</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>An error report is sent to the Marketplace backend and can be viewed there including the contact information (email).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS005</th>
<th>Provide a Register form for new Users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Provides a form to register a new user in vf-OS</td>
<td></td>
</tr>
<tr>
<td>Why: To enable a common behaviour of creating new user accounts</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>A user must be created and able to login afterwards</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS006</th>
<th>Provide User Login Form</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Shows a popup of a User Login Form if an action needs to be authorised, and the user is not logged in currently</td>
<td></td>
</tr>
<tr>
<td>Why: To give a common login mask including a behaviour template for “authorised only” actions</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Only authorised users shall be able to take action for specific actions with limited access</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS007</th>
<th>Post User Credentials</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment and Security</td>
<td></td>
</tr>
<tr>
<td>What: The Frontend Environment makes a call to the security component with the entered user credentials in order to authorise the user and gets the user permissions</td>
<td></td>
</tr>
<tr>
<td>Why: To provide the security component with valid user information in order to get a response</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The Frontend Environment should get a response from the security component in order to process the response</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS008</th>
<th>Process User Credentials</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Receives the response from the Security component and evaluates it</td>
<td></td>
</tr>
<tr>
<td>Why: To continue the workflow with permitted access</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The response of the security should contain one of the following: Successful user login including authorised permission to take action Successful user login including unauthorised permission to take action Invalid User Credentials</td>
<td></td>
</tr>
<tr>
<td>FEUS009</td>
<td>Provide a button to reset a password</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Provides a Button which forwards to a form to reset the password</td>
<td></td>
</tr>
<tr>
<td>Why: A user is not forced to create a new user account. They can just reset the password if it is forgotten</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>A new password is sent via mail to the user</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS010</th>
<th>Provide a form to reset a password</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Provides a form to fill in only the email address of the user, which is already registered</td>
<td></td>
</tr>
<tr>
<td>Why: To be able to send a password reset mail to the right user</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The password reset mail is sent to the user</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS011</th>
<th>Translate text</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Provides an option to translate the current language in any other language</td>
<td></td>
</tr>
<tr>
<td>Why: To support the community and decrease the language barrier</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Translation can be saved and viewed afterwards in the vf-OS Asset</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS012</th>
<th>Save Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: The translation of the text can be saved and sent to the developer for integration</td>
<td></td>
</tr>
<tr>
<td>Why: The translation is saved to provide it for the community. The developer must make sure, that the new translation doesn’t contain any explicit illegal content. It should be reviewed by the developer (eg with translation tools such as Google Translate)</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Translation request is sent to the Marketplace Backend successfully</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS013</th>
<th>Switch language</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: The user can choose between several (if provided) languages and select the intended language to use for this vf-OS Asset</td>
<td></td>
</tr>
<tr>
<td>Why: To provide any language for vf-OS Assets</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The text in the vf-OS Asset is changed to the provided translation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS014</th>
<th>Provide UI Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Developers can take UI elements from the UI repository of the FEE to use them in their vf-OS Assets, which are under development</td>
<td></td>
</tr>
<tr>
<td>Why: To ease and accelerate the development of vf-OS Assets</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>UI elements are available in the vf-Studio</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS015</th>
<th>Show Notification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Notification will be shown on a display, in case of a triggered event.</td>
<td></td>
</tr>
<tr>
<td>Why: To make a user aware of noteworthy information</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>A notification is shown when it is expected after a triggered event</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FEUS016</th>
<th>Dismiss Notification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Frontend Environment</td>
<td></td>
</tr>
<tr>
<td>What: Developers can add a dismiss functionality in order to make notifications disappear</td>
<td></td>
</tr>
</tbody>
</table>
4.1.3.2 UI Mockups and Sequence Diagrams

The following sub-sections describe UI mockups and sequence diagrams to enlighten vf-OS internal interactions.

4.1.3.2.1 Report Errors

This feature sends error reports to the developer, so that the developer can react quickly to troubleshoot the application. There, the developer gets information about the user, the used system, and the error message itself. For this feature, an internet connection is necessary, because of a direct communication with the Marketplace.

The main steps/functionaliies are:

- Intercept exceptions during runtime of vf-Assets
- Create an error report with all retrievable information (Sender, Operating System, Error Message, and time)
- Send the error report to the Marketplace, where the developer gets insight in the administration view.

![Sequence Diagram]

*The confirmation will only be sent to the vf-OS Asset if the request was made manually.*

Figure 42: Report Errors Sequence Diagram
The UI for Report Errors is as follows:

![Report Errors UI Mockup](image)

**Figure 43: Report Errors UI Mockup**

4.1.3.2.2 Register User
This feature enables new users to register to vf-OS, which then enables them to log in and use its functionalities. For this, the user only has to enter their first name, last name, and email address. It is also required to accept vf-OS Terms of Service and Privacy Policy.

The main steps/functionalities are:
- Validate and submit user information
- Forward information to the security component, in order to get a confirmation about the registration process

![Register User Sequence Diagram](image)

**Figure 44: Register User Sequence Diagram**
The UI for Register User is as follows:

![Register User UI Mockup](image)

**Figure 45: Register User UI Mockup**

### 4.1.3.2.3 Login User

This feature enables new users to login into vf-OS. After that, the users are able to use all vf-OS functionalities within the scope of their authorisation. For the login, the user only needs to enter its credentials (username and password).

The main steps/functionalities are:
- Receive user credentials, in which the password is of course encrypted
- Forward credentials to the security component, in order to get session information
- Provide session information to the vf-Asset
The UI for Login User is as follows:

![Login UI Mockup](image)

**Figure 47: Login User UI Mockup**

### 4.1.3.2.4 Reset Password

This feature enables users to reset their password in case of a forgotten password. The user will then be provided with a new password via mail by the vf-OS Security component.

The main steps/functionalties are:

- Receive the command to reset the password
- Forward command to the security component, in order to initialise the password reset process
- Provide a confirmation to the vf-Asset
The UI for Reset Password is as follows:

![Reset Password UI Mockup](image)

**4.1.3.2.5 Add new Language**

This feature enables users to add a new language to a vf-Asset. For this, a tool is provided that shows all strings from a vf-Asset and the default language (English) that can be translated. After the user has translated the existing strings, they can save it locally and also send it to the developer to include the translation in the default version of the vf-Asset.

The main steps/functionalities are:
- Translate the current strings into a new language
- Save the translation locally and send it to the developer for a permanent language option
- The developer has to review the new translation and has to make sure that there are no obvious and knowing violations
- The developer has to release a new version of a vf-OS Asset with updated languages
Figure 50: Add new Language Sequence Diagram

The UI for Add new Language is as follows:

![UI Mockup Image]

Figure 51: Add new Language UI Mockup

4.1.3.2.6 Switch Language
This feature enables users to switch the language of a vf-Asset at runtime. The default language is always English, but users are able to add new languages that also can be used by others.

The main steps/functionalities are:
• Choose the language to be used in the vf-Asset
• Translate text into the targeted language

Figure 52: Switch Language Sequence Diagram

The UI for Switch Language is as follows:

Figure 53: Switch Language UI Mockup

4.1.3.2.7 Manage Notification
This feature enables users to get notifications of important messages or pending interactions. The developer can easily integrate this behaviour via the Notification Template.

The main steps/functionalities are:

- Show the notification on the screen
- Remove the notification from the screen
- Set a reminder for a later appearance

![Sequence Diagram]

Figure 54: Manage Notification Sequence Diagram

The UI for Switch Language is as follows:
4.1.3.3 Interaction description

The following figure depicts how the Frontend Environment interacts with other vf-OS components and also shows the data structures of the exchange data.

4.1.3.3.1 Report Errors

Error Reports are sent from a Frontend Environment Behaviour Template to the marketplace. At first, all important and retrievable information are gathered and processed to create a new Report. The retrievable information consists of:

- **Sender**: The information about the user, who discovered the error to eventually contact them for further information.
- **AppInformation**: Contains information of the vf-Asset to identify the exact version of the software. Sometimes old versions are in circulation where bugfixes are already performed in an actual version.
- **ClientOS**: Contains information about the environment of the vf-Asset to eventually detect incompatibilities with Operating System updates or major versions.
- **ExceptionInfo**: contains information about the exception that has been raised and caused this error. This helps the developer to detect the exact location of the bug.

![Figure 57: Report Errors Information Model](image)

### 4.1.3.3.2 Login User
The grant access for users, they have to log in first with user credentials. The user credentials are then sent to the vf-OS Security component, which takes over the authentication and authorisation. The user credentials consist of:

- **Name**: represents the username of the user. Alternatively, the user can also use its email address
- **Password**: an encrypted secret of the user, which is needed to grant access to the intended user account

![Figure 58: Login User Information Model](image)

### 4.1.3.3.3 Register User
For the registration of new users, some user information is required. The user information is needed to identify the user and to have a contact for notifications and other emails. The user information consists of:

- **UserID**: Represents the unique identifier of a user. This ID is set by the security component.
- **Email**: the contact address of a user, which all notifications and messages are sent to
• **LastName**: Combined with the first name to get the complete real name of a user
• **FirstName**: Combined with the last name to get the complete real name of a user

![User Information Model](image)

**Figure 59: Register User Information Model**

### 4.1.4 Process Enabler Designer

The Process Designer is responsible for allowing users to model multiple manufacturing workflows so orchestrating the various assets available within a collaborative framework. The tool will be a reactive, extensible, and online workspace supporting a BPMN-like model and be utilisable by vApps where process design and orchestration is appropriate.

#### 4.1.4.1 Behaviour and Functionality

The Process Designer component provides a set of functionality that broadly is as follows:

- **BPMN2.0 Modelling and Rendering Service**: This service will provide the means of rendering and modelling a process in BPMN format. It will connect to the Storage component for saving and retrieving a process model for designing and editing. Once developed, the BPMN will be deployed as an asset within a relevant Asset (eg vApp) through the Studio. At runtime, the Asset will run the BPMN via the Process Execution engine. It will also have the responsibility to make sure that the process model is a valid BPMN.

- **Process Toolbox Service**: The Process Toolbox service has the functionality of rendering the toolbox elements. It will obtain the BPMN elements from the BPMN2.0 Modelling and Rendering Service. These elements will include Flow Objects (Events, Activities, and Gateways), Data, Connecting Objects, Swim lanes, and Artefacts. It will also have the ability to make a call to the Assets Store to get a list of assets that can be used in the design of a process.
The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDUS001</td>
<td><strong>Open Process Designer</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Process Designer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Open Process Designer UI</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the UI can be loaded by the vf-Studio</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The Process Designer is successfully opened</td>
<td></td>
</tr>
</tbody>
</table>

| PDUS002 | **Load UI elements** |
| **Description** |
| **Who:** Process Designer |
| **What:** Load the UI elements into the main UI |
| **Why:** So that the developer can access the Process Designer elements |
| **Acceptance Criteria** |
| The developer can access every single element from the UI |

| PDUS003 | **Credentials for the vf-OS Storage** |
| **Description** |
| **Who:** Process Designer |
| **What:** Provide credentials to connect to the vf-OS Storage |
| **Why:** So that the developer can access to those assets stored in the vf-OS Storage |
| **Acceptance Criteria** |
| The Process Designer is authorised to access the vf-OS Storage |

<p>| PDUS004 | |
| <strong>Description</strong> |</p>
<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Who: Process Designer</th>
<th>What: Connect to the vf-Store</th>
<th>Why: So that the developer can access to those assets published in the vf-Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDUS005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Process Designer has access to the vf-Store</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDUS006</td>
<td><strong>Description</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Who: Process Designer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Connect to the SDK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that the developer can have access to SDK assets to design models</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Process Designer has access to the SDK libraries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDUS007</td>
<td><strong>Description</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Who: Process Designer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Validate the model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: Validation, ie to check the construct against BPMN Construct Specifications</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The model is compared against the Specification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDUS008</td>
<td><strong>Description</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Who: Process Designer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Notify the developer that a model (or a part) were not created</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that the developer can make use of the feedback and prevent invalid models</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The model is not saved, and the user is notified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDUS009</td>
<td><strong>Description</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Who: Process Designer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Deploy a process model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that the developer can embed the designed model in a vApp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The model is deployed to the vf-Studio</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDUS002</td>
<td><strong>Description</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Who: Process Designer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Open Process Model (Read BPMN from XML File)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that the Process Model (formed of BPMN Elements) can be visualised by the developer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The BPMN elements can be pulled to Canvas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDUS010</td>
<td><strong>Description</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Who: Process Designer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Place BPMN Elements in Canvas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that the developer can select a BPMN element and add it to the canvas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The UI supports the BPMN, they're visible for the developer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Who:</td>
<td>What:</td>
<td>Why:</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
<td>---------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Filter BPMN Elements</td>
<td>Who: Process Designer</td>
<td>Filter between different Element Types</td>
<td>So that the toolbox can be sorted, searched and filtered</td>
<td></td>
</tr>
<tr>
<td>PDUS012 Select BPMN Element</td>
<td>Description</td>
<td>Who: Process Designer</td>
<td>What: Get the details of the selected BPMN element</td>
<td>Why: So that the user has control over the Element</td>
</tr>
<tr>
<td>PDUS013 Remove Selected Element</td>
<td>Description</td>
<td>Who: Process Designer</td>
<td>What: Remove the Selected Element</td>
<td>Why: So that the developer is able to remove the selected element</td>
</tr>
<tr>
<td>PDUS014 Renders Property UI</td>
<td>Description</td>
<td>Who: Process Designer</td>
<td>What: Render the Property UI</td>
<td>Why: So that the developer can have access to the properties of a given element</td>
</tr>
<tr>
<td>PDUS015 Loads Value of Properties</td>
<td>Description</td>
<td>Who: Process Designer</td>
<td>What: Retrieve property values from the process model</td>
<td>Why: So that the values of the property are shown</td>
</tr>
<tr>
<td>PDUS016 Connect to Storage</td>
<td>Description</td>
<td>Who: Process Designer</td>
<td>What: To connect to the Process Models Store</td>
<td>Why: So that the developer can access for CRUD operations</td>
</tr>
<tr>
<td>PDUS017 Create Process Model</td>
<td>Description</td>
<td>Who: Process Designer</td>
<td>What: To create a Process Model</td>
<td>Why: So that a process model can be persisted for future reference</td>
</tr>
<tr>
<td>PDUS018 Connect to Storage</td>
<td>Description</td>
<td>Who: Process Designer</td>
<td>What: To connect to the Process Models Store</td>
<td>Why: So that the developer can access for CRUD operations</td>
</tr>
</tbody>
</table>

Acceptance Criteria

The BPMN Elements can be filtered
The Developer can get the selected element
The developer can remove the selected element
The UI is loaded and visible
The Values are visible and accessible from the Property UI
The Process Designer is connected to the Process Models Store
The developer is able to create a Process Model and save it on the Store
<table>
<thead>
<tr>
<th>PDUS019</th>
<th>Read Process Model</th>
</tr>
</thead>
</table>
| **Who:** Process Designer  
**What:** To read a Process Model  
**Why:** So that the process model can be loaded in the Designer |
| **Acceptance Criteria:**  
The read process model is loaded into the canvas |

<table>
<thead>
<tr>
<th>PDUS020</th>
<th>Search Process Model</th>
</tr>
</thead>
</table>
| **Who:** Process Designer  
**What:** To search process models in the Store  
**Why:** So that the developer can access in a faster way to the desired model |
| **Acceptance Criteria:**  
A search criteria can be introduced and the Store retrieves those models matching the criteria |

<table>
<thead>
<tr>
<th>PDUS021</th>
<th>Connect to Storage</th>
</tr>
</thead>
</table>
| **Who:** Process Designer  
**What:** To connect to the Process Models Store  
**Why:** So that the developer can access for CRUD operations |
| **Acceptance Criteria:**  
The Process Designer is connected to the Process Models Store |

<table>
<thead>
<tr>
<th>PDUS022</th>
<th>Update Process Model</th>
</tr>
</thead>
</table>
| **Who:** Process Designer  
**What:** To update a process model  
**Why:** So that updates to a model are saved overwriting the previous version |
| **Acceptance Criteria:**  
The Process Model in the Store is updated |

<table>
<thead>
<tr>
<th>PDUS023</th>
<th>Versioning</th>
</tr>
</thead>
</table>
| **Who:** Process Designer  
**What:** To manage versions on a single process model  
**Why:** So that new versions of a model are saved keeping older versions untouched |
| **Acceptance Criteria:**  
A new version of the Process Model is created |

<table>
<thead>
<tr>
<th>PDUS024</th>
<th>Connect to Storage</th>
</tr>
</thead>
</table>
| **Who:** Process Designer  
**What:** To connect to the Process Models Store  
**Why:** So that the developer can access for CRUD operations |
| **Acceptance Criteria:**  
The Process Designer is connected to the Process Models Store |

<table>
<thead>
<tr>
<th>PDUS025</th>
<th>Search Process Model</th>
</tr>
</thead>
</table>
| **Who:** Process Designer  
**What:** To search process models in the Store  
**Why:** So that the developer can access in a faster way to the desired model |
| **Acceptance Criteria:**  
A search criteria can be introduced and the Store retrieves those models matching the criteria |

<table>
<thead>
<tr>
<th>PDUS026</th>
<th>Description</th>
</tr>
</thead>
</table>
Delete Process Model

Who: Process Designer
What: To delete a process model from the Store
Why: So that the developer can maintain and remove outdated Models, incorrect models, etc

Acceptance Criteria
The Process Model is deleted

4.1.4.2 UI mockups and Sequence Diagrams
The following subsections describe the UI mockups and sequence diagrams describing the interaction between the sections of the Process Designer.

4.1.4.2.1 Load the User Interface
This feature provides the capability to load the UI of the Process Designer.

The main steps/functionalities are as follows:
- Open Process Designer
- Load UI Elements
- Credentials for the vf-OS Storage
- Connect to the vf-Store
- Connect to the SDK

![Sequence Diagram]

Figure 61: Load User Interface Sequence Diagram
The UI for the Process Designer is as follows:

![Process Designer UI](image)

**Figure 62: Process Designer UI**

The UI for the Process Models Store, for retrieving Process Models, is as follows:

![Process Model Store UI](image)

**Figure 63: Process Model Store UI**
The following UI is how the selection for the Tasks is available in the BPMN Toolbox:

![BPMN Service Task Selection](image)

**Figure 64: BPMN Service Task Selection**

This is the UI of a collapsible process, with both the collapsed version, and the expanded version:

![A BPMN Process Showing Collapsible Processes](image)

**Figure 65: A BPMN Process Showing Collapsible Processes**

### 4.1.4.2.2 Validate User Operations

This feature allows the file to be validated against the Process Specification. It will inform the user when they have committed an error in the Process Model and when they have made a successful process model. The steps include:

- Validate User Constructs Against Specifications
- Invalid Constructs are not created
4.1.4.2.3 **Deploy Process Model**

This feature allows a process model to be deployed as part of a vApp, via the vf-Studio. The main steps/functionalities are as follows:

- **Deploy a Process Model**

  1. Open Process Model (Read BPMN from XML file)
  2. Place BPMN Elements in Canvas
  3. Filter BPMN Elements
4.1.4.2.5 Remove BPMN Elements
This feature removes a BPMN Element from the canvas. This step includes:

- Select BPMN Element
- Remove Selected BPMN Element

4.1.4.2.6 Show Property Values
This feature reads and displays the properties of a BPMN element so that the user can view them, or edit them. The steps include:
- Renders Property UI
- Load Value of Property

4.1.4.2.7 Create Process Model

This feature creates the initial BPMN model ready for user interaction. The steps include:

- Connect to Storage
- Create Process Model

Figure 70: Show Property Values Sequence Diagram
4.1.4.2.8 Read Process Model
This feature loads a previously created BPMN model. This step includes:

- Connect to Storage
- Read Process Model
- Search Process Model
Figure 72: Read Process Model Sequence Diagram
4.1.4.2.9 Update Process Model
This feature updates a previously saved model, either overwriting the previous version or creating a new version. The steps include:

- Connect to Storage
- Update Process Model (overwrites previous version)
- Versioning (creates a new version)

Figure 73: Update Process Model Sequence Diagram
4.1.4.2.10 **Delete Process Model**
This feature deletes a selected process model and informs the user. The steps include:

- Connect to Storage
- Search Process Model
- Delete Process Model

![Delete Process Model Diagram](image)

**Interaction description**
From the previous description of the functionality covered by the Process Designer component, a deeper level of detail regarding the main modules of the component and the interaction between those modules and other vf-OS components emerges. Whilst the next Figure 76 shows the Architecture diagram, as presented in D2.1, the accompanying text focuses on the interactions and data exchange between the Process Designer and other vf-OS components.
In order to clarify the interactions between components, the information exchanged between process designer subcomponents and other components has been detailed (see). The main interactions of Process Designer component’s classes with other components are:

- **Process Toolbox Services**: This component is in charge of pulling assets together so that the other sub-components have one place to access them. The main information flows are:
  - It receives the assets from the Process Assets API (Interaction with the Marketplace component)
  - It receives the sdkAssets from the SDK Assets API (Interaction with the SDK component)
  - It sends the elements to the Toolbox Frontend UI (Interaction with the Studio component)
  - It sends the BPMN elements to the BPMN 2.0 Modelling and Rendering Service (Interaction with the Storage and Studio components)

- **BPMN 2.0 Modelling and Rendering Service**: This component renders the BPMN elements to the Process Designer UI, and is responsible for APIs and components interacting with the created process models. The main flow of information includes:
  - It receives the BPMN elements from the Process Toolbox Services (Interaction with the Storage and Marketplace Components)
4.1.5 Data Mapping

The Data Mapping component provides the features to define Manufacturing Maps that will allow the transformation and integration of data. It also provides basic functionalities for semantic homogenisation in a context of heterogeneous data. The developed application will enable a business analyst driven approach for the automatic linking of organisations’ data schemas to the reference data model. vf-OS, the Data Mapping, and the vApps will use publicly available manufacturing ontologies as reference data model so an evolutionary data model can be supported in the form of crowdsourcing techniques. The Manufacturing Maps will be available in the vf-OS Data Storage and exported as Transformation Services of particular valuable use for the Process Designer.

4.1.5.1 Behaviour and Functionality

The Data Mapping component provides a set of functionality as follows:

- **Maps Designer:** Where the Manufacturing Maps can be generated. A Manufacturing Map file describes the rules to be executed to transform a specific syntax format A into format B which could then, for example, be used as part of a process. It will offer the user the possibility to annotate these maps with additional semantic metadata.
• **Linked Data**: Where Linked Data functionality on top of the reference data model is offered. It also supports crowdsourcing functionalities to update the reference data model.

• **Ontology Management**: Where ontological (concepts in OWL2, RDFS) and linked (RDF) datasets can be managed. It provides functionality for generic data (CRUD) management of the content stored in its semantic backend in the vf-OS Storage.

Below is a story map where the main features, epics and user stories for the Data Mapping component have been identified (see Figure 78).

### Figure 78: Data Mapping Story Map

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DMUS001</strong></td>
<td><strong>Connect to Datasource</strong></td>
</tr>
<tr>
<td><strong>Who</strong>: Data Mapping</td>
<td><strong>Description</strong>&lt;br&gt;<strong>What</strong>: Open filesystem and access the schema of the datasource (both for source and target schemas)&lt;br&gt;<strong>Why</strong>: So that the schema can be loaded and, thus, the mappings can be performed</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The schemas (source and/or target) are successfully retrieved</td>
</tr>
<tr>
<td><strong>DMUS002</strong></td>
<td><strong>Read XML</strong></td>
</tr>
<tr>
<td><strong>Who</strong>: Data Mapping</td>
<td><strong>Description</strong>&lt;br&gt;<strong>What</strong>: Interprets eg XML schema files&lt;br&gt;<strong>Why</strong>: So that the schema can be loaded and, thus, the mappings can be performed</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The eg XML schema (source and/or target) is successfully interpreted and</td>
</tr>
<tr>
<td>DMUS003</td>
<td>Read CSV</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Interprets CSV schema files</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the schema can be loaded and, thus, the mappings can be performed</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The CSV schema (source and/or target) is successfully interpreted and retrieved</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMUS004</th>
<th>Read JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Interprets JSON schema files</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the schema can be loaded and, thus, the mappings can be performed</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The JSON schema (source and/or target) is successfully interpreted and retrieved</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMUS005</th>
<th>Read TXT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Interprets plain text files</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the file can be loaded and, thus, the mappings can be performed</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The plain text schema (source and/or target) is successfully interpreted and retrieved</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMUS006</th>
<th>Read XLS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Interprets Excel (XLS) files</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the Excel spreadsheets can be loaded and, thus, the mappings can be performed</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The XLS file (source and/or target) is successfully interpreted and retrieved</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMUS007</th>
<th>Read MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Interprets MySQL schemas</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the content of MySQL schema can be loaded and, thus, the mappings can be performed</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The MySQL schema (source and/or target) is successfully interpreted and retrieved</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMUS008</th>
<th>Display UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> show Mapping UI</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the source schema can be displayed and, thus, the mappings can be performed</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The Mapping UI is successfully shown</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMUS009</th>
<th>Load Source Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Loads source schema</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the source schema can be accessed</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The source schema is successfully loaded</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DMUS010</th>
<th>Display Source Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Display source schema</td>
<td></td>
</tr>
</tbody>
</table>
### Why:
The source schema can be viewed

#### Acceptance Criteria
The source schema is successfully displayed

### DMUS011 Analyse Source Schema

#### Description
Who: Data Mapping  
What: Analyses source schema  
Why: So that the source schema can be manipulated

#### Acceptance Criteria
The source schema is successfully analysed

### DMUS012 Connect to ontology

#### Description
Who: Data Mapping  
What: Connects to domain (or vf-OS) ontology  
Why: So that alternative concepts can be suggested for the concepts present in the source schema

#### Acceptance Criteria
The ontology is successfully connected

### DMUS013 Suggest Semantic Concepts for Source Schema

#### Description
Who: Data Mapping  
What: Suggests alternative (or linked) concepts to the concepts present in the source schema  
Why: So that a crowdsourced domain (or vf-OS) ontology can be populated

#### Acceptance Criteria
Relevant concepts are suggested for a given concept

### DMUS014 Display UI

#### Description
Who: Data Mapping  
What: Show Mapping UI  
Why: So that the target schema can be displayed and, thus, the mappings can be performed

#### Acceptance Criteria
The Mapping UI is successfully shown

### DMUS015 Load Target Schema

#### Description
Who: Data Mapping  
What: Loads target schema  
Why: So that the target schema can be accessed

#### Acceptance Criteria
The target schema is successfully loaded

### DMUS016 Display Target Schema

#### Description
Who: Data Mapping  
What: Display target schema  
Why: So that the target schema can be viewed

#### Acceptance Criteria
The target schema is successfully displayed

### DMUS017 Analyse Target Schema

#### Description
Who: Data Mapping  
What: Analyses target schema  
Why: So that the target schema can be manipulated

#### Acceptance Criteria
The target schema is successfully analysed

### DMUS018 Connect to ontology

#### Description
Who: Data Mapping  
What: Connects to domain (or vf-OS) ontology  
Why: So that alternative concepts can be suggested for the concepts present in the target schema
<table>
<thead>
<tr>
<th>Description</th>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DMUS019 Suggest Semantic Concepts for Target Schema</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The ontology is successfully connected</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Suggests alternative (or linked) concepts to the concepts present in the target schema</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that a crowdsourced domain (or vf-OS) ontology can be populated</td>
<td></td>
</tr>
<tr>
<td><strong>DMUS020 Connect to Storage</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Relevant concepts are suggested for a given concept</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Connect to the vf-OS Data Storage with the credentials as directed by the vf-OS Security component</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the Maps can be read, searched and filtered after save</td>
<td></td>
</tr>
<tr>
<td><strong>DMUS021 Read file</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The Data Storage is accessible</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Read file</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the Maps can be loaded into the Data Mapping component</td>
<td></td>
</tr>
<tr>
<td><strong>DMUS022 Connect to Storage</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The file is successfully read</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Connect to the vf-OS Data Storage with the credentials as directed by the vf-OS Security component</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the Maps can be retrieved after save</td>
<td></td>
</tr>
<tr>
<td><strong>DMUS023 Search Map</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The Data Storage is accessible</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Search map</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the Maps can be searched into the Data Storage component</td>
<td></td>
</tr>
<tr>
<td><strong>DMUS024 Filtering in Storage</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The list of maps resulted after the search matches the searching criteria specified</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Apply filters</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the Maps stored in the Data Storage can be filtered according to a user specified criteria</td>
<td></td>
</tr>
<tr>
<td><strong>DMUS025 Preview Map</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The list of maps is filtered out with the specified criteria</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> preview map</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> so that the Maps can be graphically previewed before loading into the Data Mapping component</td>
<td></td>
</tr>
<tr>
<td><strong>DMUS026 Connect to Storage</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The map is graphically viewed by the user</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Data Mapping</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Connect to the vf-OS Data Storage with the credentials as directed by the vf-OS Security component</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> So that the Maps can be stored/persisted</td>
<td></td>
</tr>
<tr>
<td><strong>DMUS027</strong></td>
<td>Annotate Map</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| **Description** | Who: Data Mapping  
What: Annotate map with metadata  
Why: So that the Maps can be searched and filtered with this metadata as parameters |
| **Acceptance Criteria** | The file is successfully annotated and the metadata is stored along with the map file |

<table>
<thead>
<tr>
<th><strong>DMUS028</strong></th>
<th>Serialise Map</th>
</tr>
</thead>
</table>
| **Description** | Who: Data Mapping  
What: The map is serialised to a computer readable format  
Why: So that the Maps can be persisted in the vf-OS Data Storage |
| **Acceptance Criteria** | The file is successfully serialised and it is ready for persisting |

<table>
<thead>
<tr>
<th><strong>DMUS029</strong></th>
<th>Persist Map</th>
</tr>
</thead>
</table>
| **Description** | Who: Data Mapping  
What: Save the map in the Data Storage  
Why: So that the Maps can be re-used, retrieved, removed, searched, and filtered |
| **Acceptance Criteria** | The file is successfully persisted in the vf-OS Data Storage |

<table>
<thead>
<tr>
<th><strong>DMUS030</strong></th>
<th>Connect to Storage</th>
</tr>
</thead>
</table>
| **Description** | Who: Data Mapping  
What: Connect to vf-OS Data Storage with the credentials as directed by the vf-OS Security component  
Why: So that the Maps can be removed from the storage |
| **Acceptance Criteria** | The Data Storage is accessible |

<table>
<thead>
<tr>
<th><strong>DMUS031</strong></th>
<th>Search Map</th>
</tr>
</thead>
</table>
| **Description** | Who: Data Mapping  
What: Search map  
Why: So that the Maps can be searched into the Data Storage component |
| **Acceptance Criteria** | The list of maps resulted after the search matches the searching criteria specified |

<table>
<thead>
<tr>
<th><strong>DMUS032</strong></th>
<th>Delete Map</th>
</tr>
</thead>
</table>
| **Description** | Who: Data Mapping  
What: Remove map  
Why: So that the persisted Maps in the vf-OS Data Storage can be removed from it |
| **Acceptance Criteria** | The selected map(s) is successfully removed from the Data Storage |

<table>
<thead>
<tr>
<th><strong>DMUS033</strong></th>
<th>Annotate Service</th>
</tr>
</thead>
</table>
| **Description** | Who: Data Mapping  
What: Annotate map for publication  
Why: So that the to-be deployed Map can be easily found in the vf-Store |
| **Acceptance Criteria** | The active map is successfully annotated |

<table>
<thead>
<tr>
<th><strong>DMUS034</strong></th>
<th>Create Service</th>
</tr>
</thead>
</table>
| **Description** | Who: Data Mapping  
What: Create self-executing service  
Why: So that the deployed Map can be executed as a stand-alone service |
<p>| <strong>Acceptance Criteria</strong> |  |</p>
<table>
<thead>
<tr>
<th>DMUS035 Deploy Service</th>
<th>The transformation service is successfully created from the active map, together with its annotations</th>
</tr>
</thead>
</table>
| **Description**        | **Who:** Data Mapping  
                           **What:** Create Docker package  
                           **Why:** So that the deployed Map can be executed and scalable if necessary as a stand-alone service |
| **Acceptance Criteria**| The active map, together with its annotations, is successfully packaged as Docker container |

<table>
<thead>
<tr>
<th>DMUS036 Connect to vf-Store</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Who:** Data Mapping  
                           **What:** Connect to vf-Store with the credentials as directed by the vf-OS Security component  
                           **Why:** So that the Transformation Services (ie the deployed maps) can be published |
| **Acceptance Criteria**     | The vf-Store is accessible |

<table>
<thead>
<tr>
<th>DMUS037 Publish Deployed Map</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Who:** Data Mapping  
                           **What:** Publish deployed map (ie transformation service)  
                           **Why:** So that the Transformation Service can be sold to vf-OS Users and re-used within eg the Process Designer |
| **Acceptance Criteria**     | The service is successfully published in the vf-Store |

<table>
<thead>
<tr>
<th>DMUS038 Get Linked Concepts</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Who:** Data Mapping  
                           **What:** Obtain Linked Concepts to a given concept passed as parameter  
                           **Why:** So that the Data Mapping can make use of the "wisdom of the crowd" for developing the maps |
| **Acceptance Criteria**     | A set of relevant Linked Concepts is made available |

<table>
<thead>
<tr>
<th>DMUS039 Get Link</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Who:** Data Mapping  
                           **What:** Obtain Links between a set of given concepts passed as parameters  
                           **Why:** So that the Data Mapping can make use of the "wisdom of the crowd" for developing the maps |
| **Acceptance Criteria** | A set of Links is made available |

<table>
<thead>
<tr>
<th>DMUS040 Is Linked</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Who:** Data Mapping  
                           **What:** Obtain whether two given concepts, passed as parameters, are linked  
                           **Why:** So that the Data Mapping can make use of the "wisdom of the crowd" for developing the maps |
| **Acceptance Criteria** | A set of links (with the number of hops between them) is made available |

<table>
<thead>
<tr>
<th>DMUS041 Add Linked Concept</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Who:** Data Mapping  
                           **What:** Add a new concept that is linked to a given concept passed as parameter  
                           **Why:** So that the Data Mapping can provide feedback to the "wisdom of the crowd" for developing future maps |
| **Acceptance Criteria** | The ontology is updated with the provided content |

| DMUS042 | Description |
### Add Link

**Who:** Data Mapping  
**What:** Add a new link between two given concepts, passed as parameters  
**Why:** So that the Data Mapping can provide feedback to the “wisdom of the crowd” for developing future maps  

**Acceptance Criteria**  
The ontology is updated with the provided content

| **DMUS043** | **Update Linked Concept** | **Description**  
| Who: Data Mapping  
What: Update a concept that is linked to a given concept passed as parameter  
Why: So that the Data Mapping can provide feedback to the “wisdom of the crowd” for developing future maps  

**Acceptance Criteria**  
The ontology is updated with the provided content

### Update Linked Concept

| **DMUS044** | **Update Link** | **Description**  
| Who: Data Mapping  
What: Update a given link between two concepts, passed as parameters  
Why: So that the Data Mapping can provide feedback to the “wisdom of the crowd” for developing future maps  

**Acceptance Criteria**  
The ontology is updated with the provided content

### Get Concept

| **DMUS045** | **Get Concept** | **Description**  
| Who: Data Mapping  
What: get concepts from the ontology  
Why: so that the Data Mapping can make use of the vf-OS Ontology for developing the maps  

**Acceptance Criteria**  
The vf-OS Ontology provides with the requested concepts

### Add Concept

| **DMUS046** | **Add Concept** | **Description**  
| Who: Data Mapping  
What: Add concepts to the ontology  
Why: So that the Data Mapping can make use of the vf-OS Ontology for developing the maps  

**Acceptance Criteria**  
The vf-OS Ontology is updated with new concepts

### Update Concept

| **DMUS047** | **Update Concept** | **Description**  
| Who: Data Mapping  
What: Update concepts in the ontology  
Why: So that the Data Mapping can make use of the vf-OS Ontology for developing the maps  

**Acceptance Criteria**  
The vf-OS Ontology is updated with new metadata about existing concepts

### Delete Concept

| **DMUS048** | **Delete Concept** | **Description**  
| Who: Data Mapping  
What: remove concepts from the ontology  
Why: so that the Data Mapping can update the vf-OS Ontology for future developing the maps  

**Acceptance Criteria**  
The vf-OS Ontology is updated with the removal of existing concepts

### Reasoning

| **DMUS049** | **Reasoning** | **Description**  
| Who: Data Mapping  
What: Reason  
Why: So that the Data Mapping can make use of the vf-OS Ontology for developing the maps  

**Acceptance Criteria**  
The vf-OS Ontology provides with a set of reasoned objects as per the
4.1.5.2 UI Mockups and Sequence Diagrams

The following sub-sections describe the UI mockups and sequence diagrams of the Data Mapping component.

4.1.5.2.1 Read Datasources

This feature provides the capability to read a set of types of datasources that will be used when performing the mapping task.

The main steps/functionalities are as follows:

- Connect to Datasource
- Read XML
- Read CSV
- Read JSON
- Read TXT
- Read XLS
- Read MySQL Database

Figure 79: Read Datasources Sequence Diagram
The UI for reading datasources is as follows:

![Read Datasources UI Mockup](image)

**Figure 80: Read Datasources UI Mockup**

### 4.1.5.2.2 Read Source Schema and Read Target Schema

This feature provides the capability to read the source and the target schemas that will be used when performing the mapping task.

The main steps/functionalities are as follows:
- Display UI
- Load Source/Target Schema
- Display Source/Target Schema
- Analyse Source/Target Schema
- Connect to Ontology
- Suggest Semantic Concepts for Source/Target Schema

![Read Source Schema Sequence Diagram](image)

**Figure 81: Read Source Schema Sequence Diagram**
Figure 82: Read Target Schema Sequence Diagram

The UI for reading the source and target schemas is as follows:

Figure 83: Read Source/Target Schemas UI Mockup

4.1.5.2.3 Manage Maps
This feature provides the capability to read, retrieve, store and delete a Manufacturing Map from the vf-OS Data Storage.

The main steps:functionalities are as follows:

- Read Map
  - Connect to Storage
  - Read file
- Retrieve Map
  - Connect to Storage
- Search Map
- Filtering in Storage
- Preview Map

- Store Map
  - Connect to Storage
  - Annotate Map
  - Serialise Map
  - Persist Map

- Delete Map
  - Connect to Storage
  - Search Map
  - Delete Map

Figure 84: Read Map Sequence Diagram
Figure 85: Retrieve Map Sequence Diagram

Figure 86: Store Map Sequence Diagram
4.1.5.2.4 Deploy and Publish Map

These features provide the capability to deploy and publish a map after it has been generated by the Business Analyst.

The main steps/functionalities are as follows:

- **Deploy Map**
  - Annotate Service
  - Create Service
  - Deploy Service

- **Publish Map**
  - Connect to vf-Store
  - Publish Deployed Map
Figure 88: Deploy Map Sequence Diagram

Figure 89: Publish Map Sequence Diagram
The UI for deploying and publishing maps is as follows:

![Deploy and Publish Map UI Mockup](image)

**Figure 90: Deploy and Publish Map UI Mockup**

### 4.1.5.2.5 Linked Concepts
This feature provides the capability to access the Linked Concepts functionality that the Data Mapping is offering to the Business Analyst when generating their Manufacturing Maps.

The main steps / functionalities are as follows:

- Get Linked Concepts
- Get Link
- Is Linked
- Add Linked Concept
- Add Link
- Update Linked Concept
- Update Link
Figure 91: Obtain Linked Concepts and/or Links Sequence Diagram

Figure 92: Add Linked Concept Sequence Diagram

Figure 93: Add Link Sequence Diagram
The UIs for accessing the Linked Concepts functionality are as follows:

**Figure 94: Update Linked Concept Sequence Diagram**

**Figure 95: Update Link Sequence Diagram**

**Figure 96: Search for Candidates UI Mockup**
4.1.5.2.6 Manage Ontology
This feature provides the capability to access the domain ontology functionality that the Data Mapping is offering to the Business Analyst when generating their Manufacturing Maps.

The main steps/functionality are as follows:
- Get Concept
- Add Concept
- Update Concept
- Delete Concept
- Reasoning
**Figure 98: Get Concept Sequence Diagram**

**Figure 99: Add Concept Sequence Diagram**

**Figure 100: Update Concept Sequence Diagram**

**Figure 101: Delete Concept Sequence Diagram**
4.1.5.3 Interaction description

From the previous description of the functionality covered by the Data Mapping component, a deeper level of detail regarding the main modules of the component and the interaction between those modules and other vf-OS components emerges. Whilst the next Figure 103 shows the Architecture diagram, as presented in D2.1, the accompanying text focuses on the interactions and data exchange between the Data Mapping and other vf-OS components.
The main interactions of Data Mapping modules with other components are:

- **Maps Designer:** Is the module in charge of offering the functionality of designing maps to the developer. This module interacts directly with the Mapping UI. These maps, through the corresponding APIs, are stored in the vf-OS Storage (via Storage Map API) and deployed in the vf-Store for future usage (via the Publish Map API). The main information flows are:
  - It sends the deployed map to the vf-Store (interaction with Marketplace component)
  - It sends and receives map when this is saved or retrieved from the vf-OS Storage (interaction with Storage component)

**Figure 103: Data Mapping Component Interaction Diagram**
• It sends a query to the Ontology Manager to retrieve concepts

• Linked Data Services: The module in charge of querying the ontology for receiving Linked Data information. The main information flows exchanged with external components are:
  • It sends the linkedConcepts reasoned from the Ontology Manager (interaction with Any vf-OS component needing this functionality)

• Ontology Manager: The module in charge of managing the ontologies that the Data Mapping component will be querying. The main information flows exchanged with external components are:
  • It sends and receives ontologies for reasoning (interaction with Storage component)

Figure 104: Data Mapping Component Classes and Information Exchanged
4.2 Engagement

4.2.1 Developer Engagement Hub

4.2.1.1 Behaviour and Functionality

The vf-OS Developer Engagement Hub is an environment whose purpose is to foster and promote the productivity of developers, whether by integrating the developed Open Source code of previously developed vApps, but also including a set of other tools to encourage development community building such as wikis, issue trackers, forums, and blogs. It supports the continuous integration of the developed vApps by triggering automated tests upon the commit of versions in the version control environment. Additionally, it is planned to allow the definition of an organised hierarchy of development projects to cope with complex environment structures that include multiple vApps.

Figure 105: Developer Engagement Hub Story Map

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
</table>
| DEUS001 Log in to DE Frontend UI | **Description**
Who: vApps Developer
What: log in the Developer Engagement Hub's Frontend UI
Why: to access the Hub services

**Acceptance Criteria**
Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment "Login User")

<table>
<thead>
<tr>
<th>DEUS002 Get Configuration</th>
<th>Description</th>
</tr>
</thead>
</table>
| Who: vApps Developer      | What: will list existing configurations for the DE
Why: select the most suitable DE configuration for the particular community building need. This may include access to different suites of DE tools, look & feel customisation, or other types of configuration (defaults) for each created DE project

**Acceptance Criteria**
Invoker got a structured list of the DE configurations
| DEUS003  | Set Configuration | Description | Who: vApps Developer  
What: will select an existing configuration for the Developer Engagement Hub  
Why: select one DE configuration for building the community. This may include access to different suites of tools, look & feel customisation, or other types of configuration  
Acceptance Criteria | The DE will be customised in accordance with the selected configuration  
A file with the DE configuration will be stored in the platform connected with the user session |
|------------|------------------|-------------|--------------------------|------------------------------------------------------------------|
| DEUS005  | Log in to DE Frontend UI | Description | Who: vApps Developer  
What: log in the Developer Engagement Hub's Frontend UI  
Why: to access the Hub services  
Acceptance Criteria | Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment "Login User") |
| DEUS006  | Edit Configuration | Description | Who: vApps Developer  
What: edit and update the configuration of the Developer Engagement Hub  
Why: customise the most suitable DE configuration for building the community. This may include access to different suites of tools, look & feel customisation, or other types of configuration  
Acceptance Criteria | Invoker accesses a UI frontend with the current DE configuration and is able to change it. The DE will be customised to reflect the updated configuration  
A file with the DE configuration will be stored in the platform connected with the user session |
| DEUS007  | Store Configuration | Description | Who: vApps Developer  
What: will store the existing configurations for the Developers Engagement Hub  
Why: for reuse purposes on later executions  
Acceptance Criteria | Invoker got a confirmation that the configuration was stored in the vf-OS Data Storage |
| DEUS051  | Log in to DE Frontend UI | Description | Who: vApps Developer  
What: log in the Developer Engagement Hub's Frontend UI  
Why: to access the Hub services  
Acceptance Criteria | Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment "Login User") |
| DEUS052  | Display Project | Description | Who: vApps Developer / Systems Engineer  
What: will retrieve and display the hierarchy of projects for a common context  
Why: have a hierarchical view of the business/project/area. A project may concern any particular context, eg a company, a department, a product, part of the product, a communication campaign, an engagement strategy, etc. Projects may contain other projects, and may contain any other resources as well, such as Issue trackers, Collaborating mechanisms, Code repositories etc  
Acceptance Criteria | Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose |
| DEUS053 Manage Project | **Description**  
Who: vApps Developer / Systems Engineer  
What: change the hierarchy of projects for a common context  
Why: change the hierarchical view of the business/project/area, and to configure projects such as defining their permissions, structure, scope, objectives etc  
**Acceptance Criteria**  
Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose |
| --- | --- |
| DEUS101 Log in to DE Frontend UI | **Description**  
Who: vApps Developer  
What: log in the Developer Engagement Hub's Frontend UI  
Why: to access the Hub services  
**Acceptance Criteria**  
Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment "Login User") |
| DEUS102 Create Project | **Description**  
Who: vApps Developer  
What: assign a name to the current development project  
Why: create a named development workspace  
**Acceptance Criteria**  
DE creates a blank workspace for the new project in the vf-OS Platform (vf-P)  
A file with the Project configuration will be stored in the platform connected with the user session |
| DEUS103 Configure Project | **Description**  
Who: vApps Developer  
What: defines project parameters and related services eg the tools used for source code version control, collaboration tools, issue Trackers  
Why: to configure the project  
**Acceptance Criteria**  
Project behaviour and Project definition file are updated to use the selected parameters |
| DEUS151 Log in to DE Frontend UI | **Description**  
Who: vApps Developer  
What: log in the Developer Engagement Hub's Frontend UI  
Why: to access the Hub services  
**Acceptance Criteria**  
Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment "Login User") |
| DEUS152 Display Issue Tracker | **Description**  
Who: Systems Engineer  
What: will retrieve and display the Issue tracking environment for a context  
Why: see the issues regarding a particular context. Issues may relate to actions to be performed, bugs to be corrected, best practices and suggestions, quality improvements, etc  
**Acceptance Criteria**  
Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose |
| DEUS153 Manage Issue Tracker | **Description**  
Who: Systems Engineer  
What: will update the Issue tracking environment for a context  
Why: to provide information such as related tickets, milestones, labels and other types of configuration, and to be able to manage the Issue Tracking system |
<table>
<thead>
<tr>
<th>DEUS201</th>
<th>Log in to DE Frontend UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> log in the Developer Engagement Hub's Frontend UI</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to access the Hub services</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS202</th>
<th>Select DE Project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> browse the Developer Engagement Hub's list of projects and select one</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to make the following actions in the scope of a specific DE project</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS203</th>
<th>Create Ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> Use the Developer Engagement Hub's UI to create a ticket</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> create a new ticket in a specific DE project</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS251</th>
<th>Log in to DE Frontend UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> log in the Developer Engagement Hub's Frontend UI</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to access the Hub services</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS252</th>
<th>Display Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Systems Engineer / vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will retrieve and display the Code Repository for a context</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> be able to browse the code, navigate through it, analyse different versions and development branches, see the relationship between code and tickets, between code and communications, etc</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS253</th>
<th>Manage Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Systems Engineer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will manage the Code repository environment for a context</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> create new code modules, create new branches of code, associate code modules or versions with tickets, tutorials or other resources, etc</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS301</th>
<th>Log in to DE Frontend UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vApps Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> log in the Developer Engagement Hub's Frontend UI</td>
<td></td>
</tr>
<tr>
<td>DEUS302</td>
<td>Select DE Project</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>browse the Developer Engagement Hub's list of projects and select one</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to make the following actions in the scope of a specific DE project</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS303</th>
<th>Stage Add Code files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>Use the Developer Engagement Hub's UI to add Source Code files to Version control</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>add source code to a specific DE project</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker accesses the Developer Engagement Hub Frontend UI for the selected purpose</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS304</th>
<th>Commit Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>Use the Developer Engagement Hub's UI to commit Source Code files to Version control</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>add source code to a specific DE project. The commit action should require justification for the commit, and if it includes the insertion of an issue ID, then the issue will be resolved with the unique ID for the commit action</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker accesses the Developer Engagement Hub Frontend UI for the selected purpose</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS351</th>
<th>Log in to DE Frontend UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>log in the Developer Engagement Hub's Frontend UI</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to access the Hub services</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS352</th>
<th>Display Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>will display a Communication area for a purpose</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>see information about a particular context. This may include videos, images and other media types, text and formatted text, and may include combined media types</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS353</th>
<th>Manage Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>will manage the resources regarding a particular communication</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>interrelate and connect resources, establish a tutorial scenario, create a sequence of communicating resources, create a set of actions to be performed to transmit information</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose</td>
<td></td>
</tr>
<tr>
<td>DEUS401</td>
<td>Log in to DE Frontend UI</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>log in the Developer Engagement Hub's Frontend UI</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to access the Hub services</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS402</th>
<th>Select DE Project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>browse the Developer Engagement Hub's list of projects and select one</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to make the following actions in the scope of a specific DE project</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS403</th>
<th>Upload Tutorial</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the selected purpose</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>Use the Developer Engagement Hub's UI to add videos, tutorials and other documentation that supports a stored module or vApp</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>add supporting media to a specific DE project</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS451</th>
<th>Log in to DE Frontend UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Invoker accesses the Developer Engagement Hub Frontend UI or needs to fill a login page before (Frontend Environment &quot;Login User&quot;)</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>log in the Developer Engagement Hub's Frontend UI</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to access the Hub services</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS452</th>
<th>Display Collaboration Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>Systems Engineer/vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>will retrieve and display the available collaboration mechanisms for a particular context</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>allow interaction of the community or of the community and developers, or with experts, or other stakeholders</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS453</th>
<th>Manage Collaboration Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>Systems Engineer/vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>will manage the collaborative environment for a context</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>allow the integration of multiple collaboration tools such as blogs, wikis, forums, chat, allowing separation of media platforms etc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEUS501</th>
<th>Log in to DE Frontend UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Invoker accesses the Developers Engagement Hub Frontend UI for the determined purpose</td>
</tr>
<tr>
<td><strong>Who:</strong></td>
<td>vApps Developer</td>
</tr>
<tr>
<td><strong>What:</strong></td>
<td>log in the Developer Engagement Hub's Frontend UI</td>
</tr>
<tr>
<td><strong>Why:</strong></td>
<td>to access the Hub services</td>
</tr>
</tbody>
</table>
4.2.1.2 UI mockups and Sequence Diagrams

The following sub-sections describe the UI mock-ups and sequence diagrams describing the interactions between the vf-OAK Studio, the Developer, and the SDK.

4.2.1.2.1 Authorisation Scenarios
The Developer Engagement Hub is a platform that will always require authentication and authorisation for all its actions, hence one initial step that is performed at every action is the check if the developer has logged in or has valid credentials, and if not, pop-up the login page. The interaction that is depicted in Figure 106 is valid for all scenarios DEUS001, DEUS005, DEUS051, DEUS101, DEUS151, DEUS201, DEUS251, DEUS301, DEUS351, DEUS401, DEUS451, and DEUS501.

![User authorisation sequence diagram](image)

Figure 106: User authorisation sequence diagram

4.2.1.2.2 Configure Developers Engagement Hub
In order to be configured, the Engagement Hub will request the SDK for the set of configurations available on the Data Storage. Upon receiving this list, the Developer will
select one configuration and retrieve that information by name, as shown in Figure 35 (User Stories DEUS002 and DEUS003).

The retrieved information will then be used to configure the Hub’s look & feel and other environment customisations. These can also be changed in the Hub’s front-end itself, and a similar procedure can be found for the update and storage of the configurations, as shown in Figure 108 (User Stories DEUS006 and DEUS007).

4.2.1.2.3 Invoke Plugin Functionalities
The Developer Engagement Hub will include the possibility of working with different tools for performing the tasks of source code hosting, issue tracking, and collaboration, etc. These are plug-in modules that can be invoked by the Developer Engagement Hub. These plugins also need to be connected to the Developer Engagement Hub and requested to be rendered by the Frontend module, as shown in Figure 109. This scenario is one that can be seen in stories DEUS152, DEUS202, DEUS252, DEUS303, DEUS352, DEUS402, DEUS452, and DEUS502.

Figure 109: Mock-up of the Developers Engagement Hub Portal
4.2.1.3 Interaction description

The following diagram (Figure 111) was taken from the global architecture definition presented in D2.1, and the subsequent text focuses on the interactions and data exchange between the Developers Engagement Hub and other vf-OS components.
In order to clarify the interactions between components, the main interactions of the Developers Engagement Hub component with other components are:

- Developers Engagement Hub (DEH) Controller: This is the central access point to all functionality of the DEH:
  - It receives all requests for functionalities of the DEH and redirects the requests to the corresponding modules
  - It interfaces with the SDK to retrieve metrics about the development of the modules
  - It includes the definition of projects and multiple levels of sub-projects, allowing all the other DEH modules in each level
  - It includes the definition of authentication, authorisations and visibility of assets in the DEH

- Documentation: Handles all the functionalities related to providing information about a project or each level of sub-project or functionalities, including:
  - Wikis
  - Upload of documents and images
  - Movies and other tutorials media

Figure 111: Developers Engagement Hub Component Interactions
• Javadoc documentation and Markdown files

• Collaboration Mechanisms: Takes care of the functionalities related to interaction and collaboration between the community of developers, including:
  • Forums
  • Communication mechanisms like chat, mailing-lists
  • Blogs

• Code Hosting: Is responsible for the storage and version control of the modules of code, and other related (can include versioned documentation as well), including multiple repositories of Git and/or SVN

• Issue Tracker: Allows the raising of issues and management of the project features, permitting multiple actions like:
  • Testing and raising of defects
  • Elicitation of new features and requirements
  • Ticket-based Management
  • Assignment of tasks and issues to multiple developers

• Continuous Integration: Allows the development of new versions and modules, analysing if the new changes break the development of the module.
  • Permitting the development of automatic unit, system and regression tests
  • Permitting the definition of interfaces that can be checked
  • Permits running a set of automated tests upon the commit of a version in a Git repository
  • Permits a set of reports pertaining if the new compilation resulted successfully, if the tests were successful and in the case of failure, alert the corresponding entities
5  Application Services and Middleware (Runtime) Components

5.1  Middleware

5.1.1  Process Enabler Execution

The Process Enabler is the component that deals with the design, enactment, and execution of processes. These functionalities divide into design and runtime and it has been split into two components according to these two different functionalities: the Process Designer and the Process Runtime. As such, the Process Execution component will execute a firmly defined workflow.

5.1.1.1  Behaviour and Functionality

The Process Execution component is composed of the following modules:

- **Process Execution Manager**: This module is the endpoint where the Process Execution receives the request to start a process execution from the vApps and/or vf-OS Assets. If a new request is received, this module will first call the Process Instance Controller module to start a new process instance. After a new process instance is running, this module will start the execution by calling the BPMN Parser module.

- **Process Instance Controller**: This module is responsible for starting and ending process instances as they are needed or not. If a process instance is closed, this module will store the process log of that instance in the Data Storage.

- **BPMN Parser**: This module will first load the BPMN model from the vApp/vf-OS Asset, and will then parse it to gather all information about the needed activities and/or resources. The parsed information is then used to reserve all needed resources (e.g., external services), which are managed by the Resource Manager.

- **BPMS**: This module is responsible for orchestrating the execution of the process model. For that, it can communicate with 3rd party services or cloud resources. Furthermore, it is able to send pause requests of the execution if needed, e.g., if a service is not available. Depending on the currently executed process step, the BPMS module can also interact amongst other things with the Data Analytics component to send process data to further analyse.

- **Service Manager**: The Service Manager module is responsible for the organisation of services that are needed for the execution of the process, returning back to the BPMS information such as service execution URI, parameters or availability. This module is used to reserve all needed services at the beginning of the execution. The instantiated service is called from the BPMS module, each time a new service has to be used.
Figure 112: Process Execution Story Map

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PEUS001</strong> Connect to the Process Manager</td>
<td><strong>Description</strong>&lt;br&gt;Who: vf-OS Process Execution&lt;br&gt;What: Connect to the Process Manager&lt;br&gt;Why: To have access for executing the process&lt;br&gt;&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;The Manager acknowledges the Process</td>
</tr>
<tr>
<td><strong>PEUS002</strong> New Process Instance</td>
<td><strong>Description</strong>&lt;br&gt;Who: vf-OS Process Execution&lt;br&gt;What: Create a new Process Instance&lt;br&gt;Why: So that the process can be executed&lt;br&gt;&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;The Manager passes the deployed process to the BPMN Parser</td>
</tr>
<tr>
<td><strong>PEUS003</strong> Request Services</td>
<td><strong>Description</strong>&lt;br&gt;Who: vf-OS Process Execution&lt;br&gt;What: Connect to the Service Manager&lt;br&gt;Why: So that additional services can be requested for executing a process&lt;br&gt;&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;The Service Manager acknowledges the request</td>
</tr>
<tr>
<td><strong>PEUS004</strong> Launch execution</td>
<td><strong>Description</strong>&lt;br&gt;Who: vf-OS Process Execution&lt;br&gt;What: Contact the BPMS module&lt;br&gt;Why: So that a process can be executed by the BPM Executing module&lt;br&gt;&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;The invoke call is successfully processed by the BPMS</td>
</tr>
<tr>
<td><strong>PEUS005</strong> Call Platform</td>
<td><strong>Description</strong>&lt;br&gt;Who: vf-OS Process Execution&lt;br&gt;What: Call the vf-P to execute a process&lt;br&gt;Why: So that a process is effectively executed&lt;br&gt;&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;The invoke call is successfully relayed to the vf-P</td>
</tr>
<tr>
<td>PEUS006</td>
<td>Receive data from Platform</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
<td>Who: vf-OS Process Execution</td>
</tr>
<tr>
<td></td>
<td>What: Data processed is sent back</td>
</tr>
<tr>
<td></td>
<td>Why: So that a process can relay the data received to the calling vApp/vf-OS Asset</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The processed data is received by the Process Instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PEUS007</th>
<th>Call External Service Provision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Who: vf-OS Process Execution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Call the external service provision</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that a process is effectively helped by the execution of 3rd party services</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The invoke call is successfully relayed to the ESP</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PEUS008</th>
<th>Receive data from ESP</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Who: vf-OS Process Execution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Data processed is sent back</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that a process can relay the data received to the calling vApp/vf-OS Asset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The processed data is received by the Process Instance</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PEUS009</th>
<th>Send usage data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Who: vf-OS Process Execution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Send usage data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that the Platform can inform the System Dashboard</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The usage data is successfully relayed to the Platform</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PEUS010</th>
<th>Save logs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Who: vf-OS Process Execution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Save execution logs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: So that the Platform can have 1st-hand information about errors in usage of Platform resources</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The logs are successfully relayed to the Platform</td>
<td></td>
</tr>
</tbody>
</table>

5.1.1.2 UI mockups and Sequence Diagrams

The following sub-sections describe the UI mockups and sequence diagrams describing the interaction between the sections of the Process Execution.

5.1.1.2.1 Deploy Process

This feature provides the capability to deploy a process within the BPMN Parser.

The main steps/functionalities are as follows:

- Connect to the Process Manager
- New Process Instance
5.1.1.2.2 Execute Process

This feature provides the capability to execute a process in the Platform.

The main steps/functionalities are as follows:

- Request Services
- Launch Execution
- Call Platform
- Receive data from Platform

5.1.1.2.3 3rd Party Execution

This feature provides the capability to execute a 3rd party service offered by the External Service Provision.

The main steps/functionalities are as follows:

- Call External Service Provision
• Receive data from ESP

5.1.1.2.4 Inform the Platform

This feature provides the capability to inform the Platform and thus the System Dashboard about executing performance KPIs.

The main steps/functionalities are as follows:

• Send usage data
• Save logs
The UI of the Process Execution is as follows:

![Process Execution UI](image)

Figure 117: Process Execution UI

5.1.1.3 Interaction description

From the previous description of the functionality covered by the Process Execution component, a deeper level of detail regarding the main modules of the component and the interaction between those modules and other vf-OS components emerges. Whilst next Figure 118 shows the Architecture diagram, as presented in D2.1, the accompanying text focuses on the interactions and data exchange between the Process Execution and other vf-OS components.
In order to clarify the interactions between components, the information exchanged between Process Execution subcomponents and other components has been detailed (see). The main interactions of Process Designer component’s classes with other components are:

- **Process Instance Controller:** This component deals with the Usage Data API, sending logs to the Platform component in relation to the deployed process. The main information flows are:
  - It sends execution logs to the Usage Data API (Interaction with the Platform component)
  - It receives the execution logs from the Process Instance (Interaction with the External Service Provision and the Platform component)
  - It receives the deployed process from the Process Execution Manager (Interaction with the vf-OS Assets component)
• Process Execution Manager: This component deals with the actual execution of the process, interacting with the Process Instance Controller, the Process Instance, and the Exec Process API. The main information flows are:
  • It receives from the deployed process from the Exec Process API (Interaction with the vf-OS Assets component)
  • It sends the deployed process to the Process Instance Controller (Interaction with the Platform component)
  • It sends the deployed process to the BPMN Parser (Interaction with the External Service Provision and the Platform component)

• BPMN Parser: This component deals with parsing the deployed process model into a format that can be read by the Platform API. The main information flows are:
  • It receives the deployed process from the Process Execution Manager (Interaction with the vf-OS Assets component)
  • It sends the parsed process to the BPMNS (Interaction with the External Service Provision and the Platform component)

• Service Manager: This component deals with the services; ensuring that the services are available for use when needed, and preventing services from overlapping. The main information flow is:
  • It sends the service info to the BPMNS (Interaction with the External Service Provision and the Platform component)

• BPMNS: this component connects the Process Instance to the Platform and 3rd Party services. It deals with passing the service information and parsed process models. The main information flows are:
  • It receives the parsed model from the BPMN Parser (Interaction with the vf-OS Assets component)
  • It receives the service info from the Service Manager (Interaction with vf-OS Assets component)
  • It passes and receives service data to the 3rd Party Execution API (Interaction with the External Service Provision component)
  • It passes and receives process data to the Platform API (Interaction with the Platform component)
5.1.2 Messaging

The Messaging component provides functionalities to enable message flow between vf-OS components. This component provides an interface that enables loosely coupled messages to be sent by one component to another and acts as a mediator for controlling message flow based on the meta-data contained in the message and status of the receiver. The messages consist of requests, reports or events that are dispatched by one
component and consumed by another component(s). These messages contain information that is needed to coordinate systems and track the progress of the overall business process. The controlling actions involve the creation of communication channels, routing, queue management etc.

### 5.1.2.1 Behaviour and Functionality

The Messaging component provides a set of functionalities that can be grouped as follows:

- **Component configuration:** This feature is used for providing, viewing, and updating the configuration and rules that will guide the functioning of the messaging components. The configurations parameters can be Maximum message size, Backup Frequency, Authentication type, Maximum retries, supported protocols, etc. Whilst the rules can be criteria for Message exclusion, Message filtering, and message broadcasting etc. These parameters are to be provided as key value pairs by the vf-OS IT admin who maintains the messaging component during initial set up. The provided details through this feature are useful for other functionalities.

- **Message Handling:** This feature is the core of the messaging component and provides necessary implementation for actual message flow. Some important functionalities encapsulated by this feature are creating communication channels, finding message destination(s) either from the message meta-data or pre-defined rules, maintaining queues for efficient message transfer, guaranteeing that messages are correctly delivered etc. The functionalities in this feature will utilise the functional implementations provided in both of the other features.

- **Performance Monitoring:** This feature is dedicated for monitoring the performance and tracing the errors that can arise during message transfers. Some of the functionalities are tracing message delivery failures, time spent in message deliveries, availability status of the messaging component itself. Additionally, the functional implementation of this feature will monitor the error logs to generate events when necessary. For instance, one such event can be threat which can be traced by invalid data being sent by same component multiple times.

Follows is a story map where the main features, epics and user stories for the messaging components have been identified (see Figure 120).
Figure 120 Messaging Story Map

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MSUS001</strong> List existing configurations</td>
<td><strong>Description</strong>&lt;br&gt;Who: vf-OS IT manager&lt;br&gt;What: Display all the configurations that are applicable for messaging component&lt;br&gt;Why: So that admin user can view and understand the configuration details&lt;br&gt;&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;Make sure that all the configuration parameters and their values are provided</td>
</tr>
<tr>
<td><strong>MSUS002</strong> Add/delete/update configuration(s)</td>
<td><strong>Description</strong>&lt;br&gt;Who: vf-OS IT manager&lt;br&gt;What: Allow modification of the configuration parameters&lt;br&gt;Why: So that configuration parameters can be adjusted when necessary&lt;br&gt;&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;Make sure that admin users can perform CRUD operations for configuration parameters</td>
</tr>
<tr>
<td><strong>MSUS003</strong> List/add/update/delete BusinessRules</td>
<td><strong>Description</strong>&lt;br&gt;Who: vf-OS IT manager&lt;br&gt;What: Allow creation and modification of business rules&lt;br&gt;Why: So that applicable business rules for messaging such as the criteria for message exclusion, message aggregation etc can be defined&lt;br&gt;&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;Make sure that admin users can perform CRUD operations for business rules</td>
</tr>
<tr>
<td><strong>MSUS004</strong></td>
<td><strong>Description</strong></td>
</tr>
</tbody>
</table>
| Add advanced configurations | Who: vf-OS IT manager  
What: Provide functionality for providing additional advanced functionalities  
Why: So that the messaging component can be configured for additional configurations such as message encryption/decryption, removal of duplicates and/or invalid streams, check for tokens, checksums etc.  
Acceptance Criteria |  
| MSUS005. Determine message destination(s) | Description  
Who: vf-OS Messaging  
What: Determines the destination for the message that has been received by the messaging component.  
Why: So that the messaging component can send the message to the correct destination. The destination is computed from the metadata contained in the message packet.  
Acceptance Criteria |  
| MSUS006. Route messages to one or more of many destinations | Description  
Who: vf-OS Messaging  
What: Create communication channels and send message to the destinations as calculated from MSUS005.  
Why: So that the messages will be sent to their corresponding destinations.  
Acceptance Criteria |  
| MSUS007. Create communication channel based on the requirement of receiver | Description  
Who: vf-OS Messaging  
What: Create communication channels to support multiple protocols that the receiver might support rather than the standard HTTP.  
Why: So that the messages can be sent to different components that might be following different protocols.  
Acceptance Criteria |  
| MSUS008. Perform message aggregation/decomposition | Description  
Who: vf-OS Messaging  
What: Perform message aggregation, decomposing messages from/into multiple messages and sending them to their destination, then recomposing the responses into one message to return to the user.  
Why: To enable message sender to send messages to multiple receivers in one message packet.  
Acceptance Criteria |  
| MSUS009. Perform data filtering and quality check | Description  
Who: vf-OS Messaging  
What: Perform message processing such as encryption/decryption, removal of duplicates and/or invalid streams, check for tokens, checksums.  
Why: To enable data protection and filtering of unwanted data in the messages based on the business rules and configurations defined.  
Acceptance Criteria |  
<p>| | Removal of possible faulty messages and the data flow is provided with end to end encryption. This is dependent on the use cases MSUS003 and MSUS004 so requires that those functionalities are implemented. |</p>
<table>
<thead>
<tr>
<th>MSUS010. Queueing and precedence management of messages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: vf-OS Messaging</td>
<td></td>
</tr>
<tr>
<td>What: Provide functionality to store the messages in a queue before being sent to the receiver. Additionally precedence management can help to define and determine more urgent or important messages.</td>
<td></td>
</tr>
<tr>
<td>Why: To prevent message loss caused by the receiving component failures. Additionally precedence management can guarantee adequate response time for critical components.</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

Make sure that the messages are not lost and critical components can have response time within the threshold limits.

<table>
<thead>
<tr>
<th>MSUS011. RESTful API to messaging using HTTP</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: vf-OS Assets</td>
<td></td>
</tr>
<tr>
<td>What: Provide interface for sending messages using HTTP.</td>
<td></td>
</tr>
<tr>
<td>Why: To enable APPs to use messaging component to send messages by using HTTP through standardised and understandable interface.</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

vApps can send and receive message through the exposed RESTful interface

<table>
<thead>
<tr>
<th>MSUS012. Messaging of continuous real-time message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: vf-OS Components/ vf-OS Assets</td>
<td></td>
</tr>
<tr>
<td>What: Provide functionality to enable sending continuous messages by following the constraint for real-time message exchanges.</td>
<td></td>
</tr>
<tr>
<td>Why: To enable streaming of real time data for the vf-OS components/vf-OS Assets that they can directly display in their client applications - eg in web browsers. An example of such cases is streaming financial stock prices, live auctions or to dynamically update live content and news etc.</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

Make sure that the messages are exchanged with minimum latency within the time constraint as required by receiving component/asset

<table>
<thead>
<tr>
<th>MSUS013. Transactional messaging</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: vf-OS Components/ vf-OS Assets</td>
<td></td>
</tr>
<tr>
<td>What: Provide functionality to send messages by following the principles of transactions.</td>
<td></td>
</tr>
<tr>
<td>Why: To provide messaging functionality so that several related messages can be coupled into a single transaction, ensuring that the messages are delivered in order, delivered only once, and are successfully retrieved from their destination queue. If any errors occur, the entire transaction is cancelled.</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

Make sure that the overall messages in the transactions are committed or rolled back

<table>
<thead>
<tr>
<th>MSUS014. Historical storage of messages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: vf-OS Messaging</td>
<td></td>
</tr>
<tr>
<td>What: Provide functionality to persist the messages for traceability purpose.</td>
<td></td>
</tr>
<tr>
<td>Why: To persist messages in external repository for traceability purpose.</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

Prove the traceability of the messages in case of disputes with the information that includes origin of message, destination(s) and timestamp.

<table>
<thead>
<tr>
<th>MSUS015. Encryption and Decryption of the messages being persisted in external repository</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: vf-OS Messaging</td>
<td></td>
</tr>
<tr>
<td>What: Provide functionality to secure the messages that will be persisted for traceability purpose.</td>
<td></td>
</tr>
<tr>
<td>Why: The backup in the external repository must be made secure with necessary functionality for encryption and decryption so that external entities will not be able to access and manipulate the messages that have been stored in history.</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

Prove the traceability of the messages in case of disputes with the information that includes origin of message, destination(s) and timestamp.
Acceptance Criteria
Messages can be stored and retrieved by using suitable encryption and decryption mechanism. External entities can’t retrieve and manipulate the messages that have been stored in the external repository. MSUS013 must be implemented before this use case.

**Description**

**Who:** vf-OS Messaging  
**What:** Track and log the performances of the messaging component.  
**Why:** For monitoring the working of the component and provide traceability of the errors for troubleshooting purposes. Additionally, this functionality will also provide mechanisms for detection of errors and generate events that will be useful for detecting abnormalities.

**Acceptance Criteria**
All the message flows are logged and errors are correctly reported

**Description**

**Who:** vf-OS Messaging  
**What:** Provide notifications of critical events or errors.  
**Why:** To detect errors and events. The types of errors or events will determine if it needs to be sent to the message sender or the system dashboard. One typical example of an error message is when a message send to the destination fails permanently. While the associated event can be similar to "message sender is sending invalid data packet at frequent intervals".

**Acceptance Criteria**
Common data failure situations are detected and notified

**Description**

**Who:** vf-OS System Dashboard  
**What:** Provide periodic performance report to the system dashboard  
**Why:** This will enable the system admin to monitor the performance and health of the messaging component and make plans for optimisation and or scaling the component in case of lower performance

**Acceptance Criteria**
Performance reports are sent to the system dashboard on periodic basis as defined by the system dashboard admin.

### 5.1.2.2 UI mockups and Sequence Diagrams

The following sub-sections describe the UI mockups and sequence diagrams describing the interaction.

#### 5.1.2.2.1 Manage Configurations and Business Rules

This feature provides the capability to manage configurations for the messaging component.

The main steps/functionality are:

- List existing configurations and rules associated with the selected instance of vf-OS Messaging component
- Perform CRUD operations on configurations and rules
Figure 121 Sequence Diagram for Managing Configurations and Rules of Messaging Component
The associated UIs for managing configurations and rules are as follows:

![Figure 122 List, View Details, Edit and Delete Configurations UI Mockup](image-url)
5.1.2.2.2 Message Handling
This feature provides the capability to send messages from one component to be received by another component. The main steps/functionality are:

- Send a message from the source component. This step will have the core content of the message along with associated meta-data. The meta-data will include destination, messaging type (e.g., Point-to-point, broadcast, publish-subscribe etc.), conditions (like real-time, transactional etc.)
- The messaging routing and pre-processing criteria are based on the defined configurations and rules which thus need to be retrieved. This can lead to errors for messages that are not within the defined compliance
- Based on the rules and message metadata pre-processing of data is performed
- Messages are pushed to the queue or immediately sent for dispatch based on the messaging type, data size and the availability of the receiver
- Messages in the queue are retrieved, the destination is read from the message metadata, route to the destination is computed and the communication channel is established and sent to the corresponding receiver. If the receiver is not available the message is pushed back in the queue and tried again until the max tries count for the message is valid
- Possible error messages during the operations are asynchronously sent back to the sender
This feature will not provide any user interface so it does not have associated UI mockups.

5.1.2.2.3 **Performance Monitoring**

This feature provides the capability for monitoring the performance and tracing of errors during the run-time execution of the messaging component. The main steps/functionality are:

- Collect the errors that will occur between the processes of sending messages between components
- The error logs are monitored along with the message queue depths to proactively find potential problems and publish notifications via dashboard for system admin
- Keep track of the performance metrics such as messaging component down time, response time, queue depths, message size etc to find correlations between performance and messaging service. The performance matrices are provided for
system admin via system dashboard for performance optimisation and scalability.

Figure 125 Sequence Diagram for Performance Monitoring of Messaging Component

This feature will not provide any user interface so it does not have associated UI mockups.

5.1.2.3 Interaction Description

Based on the description of the functionality covered by the messaging component we can observe a number of interactions that the component will have with other vf-OS components. In this section is presented a detailed representation of interactions with other vf-OS components and also some internal interactions between sub-components of Messaging component. The following figure shows the flow of information between the messaging subcomponents and other vf-OS components.
In order to clarify the interactions between components the main interactions of messaging component with other components are as explained below:

- **Config Storage Access:** This provides necessary interaction with the vf-OS storage component and is used for storing configuration details of the messaging component which includes parameters as discussed in the behaviour and functionality section. The main information flows are:
  - Put/Get configuration details into/from the storage. The generic representation of data model used for this interaction is provided by using a list of key value pairs.
- Request Handler: This provides necessary interactions for actual message flow. This functionality is utilised by any vf-OS component wanting to send messages by utilising broadcasting and point-to-point messaging pattern. Additionally, this is also utilised by the publish/subscribe component to make message transfers using publish-subscribe pattern. The main information flows exchanged with external components are:
  - vf-OS components can provide messages to be dispatched to other components
  - Messaging component can provide messages to be dispatched to other components
  - Publish/Subscribe component can provide messages to be published to the subscribers

- Histories: This provides necessary interaction with the storage component to store messages that flow through the messaging component. These historical records can be retrieved for traceability purpose when necessary. The main information flows exchanged with historical storage are:
  - Messaging component can put messages into the storage with a timestamp attached
  - Messaging component can get traces of messages from the storages with defined parameters like message source, message destination, timestamp etc

- Monitoring: This provides necessary interaction with the system dashboard component to publish performance, errors, and events during the messaging process. The main information flows exchanged with system dashboard are:
  - Messaging component can publish errors and event logs with optional criticality tag
  - Messaging component can publish periodic performance metrics collected during the execution of the messaging component

Figure 127 shows the classes with external interactions and associated data model. Most of the entities in the data model are explicit by name and attribute. The message head contains the information necessary to route messages (topic identifiers in the case of publish/subscribe). Message headers are coded so that the message handler can receive all the necessary information needed to route and prioritise the message. The Message body includes the actual message that needs to be transferred. The message meta-data provides additional information that will be useful for pre-processing, context generation etc. The Message meta-model can be aided by the use of a precise data dictionary that documents metadata.
5.1.3 Publish/Subscribe

The Publish/Subscribe component provides the implementation of publish/subscribe messaging pattern which is used to communicate messages between different system components without these components knowing anything about each other’s identity. This is one of the messaging patterns specifically used in even driven architecture and is thus an extension of the vf-OS messaging component with extended functionalities. The extended functionality is mainly across the topic based and content based messaging. In a topic-based system, messages are published to “topics” or named logical channels and subscribers in a topic system will receive all messages published to that topic. In this scenario, all the subscribers will receive the same messages. The publisher is responsible for defining the topics of messages to which subscribers can subscribe. While, in a content-based system, messages are only delivered to a subscriber if the attributes or content of those messages match constraints defined by the subscriber. The subscriber is responsible for classifying the message or providing the constraint for defining the part of the message that they are interested. Note that for actual message flow this component will utilise the functionality provided by vf-OS messaging component to establish communication channels with the receiver.

5.1.3.1 Behaviour and Functionality

Publish/Subscribe component provides a set of functionalities that can be grouped as follows:
- **Component configuration:** Since this component is an extension of messaging component this is provided by the messaging component.
- **Publication/Subscription Handling:** This feature is the core of the publish/subscribe component and provides an implementation of the publish-subscribe messaging pattern. Some important functionality encapsulated by this subcomponent are that it maintains lists of publishers and subscribers, maintains topics to be published by the publisher, maintains subscription content details for subscribers etc. Additionally, this feature also encapsulates inspection the topic-related information or the content information that is included in each published message. This component publishes the message to the communication infrastructure by creating a communication channel by using the feature provided by the messaging component.

- **Performance Monitoring:** This feature is dedicated to monitoring the performance and tracing the errors that can arise during run-time execution of the publish/subscribe component. This functionality is also provided by the messaging component and hence is not discussed here.

Following, there is a story map where the main features, epics and user stories for the Publish/Subscribe components have been identified (see Figure 128).
The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSUS004.</td>
<td><strong>Register publisher</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Who:</strong> vf-OS Component</td>
</tr>
<tr>
<td></td>
<td><strong>What:</strong> Allows registration of publisher with necessary details.</td>
</tr>
<tr>
<td></td>
<td><strong>Why:</strong> So that the vf-OS component that will produce messages can register to the publish/subscribe component and can start publishing messages. The publishing component can also define the various topics to which the messages will be published.</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td></td>
<td>vf-OS components are allowed to register themselves as message publisher through provided service interface and allowed to define message topics and/or message topic hierarchies. This is also dependent on PSUS011 so that the registered message producers can publish its messages.</td>
</tr>
<tr>
<td>PSUS005.</td>
<td><strong>Unregister publisher</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Who:</strong> vf-OS Component</td>
</tr>
<tr>
<td></td>
<td><strong>What:</strong> Allow deregistration of the message producer.</td>
</tr>
<tr>
<td></td>
<td><strong>Why:</strong> So that the message producers can be removed from the list of publishers.</td>
</tr>
</tbody>
</table>

Figure 128 Publish/Subscribe Story Map
### Acceptance Criteria

The component that has been deregistered cannot send messages through this component and all the corresponding topics are removed. All the subscribers that have subscribed to that data source and/or topics will be notified of deregistration through PSUS006.

<table>
<thead>
<tr>
<th>PSUS006. Notify un-registration of publishers to subscribers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: vf-OS Publish/Subscribe</td>
</tr>
<tr>
<td>What: Provide notification of publisher being un-registered.</td>
</tr>
<tr>
<td>Why: So that the subscribers will be aware of the messages that will no longer be available to which they had previously subscribed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PSUS007. Discover available messages for subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: vf-OS Component</td>
</tr>
<tr>
<td>What: Discovery of various messages that can be collected from publish/subscribe component.</td>
</tr>
<tr>
<td>Why: So that the subscriber can discover different types of message and details of the message type and topics are provided by publish/subscribe component.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PSUS008. Subscribe to messages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: vf-OS Component</td>
</tr>
<tr>
<td>What: Subscribe to data producers.</td>
</tr>
<tr>
<td>Why: To enable interested entities to subscribe to messages which can be entire message, messages by topics provided by publisher or to part of the message content by specifying the criteria</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PSUS009. Unsubscribe to messages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: vf-OS Component</td>
</tr>
<tr>
<td>What: Allow un-subscription of messages that has been previously subscribed</td>
</tr>
<tr>
<td>Why: So that the message subscribers choose to not to receive certain messages that they have previously subscribed. In this case the component will be removed from the list of subscribers. This use-case is also applicable when the publisher invokes PSUS005 when the subscriber is automatically unsubscribed and notified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PSUS010. Notify availability of new messages available for subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: vf-OS Publish/Subscribe</td>
</tr>
<tr>
<td>What: Provide notifications to registered subscribers regarding the new types of messages available for subscription. This use-case is triggered when new message publisher is registered or publisher creates new message.</td>
</tr>
<tr>
<td>Why: To provide information to subscribers about new messages available through the publish/subscribe component that might be interesting for them. Additionally the notification can be generated based on the classification of the messages or topics that the subscribers have previously subscribed. The subscriber can choose to get such notifications or not.</td>
</tr>
</tbody>
</table>

### Acceptance Criteria

All the subscribers that had subscribed to the message by topic or content are asynchronously notified about the un-registration action of the publisher.

Make sure that interested subscribers can find the list of data publishers based on different search criteria.

vf-OS components are allowed to subscribe to messages and will receive a corresponding message when published. The acceptance is that this is also dependent on PSUS011.

The unsubscribing component will not receive any messages that they have just unsubscribed to. In case of automatic un-subscription the component are provided with suitable notification.
### PSUS011. Publish messages by message producers

Notifications are pushed to the registered subscribers based on their interest as soon as new publishers or topics are registered.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| **Who:** vf-OS Component  
**What:** Provide interface for publishing messages  
**Why:** To enable components to publish messages which can also be classified by topics. The publishing entity doesn’t need to take care of the delivery of the message to subscribers. |

<table>
<thead>
<tr>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>vf-OS components including vApps can publish message through the provided interface</td>
</tr>
</tbody>
</table>

### PSUS12. Send data to subscribers

Description

Who: vf-OS Publish/Subscribe  
**What:** Push messages to the subscribers based on their interest.  
**Why:** To enable subscribers to receive messages that they have subscribed based on topics or content constraint that they have provided during subscription in PSUS008. This will prevent the message consumer to have to continuously poll for messages and will receive the message as soon as the messages are published.

<table>
<thead>
<tr>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make sure that the messages are published to subscribers with minimum latency within the time constraint as required by subscribing entity.</td>
</tr>
</tbody>
</table>

### PSUS13. Classify messages based on contextual information

Describeation

Who: vf-OS Publish/Subscribe  
**What:** Messages and/or messages content classification based on contextual information of the publisher and/or message metadata.  
**Why:** To provide dynamic messages classification capability to associate received messages into different topics, so that subscribers will benefit from receiving messages that will be more valuable to them.

<table>
<thead>
<tr>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make sure that the classification mechanism works and messages are not lost during the classification process.</td>
</tr>
</tbody>
</table>

## 5.1.3.2 UI mockups and Sequence Diagrams

### 5.1.3.2.1 Register Publishers and Subscribers

This feature provides the capability for publishers and subscribers to register and unregister to the Publish/Subscribe component as message producer and consumers respectively.

Steps for publisher registration and deregistration is provided in the sequence diagram Figure 129 and mainly involves following steps:

- Publisher sends requests for registration with its message definition and list of topics that will be used for providing messages
- The detail from the publisher is stored in the publisher/subscribers list and a unique identifier for the publisher is provided which will be utilised by the publisher for publishing data
- If the publisher wishes to un-register then they can send the request for un-subscription along with their identifier and if successful all the corresponding subscribers are also notified
Steps for subscription and un-subscription from a message consumer is provided in the sequence diagram in Figure 130 and mainly involves following steps:

- Subscriber can send discovery requests to find the different types of messages and/or topics that are provided by the publish/subscribe component
- Subscriber can subscribe to the message which can be entire message, by topics or by content. The interest of the subscriber is registered and is responded with acknowledgement together with subscription identifier
- If the subscriber wishes to unsubscribe then they can send the request for un-subscription along with the subscription identifier
5.1.3.2.2 Publication and Subscription of Messages

This feature provides the capability to publish message by the publisher and to be pushed to all the subscribers that have subscribed to the message. Steps for publishing messaging is provided in the sequence diagram in Figure 131 and mainly involves following steps:

- Publisher sends requests for publishing a message with message packet and publisher identifier
- The data is collected and processed for possible classification of the message into topics. And additionally it may also be stored in the message storage for traceability purpose
- The next step is pushing data to the subscriber which is explained in the following scenario
Steps for sending messages to the subscribers are provided in the sequence diagram Figure 132 and mainly involves following steps:

- Get a list of subscribers that have subscribed to the message that has been published by publisher
- For each subscriber, a message is provided utilising the communication channel that will be provided by the vf-OS Messaging component. During this entire process, the overall performance is traced for errors and other performance metrics.
5.1.3.3 Interaction Description

Based on the description of the functionality covered by the messaging component a number of interactions that the component will have with other vf-OS components can be observed. Presented in this section is a detailed representation of interactions with other vf-OS components and internal interactions between subcomponents of the Messaging component. The following figure shows the flow of information between the publish/subscribe subcomponents and vf-OS components.
To clarify the interactions between components the main interactions of publish/subscribe component with other components are as explained below:

- **Storage Access**: This provides the necessary interaction with the vf-OS storage component to get/put the list of publishers and subscribers that are registered in the publish/subscribe component. The list of subscribers can have list of topics that they have defined for publishing messages. While the list of subscribers can have a list of content that they have subscribed to. The main information flows are:
• Put/Get publishers and subscribers that will use the publish/subscribe component for message publishing and subscription

• Publish/Subscribe Handler: This provides interactions for actual message flow by using publish/subscribe pattern. This functionality is utilised by any vf-OS component/vApps wishing to publish or subscribe to messages. This sub-component utilises the functionality provided by messaging component for creation of messaging channel to publishers or subscribers as needed. The main information flows exchanged with external components are:
  • vf-OS components/vApps can register themselves as publisher and define topics/topics hierarchies
  • vf-OS components/vApps can publish messages
  • Vf-OS components/vApps can subscribe to messages and define the constraint to get part of message that they want to receive
  • Vf-OS components/vApps can subscribe to messages

• Monitoring: This provides interaction with the messaging component to publish performance, errors and events during the message publication process. These performance metrics are provided to the system dashboard via the messaging component for overall evaluation of the vf-OS messaging middleware solution. The main information flows exchanged with messaging component are:
  • Publish errors and event logs with optional criticality tag
  • Publish periodic performance metrics collected during execution time

Figure 134 shows the classes with external interactions and associated data model.

![Diagram](image-url)
5.2 Data Management

5.2.1 Data Storage

The storage component implements a scalable data storage system, capable of handling real-time sensor data and events, as well as other types of non-real-time heterogeneous data.

5.2.1.1 Behaviour and Functionality

The Storage component provides a set of functionalities that could be grouped as follows:

- **Database management**: It manages the databases, allowing to creating and dropping a database. It also allows making general queries on the structures in each database: Tables, views, etc.

- **Data structures management**: It manages the data structures in each database. The data structures are tables, views, indexes, metrics, etc. It allows to creating, deleting and modifying data structures.

- **Entities management**: It manages the entities. The entities are the real data to be stored in databases. It allows to insert, update, delete and query data in the databases.

- **Permissions management**: It manages the permissions. It allows the defining of roles and users as well as granting and revoking different permissions to users and roles.

On the other hand, three distinct types of storage have been defined: Relational, Time-series and NoSQL. Each of them will be developed in a different Release.

Follows is a story map where the main features, epics and user stories for the storage components have been identified (see Figure 200).
The description of each user story is as follows:

### DSUS001 Create SQL Database

**Description**

**Who:** A component/vf-OS asset/vApp  
**What:** Creates a SQL database  
**Why:** To be able to store and retrieve information

**Acceptance Criteria**  
The database is successfully created

### DSUS002 Drop SQL Database

**Description**

**Who:** A component/vf-OS asset/vApp  
**What:** Drops a SQL database  
**Why:** Because it is not needed any more
<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Description</th>
<th>Who:</th>
<th>What:</th>
<th>Why:</th>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSUS101</td>
<td>Create TimeSeries Database</td>
<td>A component/vf-OS asset/vApp</td>
<td>Creates a Time Series database</td>
<td>To be able to store and retrieve information of time series</td>
<td>The database is successfully created</td>
</tr>
<tr>
<td>DSUS102</td>
<td>Drop TimeSeries Database</td>
<td>A component/vf-OS asset/vApp</td>
<td>Drops an existing TimeSeries database</td>
<td>Because it is not needed anymore</td>
<td>The database is successfully dropped</td>
</tr>
<tr>
<td>DSUS201</td>
<td>Create NoSQL Database</td>
<td>A component/vf-OS asset/vApp</td>
<td>Creates a NoSQL database</td>
<td>To be able to store and retrieve very large amounts of information of semi structured data in a horizontally scalable way and with great availability</td>
<td>The database is successfully created</td>
</tr>
<tr>
<td>DSUS202</td>
<td>Drop NoSQL Database</td>
<td>A component/vf-OS asset/vApp</td>
<td>Drops an existing NoSQL database</td>
<td>Because it is not needed anymore</td>
<td>The database is successfully dropped</td>
</tr>
<tr>
<td>DSUS003</td>
<td>Describe SQL Database</td>
<td>A component/vf-OS asset/vApp</td>
<td>Ask for a description of a SQL database</td>
<td>To know its main characteristics</td>
<td>The database characteristics are returned</td>
</tr>
<tr>
<td>DSUS004</td>
<td>Catalog of SQL Database tables</td>
<td>A component/vf-OS asset/vApp</td>
<td>Ask for the list of tables of a database</td>
<td>To know the existing tables</td>
<td>The list of tables of the database is returned</td>
</tr>
<tr>
<td>DSUS005</td>
<td>Catalog of SQL Database views</td>
<td>A component/vf-OS asset/vApp</td>
<td>Ask for the list of views of a database</td>
<td>To know the existing views</td>
<td>The list of views of the database is returned</td>
</tr>
<tr>
<td>DSUS103</td>
<td>List Metric names</td>
<td>A component/vf-OS asset/vApp</td>
<td>Ask for the list of metrics of a database</td>
<td>To know the existing metrics</td>
<td>The list of metrics of the database is returned</td>
</tr>
<tr>
<td>DSUS104</td>
<td></td>
<td>A component/vf-OS asset/vApp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>List tag names</td>
<td>Who: A component/vf-OS asset/vApp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: Queries for a list of tags that meet a criteria</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: To get the list of tags that meet a criteria</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>The list of tags is returned</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSUS105 List tag values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Queries for a list of tags and values that meet a criteria</td>
</tr>
<tr>
<td></td>
<td>Why: To get the list of tags and values that meet a criteria</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>The list of tags and values is returned</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSUS203 Describe NoSQL Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td>What: Ask for a description of a NoSQL database</td>
</tr>
<tr>
<td>Why: To know its main characteristics</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSUS204 Catalog of NoSQL Database tables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td></td>
</tr>
<tr>
<td>What: Ask for the list of tables of a database</td>
<td></td>
</tr>
<tr>
<td>Why: To know the existing tables</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>The list of tables of the database is returned</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSUS005 Catalog of NoSQL Database views</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td></td>
</tr>
<tr>
<td>What: Ask for the list of views of a database</td>
<td></td>
</tr>
<tr>
<td>Why: To know the existing views</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>The list of views of the database is returned</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSUS006 Create SQL Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td></td>
</tr>
<tr>
<td>What: Creates a SQL table in a database</td>
<td></td>
</tr>
<tr>
<td>Why: To be able to store and retrieve information</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>The table is successfully created</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSUS007 Drop SQL Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td></td>
</tr>
<tr>
<td>What: Drops an existing SQL database</td>
<td></td>
</tr>
<tr>
<td>Why: Because it is not needed anymore</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>The database is successfully dropped</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSUS008 Alter SQL Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td></td>
</tr>
<tr>
<td>What: Alters the structure of an existing SQL table in a database</td>
<td></td>
</tr>
<tr>
<td>Why: To add a column, delete it, change the datatype of a column...</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>The table is successfully altered</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSUS106 Delete Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td></td>
</tr>
<tr>
<td>What: Drops an existing metric and the data associated with it</td>
<td></td>
</tr>
<tr>
<td>Why: Because it is not needed anymore</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>The metric is successfully deleted</td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DSUS206</td>
<td>The metric is successfully dropped</td>
</tr>
<tr>
<td>DSUS207</td>
<td>Description</td>
</tr>
<tr>
<td>DSUS208</td>
<td>Description</td>
</tr>
<tr>
<td>DSUS009</td>
<td>Description</td>
</tr>
<tr>
<td>DSUS010</td>
<td>Description</td>
</tr>
<tr>
<td>DSUS011</td>
<td>Description</td>
</tr>
<tr>
<td>DSUS209</td>
<td>Description</td>
</tr>
<tr>
<td>DSUS210</td>
<td>Description</td>
</tr>
<tr>
<td>DSUS211</td>
<td>Description</td>
</tr>
</tbody>
</table>

**Create NoSQL Table**
- **Who:** A component/vf-OS asset/vApp
- **What:** Creates a NoSQL table in a database
- **Why:** To be able to store and retrieve information

**Acceptance Criteria**
The table is successfully created.

**Drop NoSQL Table**
- **Who:** A component/vf-OS asset/vApp
- **What:** Drops an existing NoSQL table in a database
- **Why:** Because it is not needed anymore

**Acceptance Criteria**
The table is successfully dropped.

**Alter NoSQL Table**
- **Who:** A component/vf-OS asset/vApp
- **What:** Alters the structure of an existing NoSQL table in a database
- **Why:** To add a column, delete it, change the datatype of a column...

**Create SQL View**
- **Who:** A component/vf-OS asset/vApp
- **What:** Creates an SQL view in a database
- **Why:** To be able to make simpler queries

**Acceptance Criteria**
The view is successfully created.

**Drop SQL View**
- **Who:** A component/vf-OS asset/vApp
- **What:** Drops an existing SQL view in a database
- **Why:** Because it is not needed anymore

**Acceptance Criteria**
The view is successfully dropped.

**Replace SQL View**
- **Who:** A component/vf-OS asset/vApp
- **What:** Replaces an old version by a new version of a view
- **Why:** To be able to change the definition of the view

**Acceptance Criteria**
The view is successfully altered.

**Create NoSQL View**
- **Who:** A component/vf-OS asset/vApp
- **What:** Creates a NoSQL view in a database
- **Why:** To be able to make simpler queries

**Acceptance Criteria**
The view is successfully created.

**Drop NoSQL View**
- **Who:** A component/vf-OS asset/vApp
- **What:** Drops an existing NoSQL view in a database
- **Why:** Because it is not needed anymore

**Acceptance Criteria**
The view is successfully dropped.
| **Replace NoSQL View** | **Who:** A component/vf-OS asset/vApp  
**What:** Replaces an old version by a new version of a view  
**Why:** To be able to change the definition of the view  
**Acceptance Criteria**  
The view is successfully altered |
| **DSUS012**  
**Create SQL Index** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Creates a SQL index of a table in a database  
**Why:** To be able to make faster queries  
**Acceptance Criteria**  
The index is successfully created |
| **DSUS013**  
**Drop SQL Index** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Drops an existing SQL index in a database  
**Why:** Because it is not needed anymore  
**Acceptance Criteria**  
The index is successfully dropped |
| **DSUS212**  
**Create NoSQL Index** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Creates a NoSQL index of a table in a database  
**Why:** To be able to make faster queries  
**Acceptance Criteria**  
The index is successfully created |
| **DSUS213**  
**Drop NoSQL Index** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Drops an existing NoSQL index in a database  
**Why:** Because it is not needed anymore  
**Acceptance Criteria**  
The index is successfully dropped |
| **DSUS014**  
**Describe SQL Table** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Ask for a description of a SQL table  
**Why:** To know its definition (columns, datatypes…)  
**Acceptance Criteria**  
The table definition is returned |
| **DSUS015**  
**Describe SQL View** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Ask for a description of a SQL view  
**Why:** To know its definition  
**Acceptance Criteria**  
The view definition is returned |
| **DSUS214**  
**Describe NoSQL Table** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Ask for a description of a NoSQL table  
**Why:** To know its definition (columns, datatypes…)  
**Acceptance Criteria**  
The table definition is returned |
| **DSUS215**  
**Describe NoSQL View** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Ask for a description of a NoSQL view  
**Why:** To know its definition  
**Acceptance Criteria**  

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Test Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSUS111</td>
<td>Create Rollup Task</td>
<td><strong>Who:</strong> A component/vf-OS asset/vApp&lt;br&gt;<strong>What:</strong> Creates a rollup task&lt;br&gt;<strong>Why:</strong> To perform periodic tasks to store aggregated data</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td>The rollup task is successfully created</td>
</tr>
<tr>
<td>DSUS112</td>
<td>List Rollup Tasks</td>
<td><strong>Who:</strong> A component/vf-OS asset/vApp&lt;br&gt;<strong>What:</strong> Lists the existing rollup tasks&lt;br&gt;<strong>Why:</strong> To know which rollup tasks exist</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td>The rollup task list is successfully retrieved</td>
</tr>
<tr>
<td>DSUS113</td>
<td>Get Rollup Task</td>
<td><strong>Who:</strong> A component/vf-OS asset/vApp&lt;br&gt;<strong>What:</strong> Lists the definition of a rollup task&lt;br&gt;<strong>Why:</strong> To know how is it defined</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td>The rollup task definition is successfully retrieved</td>
</tr>
<tr>
<td>DSUS114</td>
<td>Delete Rollup Tasks</td>
<td><strong>Who:</strong> A component/vf-OS asset/vApp&lt;br&gt;<strong>What:</strong> Deletes a rollup task&lt;br&gt;<strong>Why:</strong> Because it is not needed any more</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td>The rollup task is successfully deleted</td>
</tr>
<tr>
<td>DSUS115</td>
<td>Update Rollup Tasks</td>
<td><strong>Who:</strong> A component/vf-OS asset/vApp&lt;br&gt;<strong>What:</strong> Updates the definition of a rollup task&lt;br&gt;<strong>Why:</strong> Because it wants to update it</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td>The rollup task definition is successfully updated</td>
</tr>
<tr>
<td>DSUS016</td>
<td>Insert Entity</td>
<td><strong>Who:</strong> A component/vf-OS asset/vApp&lt;br&gt;<strong>What:</strong> Inserts an entity or a set of entities in a table of a database&lt;br&gt;<strong>Why:</strong> To store them</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td>The entity(s) is(are) correctly stored</td>
</tr>
<tr>
<td>DSUS017</td>
<td>Delete Entity</td>
<td><strong>Who:</strong> A component/vf-OS asset/vApp&lt;br&gt;<strong>What:</strong> Deletes a set of entities from a table of a database&lt;br&gt;<strong>Why:</strong> Because it is not needed any more</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td>The database is successfully dropped</td>
</tr>
<tr>
<td>DSUS107</td>
<td>Insert DataPoint</td>
<td><strong>Who:</strong> A component/vf-OS asset/vApp&lt;br&gt;<strong>What:</strong> Inserts a datapoint or a set of datapoints&lt;br&gt;<strong>Why:</strong> To store them</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong></td>
<td>The datapoint(s) is(are) correctly stored</td>
</tr>
<tr>
<td>DSUS108</td>
<td></td>
<td><strong>Description</strong></td>
</tr>
</tbody>
</table>
| **Delete DataPoint** | **Who:** A component/vf-OS asset/vApp  
**What:** Deletes a set of datapoints from a database  
**Why:** Because it is not needed anymore  
**Acceptance Criteria**  
The datapoint(s) is(are) correctly deleted |
| **DSUS216**  
**Insert Entity** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Inserts an entity or a set of entities in a table of a database  
**Why:** To store them  
**Acceptance Criteria**  
The entity(s) is(are) correctly stored |
| **DSUS217**  
**Delete Entity** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Deletes a set of entities from a table of a database  
**Why:** Because it is not needed anymore  
**Acceptance Criteria**  
The entity(s) is(are) correctly deleted |
| **DSUS018**  
**Merge entity** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Updates an existing entity(s) updating the properties (not deleting them)  
**Why:** To change the value of any property(s)  
**Acceptance Criteria**  
The entity(s) property(s) is(are) correctly updated |
| **DSUS019**  
**Update entity** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Updates an existing entity(s) replacing it all with the new values  
**Why:** To change the value of all properties of an entity, so it can be used to delete any property  
**Acceptance Criteria**  
All the entity(s) property(s) is(are) replaced by the new values |
| **DSUS218**  
**Merge entity** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Updates an existing entity(s) updating the properties (not deleting them)  
**Why:** To change the value of any property(s)  
**Acceptance Criteria**  
The entity(s) property(s) is(are) correctly updated |
| **DSUS020**  
**Query entity** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Performs a query on a database table(s) or view(s)  
**Why:** To get a set of entities  
**Acceptance Criteria**  
The entity(s) is(are) correctly retrieved |
| **DSUS109**  
**Query Metrics** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Performs a query on any metric(s)  
**Why:** To get a set of datapoints and tags  
**Acceptance Criteria**  
The datapoints and tags are correctly retrieved |
| **DSUS110**  
**Query Metric Tags** | **Description**  
**Who:** A component/vf-OS asset/vApp  
**What:** Performs a query on any metric(s)  
**Why:** To get a set of tags (no datapoints)  
**Acceptance Criteria** |
<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSUS220</td>
<td>Query entity</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Performs a query on a database table(s) or view(s)</td>
</tr>
<tr>
<td></td>
<td>Why: To get a set of entities</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The entity(s) is(are) correctly retrieved</td>
</tr>
<tr>
<td>DSUS021</td>
<td>Create role</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Creates a role</td>
</tr>
<tr>
<td></td>
<td>Why: To manage permissions on a database</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The role is successfully created</td>
</tr>
<tr>
<td>DSUS022</td>
<td>Drop role</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Drops a role</td>
</tr>
<tr>
<td></td>
<td>Why: Because it is not needed anymore</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The role is successfully dropped</td>
</tr>
<tr>
<td>DSUS023</td>
<td>Query roles</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Queries the existing roles of a database</td>
</tr>
<tr>
<td></td>
<td>Why: To get the list of the existing roles and their permissions</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The list of roles is successfully returned</td>
</tr>
<tr>
<td>DSUS024</td>
<td>AddRoleMember</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Adds a user to be a member of a role</td>
</tr>
<tr>
<td></td>
<td>Why: So that that user inherits the permissions of the role</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The user is a member of a role</td>
</tr>
<tr>
<td>DSUS025</td>
<td>DeleteRoleMember</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Deletes a user as a member of a role</td>
</tr>
<tr>
<td></td>
<td>Why: So that the user does not have the permissions of that role anymore</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The user is not member of a role anymore</td>
</tr>
<tr>
<td>DSUS21</td>
<td>Create role</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Creates a role</td>
</tr>
<tr>
<td></td>
<td>Why: To manage permissions on a database</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The role is successfully created</td>
</tr>
<tr>
<td>DSUS22</td>
<td>Drop role</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Who: A component/vf-OS asset/vApp</td>
</tr>
<tr>
<td></td>
<td>What: Drops a role</td>
</tr>
<tr>
<td></td>
<td>Why: Because it is not needed anymore</td>
</tr>
<tr>
<td></td>
<td>Acceptance Criteria</td>
</tr>
<tr>
<td></td>
<td>The role is successfully dropped</td>
</tr>
</tbody>
</table>

D2.2: Functional Specifications & Mockups - Vs: 1.0 - Public
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Query roles</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Queries the existing roles of a database</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: To get the list of the existing roles and their permissions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The list of roles is successfully returned</td>
</tr>
<tr>
<td>DSUS224</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddRoleMember</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Adds a user to be member of a role</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: So that that user inherits the permissions of the role</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The user is member of a role</td>
</tr>
<tr>
<td>DSUS225</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeleteRoleMember</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Deletes a user as a member of a role</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: So that that user does not have the permissions of that role anymore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The user is not member of a role anymore</td>
</tr>
<tr>
<td>DSUS026</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Creates a user</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: To manage permissions on a database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The user is successfully created</td>
</tr>
<tr>
<td>DSUS027</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drop user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Drops a user</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: Because it is not needed anymore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The user is successfully dropped</td>
</tr>
<tr>
<td>DSUS028</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query users</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Queries the existing users of a database</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: To get the list of the existing users and their permissions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The list of users is successfully returned</td>
</tr>
<tr>
<td>DSUS116</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Creates a user</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: To manage permissions on a database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The user is successfully created</td>
</tr>
<tr>
<td>DSUS117</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drop user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Drops a user</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: Because it is not needed anymore</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The user is successfully dropped</td>
</tr>
<tr>
<td>DSUS226</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: A component/vf-OS asset/vApp</td>
<td>Creates a user</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: To manage permissions on a database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.2.1.2 UI Mockups and Sequence Diagrams

The following sub-sections describe the sequence diagrams describing the interaction. This component does not have any user interface, so there is no UI Mockup.

5.2.1.2.1 Relational database API

User Stories, from DSUS001 to DSUS031 are related to relational data storage. They all follow this sequence diagram pattern.
In every different story, it changes the cmdName, the contents of cmdParams and the contents of cmdResponse. For example:

<table>
<thead>
<tr>
<th>User Story</th>
<th>cmdName</th>
<th>cmdParams</th>
<th>cmdResponse</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSUS001</td>
<td>createSQLDatabase</td>
<td>databaseName</td>
<td>OK/Error code</td>
</tr>
<tr>
<td>DSUS002</td>
<td>dropSQLDatabase</td>
<td>databaseName</td>
<td>OK/Error code</td>
</tr>
<tr>
<td>DSUS006</td>
<td>createSQLTable</td>
<td>tableDefinition</td>
<td>OK/Error code</td>
</tr>
<tr>
<td>DSUS020</td>
<td>queryEntity</td>
<td>querySpec</td>
<td>recordset/Error code</td>
</tr>
</tbody>
</table>

5.2.1.2.2 Timeseries database API
User Stories, from DSUS101 to DSUS117 are related to time series data storage. They all follow this sequence diagram pattern.
### 5.2.1.2.3 NoSQL database API

User Stories, from DSUS201 to DSUS231 are related to NoSQL data storage. They all follow this sequence diagram pattern.

![NoSQL Database API Sequence Diagrams Pattern](image)

In every different story it changes the `cmdName`, the contents of `cmdParams` and the contents of `cmdResponse`. For example:

<table>
<thead>
<tr>
<th>User Story</th>
<th>cmdName</th>
<th>cmdParams</th>
<th>cmdResponse</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSUS201 Create NoSQL Database</td>
<td>createNoSQLDatabase</td>
<td>databaseName</td>
<td>OK/Error code</td>
</tr>
<tr>
<td>DSUS202 Drop NoSQL Database</td>
<td>dropNoSQLDatabase</td>
<td>databaseName</td>
<td>OK/Error code</td>
</tr>
<tr>
<td>DSUS206 Create SQL Table</td>
<td>createNoSQLTable</td>
<td>tableDefinition</td>
<td>OK/Error code</td>
</tr>
<tr>
<td>DSUS220 Query entity</td>
<td>queryNoSQLEntity</td>
<td>querySpec</td>
<td>recordset/Error code</td>
</tr>
</tbody>
</table>

### 5.2.1.3 Interaction Description

From the previous description of the functionality covered by the Storage module, a deeper level of detail regarding the main subcomponents of the component emerges. Following there is a picture showing the flow of information exchange between the Storage component and vf-OS components/Assets/vApps.
Figure 139: Storage Component Interaction Diagram

As it can be seen the users of the Storage components may be other components, vf-OS assets and vApps. For each kind of database there is three main information flows:

- A request to save data:
  - The request will arrive at an XXXRequestProcessor1. The specification of the request will arrive in a DBrequest class. In this class, apart from the command and the command parameters, the database user and password must be settled.
  - The XXXRequestProcessor will process the request and return the result in an XXXResponse class. This class will contain a success/failure code, and an error code in case of failure.

- A request to query data:
  - The request will arrive at an XXXRequestProcessor. The specification of the request will arrive in a DBrequest class. In this class, apart from the command and the command parameters, the database user and password must be settled.
  - The XXXRequestProcessor will process the request and return the result in an XXXResponse class. This class will contain a success/failure code, an error code in case of failure and a recordset or TagsDataPointSet (in the case of TimesSeriesBD) with the actual data.

---

1 XXX: RelationalDB or TimeSeriesDB or NoSQLDB
A request to administrate the data (create/drop databases, tables, views, indexes...):

- The request will arrive to a XXXRequestProcessor. The specification of the request will arrive in a DBrequest class. In this class, apart from the command and the command parameters, the database user and password must be settled.
- The XXXRequestProcessor will process the request and return the result in an XXXResponse class. This class will contain a success/failure code, an error code in case of failure and a recordset with the actual data.

Figure 140: Storage Component Classes and Information Exchanged

5.2.2 Data Transformation

The Data Transformation component provides the features to integrate data from concrete existing software systems by executing the Manufacturing Maps created in the Data Mapping component, ie transforming data from its source format to its destination format. The maps created in the Data Mapping component will be deployed and encapsulated as services to be finally exposed as software mini-packages, ie Docker containers. These mini-packages, containing the transformation routines, will be uploaded and published in the vf-Store to advertise and commercialise them.
5.2.2.1 Behaviour and Functionality

The Data Transformation component provides a set of functionalities that could be grouped on the following features:

- **Transform**: where the Manufacturing Maps can be executed in the form of a standalone service. This service contains the rules defined and deployed from the Manufacturing Map to transform a specific syntax format A into format B which could then, for example, be used as part of a process

- **Submit Usage Data**: where usage data will be captured and communicated to the Platform

Following, there is a story map where the main features, epics and user stories for the Data Transformation component have been identified (see Figure 141).

![Data Transformation Story Map](chart)

**Figure 141: Data Transformation Story Map**

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTUS001</td>
<td>Get invocation</td>
</tr>
</tbody>
</table>

**Description**

**Who:** Data Transformation  
**What:** get invocation request from Service Call  
**Why:** so that the Transformation engine can execute the right transformation service  

**Acceptance Criteria**  
The invocation is relayed to the transformation engine

<table>
<thead>
<tr>
<th>DTUS002</th>
<th>Description</th>
</tr>
</thead>
</table>
**Connect to vF-Store**

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Data Transformation</th>
<th>What: connect to vF-Store with the credentials as directed by the vF-OS Security component</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why:</strong></td>
<td>so that the transformed data can be stored</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

The vF-Store is accessible

**Unpack Routines**

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Data Transformation</th>
<th>What: unpack a mapping routines set</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why:</strong></td>
<td>so that the Transformation engine can read the transformation steps determined in the Map</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

The routines are successfully unpacked and the transformation steps are available for the transformation engine

**Read Data**

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Data Transformation</th>
<th>What: reads source data as input parameter from the invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why:</strong></td>
<td>so that the input data can be transformed by executing the transformation services</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

The input data is available for transformation

**Transform**

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Data Transformation</th>
<th>What: transforms the data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why:</strong></td>
<td>so that the routines of the mapping are executed</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

The transformation is successfully executed

**Push Transformed Data**

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Data Transformation</th>
<th>What: pushes transformed data back to the calling vApp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why:</strong></td>
<td>so that the transformed (output) data can continue its way in the processing within the vApp</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

The transformed (output) data is available for the vApp

**Store Transformed Data**

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Data Transformation</th>
<th>What: store the transformed data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why:</strong></td>
<td>so that the transformed data is accessible without having to re-execute the transformation service again</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

The transformed data is successfully stored

**Submit Usage Data**

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Data Transformation</th>
<th>What: submit usage data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why:</strong></td>
<td>so that the platform can make use of this data for monitoring purposes</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

The usage data is successfully received by the Platform

---

5.2.2.2 UI mockups and Sequence Diagrams

As the Data Transformation component is a service-based component, there are no UIs attached to it. Therefore, this sub-section only describes its sequence diagrams.

5.2.2.2.1 Transform
This story deals with the preparation steps prior to executing a transformation service and the steps to execute a map.

The main steps/functionalities are as follows:

- **Preparations:**
  - Get invocation
  - Connect to vf-Store

- **Execute Map:**
  - Unpack Routines
  - Read Data
  - Transform
  - Push Transformed Data
  - Store Transformed Data

![Transform Sequence Diagram](image)

**Figure 142: Transform Sequence Diagram**

### 5.2.2.2 Submit Usage Data

This feature provides the capability to deploy and publish a map after it has been generated by the Business Analyst.

There is only one step corresponding to this feature:

- Submit Usage Data
5.2.2.3 Interaction description

From the previous description of the functionality covered by the Data Transformation component, a deeper level of detail regarding the main modules of the component and the interaction between those modules and other vf-OS components emerges. Whilst next Figure 144 shows the Architecture diagram, as presented in D2.1, the accompanying text focuses on the interactions and data exchange between the Data Transformation and other vf-OS components.
The main interactions of Data Transformation modules with other components are:

- **Transformation Engine (Functional Implementation):** it is the module in charge of offering the functionality of effectively executing the transformation routines. These transformation routines, through the corresponding APIs store the transformed data and receive the execution command together with the data to be transformed. The APIs to carry out these activities are the Store Transformed Data API and the Service Call interfaces. The main information flows are:
  - It sends the transformed data to the vf-Store (interaction with Storage component)
  - It receives the execution command together with the data to be transformed and sends back the transformed data from a vApp (via the Process Execution) or directly from the Messaging

- **Monitoring Services:** it is the module in charge of submitting monitoring and usage data of the executions of transformations. The main information flows exchanged with external components are:
• It sends the usageData of the transformations executed (interaction with the Platform component)

![Data Transformation Component Classes and Information Exchanged](image)

**Figure 145: Data Transformation Component Classes and Information Exchanged**

### 5.2.3 Data Analytics

The Data Analytics component provides the features for deriving events from stream and historic process data within the manufacturing domain. Data from the sources will be fed into classical staging and final information warehouses and where possible existing partner applications will be used. The data targets can be individuals on the shop floor to executive roles and the key is to filter and display it with the relevant user in mind.

#### 5.2.3.1 Behaviour and Functionality

The Data Analytics component provides a set of functionalities that could be grouped on the following features:

- **Visual Analytics**: where the analytics algorithms can be executed in the form of live graphs. These graphics should be useful for those organisations that do not want to dig deeply into their data and do not want to make use of advanced analytics techniques
- **Advanced Analytics**: where several analytics algorithms can be executed on the basis of the data received and feedback and forecasting to the user in the form of a report or a graphical representation
- **Stream Analytics**: where rules for analysing certain, ie those compliant with rules, thresholds, alerts, etc, data lively published coming from the source will be specified.

Following, there is a story map where the main features, epics and user stories for the Data Analytics component have been identified (see Figure 146).
Figure 146: Data Analytics Story Map

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
</table>
| DAUS001 Create analytics | **Description**  
Who: Data Analytics  
What: create analytics  
Why: so that the user can perform the analytics that fit their needs  

**Acceptance Criteria**  
The user-defined analytics is successfully created |
| DAUS002 Read analytics | **Description**  
Who: Data Analytics  
What: read existing analytics  
Why: so that the user can re-use any analytics already created  

**Acceptance Criteria**  
The user-desired analytics is successfully read |
| DAUS003 Update analytics | **Description**  
Who: Data Analytics  
What: update analytics  
Why: so that the user can update the read analytics with further/different configuration  

**Acceptance Criteria** |
<table>
<thead>
<tr>
<th>DAUS004</th>
<th>Delete analytics</th>
<th>The new version of the analytics is successfully updated</th>
</tr>
</thead>
</table>
|          | **Description** | Who: Data Analytics  
What: delete analytics  
Why: so that the user can remove outdated analytics from the storage |
|          | **Acceptance Criteria** | The desired analytics is successfully removed from the repository |
| DAUS005  | Create query    | Description |
|          | **Description** | Who: Data Analytics  
What: create an analytics query  
Why: so that the user can personalise the analytics that fit their needs |
|          | **Acceptance Criteria** | The user-defined query analytics is successfully created |
| DAUS006  | Read query      | Description |
|          | **Description** | Who: Data Analytics  
What: read an analytics query  
Why: so that the user can re-use an already created analytics query |
|          | **Acceptance Criteria** | The user-defined query analytics is successfully read |
| DAUS007  | Execute query   | Description |
|          | **Description** | Who: Data Analytics  
What: execute an analytics query  
Why: so that the user can make use of an analytics query |
|          | **Acceptance Criteria** | The user-defined query analytics is successfully executed |
| DAUS008  | Specify type of query  
(eg OLAP, Data Extraction...) | Description |
|          | **Description** | Who: Data Analytics  
What: typify an analytics query  
Why: so that the user can specify which is the type of the query to be used |
|          | **Acceptance Criteria** | The user-defined query analytics is successfully typified |
| DAUS009  | Visualise query | Description |
|          | **Description** | Who: Data Analytics  
What: view an analytics query  
Why: so that the user can visualise, and check, an analytics query |
|          | **Acceptance Criteria** | The user-defined query analytics is successfully displayed |
| DAUS010  | Update query | Description |
|          | **Description** | Who: Data Analytics  
What: update an analytics query  
Why: so that the user can maintain different versions of the analytics queries |
|          | **Acceptance Criteria** | The user-defined query analytics is successfully updated |
| DAUS011  | Delete query | Description |
|          | **Description** | Who: Data Analytics  
What: delete an analytics query  
Why: so that the user can remove analytics queries that are no longer used/necessary |
|          | **Acceptance Criteria** | The user-defined query analytics is successfully dropped |
| DAUS012  | Select type of graph | Description |
|          | **Description** | Who: Data Analytics  
What: select type of analytics graph  
Why: so that the user can specify which is the type of graph to be produced |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAUS013</td>
<td><strong>Select parameters</strong></td>
<td>The type of graph is successfully selected</td>
</tr>
</tbody>
</table>
|               | **Who:** Data Analytics  
**What:** select parameters to be displayed in the graph  
**Why:** so that the user can specify which are the KPIs to be calculated/shown |                                                                                     |
|               | **Acceptance Criteria**                                                     | The KPIs/parameters are successfully selected                                      |
|               | **Description**                                                             |                                                                                     |
| DAUS014       | **Select time span**                                                        | The time span is successfully selected                                              |
|               | **Who:** Data Analytics  
**What:** select time span to be displayed in the graph  
**Why:** so that the user can specify which is the time span that will cover the graph to be calculated/shown |                                                                                     |
|               | **Acceptance Criteria**                                                     |                                                                                     |
|               | **Description**                                                             |                                                                                     |
| DAUS015       | **Visualise graph**                                                         | The graph is successfully displayed                                                 |
|               | **Who:** Data Analytics  
**What:** view an analytics graph  
**Why:** so that the user can visualise, and check, the results of an analytics query in the form of a graph |                                                                                     |
|               | **Acceptance Criteria**                                                     |                                                                                     |
|               | **Description**                                                             |                                                                                     |
| DAUS016       | **Select dataset**                                                          | The desired dataset is successfully selected                                        |
|               | **Who:** Data Analytics  
**What:** select dataset  
**Why:** so that the user can run the regression analytics |                                                                                     |
|               | **Acceptance Criteria**                                                     | The type of regression is successfully selected                                     |
|               | **Description**                                                             |                                                                                     |
| DAUS017       | **Select regression type** (linear, least squares, polynomial...)           | The grade is successfully selected                                                   |
|               | **Who:** Data Analytics  
**What:** select type of regression algorithms  
**Why:** so that the user can specify which is the type of regression to be executed |                                                                                     |
|               | **Acceptance Criteria**                                                     |                                                                                     |
|               | **Description**                                                             |                                                                                     |
| DAUS018       | **Select grade of equation (in case of polynomial)**                        | The user judges acceptable the RMSE (Root-Mean-Square-Deviation Error) and the R2 provided by the executed regression method |
|               | **Who:** Data Analytics  
**What:** select the grade of the regression equation to be calculated, in case the regression selected is a polynomial  
**Why:** so that the user can specify which is the grade for the regression to be executed |                                                                                     |
|               | **Acceptance Criteria**                                                     | The user judges acceptable the RMSE (Root-Mean-Square-Deviation Error) and the R2 provided by the executed regression method |
|               | **Description**                                                             |                                                                                     |
| DAUS019       | **Show results of regression**                                              |                                                                                     |
|               | **Who:** Data Analytics  
**What:** show the results of the regression method  
**Why:** so that the user can assess if the results are satisfactory enough for their purposes |                                                                                     |
|               | **Acceptance Criteria**                                                     |                                                                                     |
|               | **Description**                                                             |                                                                                     |
| DAUS020       | **Forecasting**                                                            |                                                                                     |
|               | **Who:** Data Analytics  
**What:** do forecasting  
**Why:** so that the user can be assessed with forecasted values as per the regression method selected |                                                                                     |
<p>|               | <strong>Acceptance Criteria</strong>                                                     |                                                                                     |
|               | <strong>Description</strong>                                                             |                                                                                     |</p>
<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAUS021 Select dataset</td>
<td>The user is shown with forecasted results that are useful for their duties</td>
<td></td>
</tr>
<tr>
<td>DAUS022 Split dataset in train and test</td>
<td>The desired dataset is successfully selected</td>
<td></td>
</tr>
<tr>
<td>DAUS023 Select ML method (eg Random Forest, Decision Tree...)</td>
<td>The dataset is split in two random sets</td>
<td></td>
</tr>
<tr>
<td>DAUS024 Train ML method</td>
<td>The type of ML is successfully selected</td>
<td></td>
</tr>
<tr>
<td>DAUS025 Test ML method</td>
<td>The ML method is trained with the &quot;train&quot; dataset</td>
<td></td>
</tr>
<tr>
<td>DAUS026 Show results of ML</td>
<td>The trained ML method is tested with the &quot;test&quot; dataset</td>
<td></td>
</tr>
<tr>
<td>DAUS027 Forecasting</td>
<td>The user judges acceptable the RMSE (Root-Mean-Square-Deviation Error) and the R2 provided by the tested ML method</td>
<td></td>
</tr>
<tr>
<td>DAUS028 Select dataset</td>
<td>The user is shown with forecasted results that are useful for their duties</td>
<td></td>
</tr>
</tbody>
</table>

**DAUS022 Split dataset in train and test**

**Description**
Who: Data Analytics  
What: select dataset  
Why: so that the to-be selected method can be trained and tested before producing forecasts

**Acceptance Criteria**
The dataset is split in two random sets

**DAUS023 Select ML method (eg Random Forest, Decision Tree...)**

**Description**
Who: Data Analytics  
What: select type of ML methods  
Why: so that the user can specify which is the type of ML to be executed

**Acceptance Criteria**
The type of ML is successfully selected

**DAUS024 Train ML method**

**Description**
Who: Data Analytics  
What: use the "train" dataset to train the ML method selected  
Why: so that the selected ML method can be trained before producing forecasts

**Acceptance Criteria**
The ML method is trained with the "train" dataset

**DAUS025 Test ML method**

**Description**
Who: Data Analytics  
What: use the "test" dataset to test the trained ML method  
Why: so that the selected ML method can be tested before producing meaningful forecasts

**Acceptance Criteria**
The trained ML method is tested with the "test" dataset

**DAUS026 Show results of ML**

**Description**
Who: Data Analytics  
What: show the results of the trained and tested ML method  
Why: so that the user can assess if the tests are satisfactory enough for their purposes

**Acceptance Criteria**
The user judges acceptable the RMSE (Root-Mean-Square-Deviation Error) and the R2 provided by the tested ML method

**DAUS027 Forecasting**

**Description**
Who: Data Analytics  
What: do forecasting  
Why: so that the user can be assessed with forecasted values as per the regression method selected

**Acceptance Criteria**
The user is shown with forecasted results that are useful for their duties

**DAUS028 Select dataset**

**Description**
Who: Data Analytics  
What: select dataset  
Why: so that the user can run the classification analytics

**Acceptance Criteria**
| DAUS029 | Split dataset in train and test | The desired dataset is successfully selected |
| Description | Who: Data Analytics | What: randomly split dataset in two sets |
| | Why: so that the to-be selected method can be trained and tested before producing forecasts |
| Acceptance Criteria | The dataset is split in two random sets |

| DAUS030 | Select Classification method (eg K-means, KNN...) | Description |
| | Who: Data Analytics | What: select type of classification algorithms |
| | Why: so that the user can specify which is the type of classification to be executed |
| Acceptance Criteria | The type of classification is successfully selected |

| DAUS031 | Train Classification method | Description |
| | Who: Data Analytics | What: use the "train" dataset to train the Classification method selected |
| | Why: so that the selected Classification method can be trained before producing forecasts |
| Acceptance Criteria | The Classification method is trained with the "train" dataset |

| DAUS032 | Test Classification method | Description |
| | Who: Data Analytics | What: use the "test" dataset to test the trained Classification method |
| | Why: so that the selected Classification method can be tested before producing meaningful forecasts |
| Acceptance Criteria | The trained Classification method is tested with the "test" dataset |

| DAUS033 | Show results of Classification | Description |
| | Who: Data Analytics | What: show the results of the trained and tested Classification method |
| | Why: so that the user can assess if the tests are satisfactory enough for their purposes |
| Acceptance Criteria | The user judges acceptable the RMSE (Root-Mean-Square-Deviation Error) and the R2 provided by the tested Classification method |

| DAUS034 | Forecasting | Description |
| | Who: Data Analytics | What: do forecasting |
| | Why: so that the user can be assessed with forecasted values as per the regression method selected |
| Acceptance Criteria | The user is shown with forecasted results that are useful for their duties |

| DAUS035 | Connect to stream source | Description |
| | Who: Data Analytics | What: connect to a source in the form of a stream |
| | Why: so that the user can run analytics on stream data |
| Acceptance Criteria | The stream source is successfully connected and is responding |

<p>| DAUS036 | Connect to vf-OS Storage | Description |
| | Who: Data Analytics | What: connect to a source stored in the vf-OS Storage |
| | Why: so that the user can run analytics on vf-OS stored data |
| Acceptance Criteria | |</p>
<table>
<thead>
<tr>
<th>DAUS037</th>
<th>ETL</th>
<th>The vf-OS Storage is successfully connected and is responding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Who: Data Analytics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>What: execute ETL routines on vf-OS Storage stored data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why: so that the user can run analytics on stored data</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The data from the vf-OS Storage is successfully loaded to the Analytics component</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS038</th>
<th>Connect to batch source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who: Data Analytics</strong></td>
<td><strong>What: execute ETL routines on vf-OS Storage stored data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Why: so that the user can run analytics on stored data</strong></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The batch source is successfully connected</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS039</th>
<th>ETL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who: Data Analytics</strong></td>
<td><strong>What: execute ETL routines on batch data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Why: so that the user can run analytics on batch data</strong></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The batch data is successfully loaded to the Analytics component</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS040</th>
<th>Create an analysis module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who: Data Analytics</strong></td>
<td><strong>What: create a Stream analysis module</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Why: so that the user can perform the analysis that fits their needs composed by schema (incoming data), thresholds, filtering sentences, alerts sentences</strong></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The user-defined Stream Analysis module is successfully created</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS041</th>
<th>Read an analysis module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who: Data Analytics</strong></td>
<td><strong>What: read a Stream analysis module</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Why: so that the user can update the analysis</strong></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The user-defined Stream Analysis module is successfully read. If there are incoherencies between sentences, the errors are also returned</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS042</th>
<th>Get analyses modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who: Data Analytics</strong></td>
<td><strong>What: get all analyses defined</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Why: so that the user can view its own analyses</strong></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The Stream analysis UI shows only the user-defined Stream Analysis modules general information are shown in the Stream analysis UI, so that the user can view / update own analyses</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS043</th>
<th>Verify an analysis module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who: Data Analytics</strong></td>
<td><strong>What: verify analysis</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Why: so that this analysis is coherent: incoming data, thresholds, and conditions of the sentences</strong></td>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The user-defined analysis module is verified: schema, thresholds and conditions. If there are some incoherencies, the first error is showed to the user, and the module is deactivated</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS044</th>
<th>Drop an Analysis module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who: Data Analytics</strong></td>
<td><strong>What: drop a Stream analysis module and all thresholds, sentences</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Why: so that the user can drop an analysis</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAUS045</td>
<td>Activate an analysis module</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Who: Data Analytics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What: activate a Stream analysis module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why: so that the user can load an analysis module for execution</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The user-defined Stream Analysis module is successfully loaded, therefore, executed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS046</th>
<th>Deactivate an analysis module</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Data Analytics</td>
<td></td>
</tr>
<tr>
<td>What: deactivate a Stream analysis module</td>
<td></td>
</tr>
<tr>
<td>Why: so that the user can unload an analysis module from execution</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The user-defined Stream Analysis module is successfully unloaded (stopped)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS047</th>
<th>Get stream data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Data Analytics</td>
<td></td>
</tr>
<tr>
<td>What: get stream data</td>
<td></td>
</tr>
<tr>
<td>Why: so that the user can view / modify the stream data associated to its own analyses</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The stream data of a stream analysis module is shown to the user, so that she/he can update its definition</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS048</th>
<th>Subscribe to stream data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Data Analytics</td>
<td></td>
</tr>
<tr>
<td>What: subscribe to stream data</td>
<td></td>
</tr>
<tr>
<td>Why: so that the analysis module will subscribe to different topics and receive all related streaming data</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The user-defined Stream Analysis module is successfully subscribed to a topic. When data is received, the incoming data is de-serialised and sent to the Stream analysis Service to analyse it</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS049</th>
<th>Update data subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Data Analytics</td>
<td></td>
</tr>
<tr>
<td>What: update a data subscription (schema of incoming data and / or topic)</td>
<td></td>
</tr>
<tr>
<td>Why: so that the incoming data will fit the analysis needs</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The user-defined updating of the incoming data is performed correctly</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS050</th>
<th>Drop data subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Data Analytics</td>
<td></td>
</tr>
<tr>
<td>What: drop a data subscription</td>
<td></td>
</tr>
<tr>
<td>Why: this streaming data subscription has become obsolete</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The data subscription is dropped correctly</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS051</th>
<th>Create threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Data Analytics</td>
<td></td>
</tr>
<tr>
<td>What: create threshold</td>
<td></td>
</tr>
<tr>
<td>Why: so that the sentences of this analysis can use this threshold in their conditions part</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The user-defined threshold is successfully created</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS052</th>
<th>Get thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Data Analytics</td>
<td></td>
</tr>
<tr>
<td>What: get thresholds</td>
<td></td>
</tr>
<tr>
<td>Why: so that the user can view / modify the name, values, types of the...</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **DAUS053** Create sentence | **Description**  
Who: Data Analytics  
What: create alert sentence  
Why: so that this analysis will create an alert when incoming data meets some conditions | **Acceptance Criteria**  
The user-defined alert sentence is successfully created |
| **DAUS054** Get sentences | **Description**  
Who: Data Analytics  
What: get sentences  
Why: so that the user can view / modify the name, types, incoming data and conditions of the sentences associated to an analysis module | **Acceptance Criteria**  
Stream analysis UI show the list of sentences available. They can be used to define the alert / filtering sentences |
| **DAUS055** Update threshold | **Description**  
Who: Data Analytics  
What: update threshold  
Why: sometimes it is necessary to change the value or to change the name of threshold to correct syntax errors, or adjust execution results | **Acceptance Criteria**  
The threshold update is performed |
| **DAUS056** Drop threshold | **Description**  
Who: Data Analytics  
What: update threshold  
Why: the threshold has become obsolete | **Acceptance Criteria**  
The threshold is successfully dropped |
| **DAUS057** Update sentence | **Description**  
Who: Data Analytics  
What: update filtering sentence  
Why: the sentence when condition must be refined. Sometimes is necessary to change conditions (condition combination, schema properties, or threshold names inside conditions to correct syntax errors, or adjust execution results | **Acceptance Criteria**  
The sentence is successfully updated |
| **DAUS058** Drop sentence | **Description**  
Who: Data Analytics  
What: drop sentence  
Why: this sentence has become obsolete | **Acceptance Criteria**  
The sentence is successfully dropped |
| **DAUS059** Activate sentence | **Description**  
Who: Data Analytics  
What: activate sentence  
Why: this sentence has been stopped before | **Acceptance Criteria**  
The sentence is successfully activated |
| Deactivate sentence | **Who:** Data Analytics  
**What:** deactivate sentence  
**Why:** this sentence must be refined or dropped |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The sentence is successfully stopped. The rest of sentences of the module are running</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS061 Execute an active analysis</th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Who:** Data Analytics  
**What:** provide the mechanism to execute an active analysis  
**Why:** the simplified mechanism |
| **Acceptance Criteria** | All active stream analysis are executing successfully, analysing incoming data, and publishing / persisting results |
| **Description** |  |

<table>
<thead>
<tr>
<th>DAUS062 Provide the mechanism to execute all active analysis</th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Who:** Data Analytics  
**What:** provide the mechanism to execute all active analysis  
**Why:** the analysis is performing with the streaming data |
| **Acceptance Criteria** | All active stream analysis are executing successfully, analysing incoming data, and publishing / persisting results |
| **Description** |  |

<table>
<thead>
<tr>
<th>DAUS063 Publish Alerts</th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Who:** Data Analytics  
**What:** publish results (alerts) of the selected analysis module  
**Why:** so that any component can subscribe to the alerts |
| **Acceptance Criteria** | Any component can subscribe to the results of the analysis |
| **Description** |  |

<table>
<thead>
<tr>
<th>DAUS064 Publish Filtered data</th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Who:** Data Analytics  
**What:** publish results (filtered data) of the selected analysis  
**Why:** so that any component can subscribe to the filtered data |
| **Acceptance Criteria** | Any component can subscribe to the results of the analysis |
| **Description** |  |

<table>
<thead>
<tr>
<th>DAUS065 Persists Alerts</th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Who:** Data Analytics  
**What:** persists results(alerts) of the selected analysis  
**Why:** so that any component can get the historical of alerts for an analysis module |
| **Acceptance Criteria** | Any component can get the historical of alerts for the analysis module |
| **Description** |  |

<table>
<thead>
<tr>
<th>DAUS066 Get alerts</th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Who:** Data Analytics  
**What:** get results (alerts) of the selected analysis  
**Why:** so that any component can work with them |
| **Acceptance Criteria** | Any component gets all alerts for the selected analysis |
| **Description** |  |

<table>
<thead>
<tr>
<th>DAUS067 Update an alert status</th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Who:** Data Analytics  
**What:** update an alert status of the selected analysis  
**Why:** so that this alert is not returned again in get Alerts method |
| **Acceptance Criteria** | Once an alert is verified it is no longer displayed in the interface |
| **Description** |  |

<table>
<thead>
<tr>
<th>DAUS068 Persists filtered data</th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Who:** Data Analytics  
**What:** persists results (filtered data) of the selected analysis  
**Why:** so that any component can get the historical of filtered data |
| **Description** |  |
5.2.3.2 UI mockups and Sequence Diagrams

The following sub-sections describe the UI mockups and sequence diagrams of the Data Analytics component.

5.2.3.2.1 Re-use Analytics

This story deals with the preparation steps to create, store and, thus, re-use analytics operations at the same time as queries.

The main steps/functionalities are as follows:

- **CRUD on Analytics:**
  - Create analytics
  - Read analytics
  - Update analytics
  - Delete analytics

- **CRUD on Query:**
  - Create query
  - Read query
  - Execute query
  - Specify type of query (eg OLAP, Data Extraction…)
  - Visualise query
  - Update query
  - Delete query

---

<table>
<thead>
<tr>
<th>DAUS069 Get filtered data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Who:</strong> Data Analytics</td>
</tr>
<tr>
<td></td>
<td><strong>What:</strong> get results (filtered data) of the selected analysis</td>
</tr>
<tr>
<td></td>
<td><strong>Why:</strong> so that any component can work with them</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAUS070 Update status of filtered data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Who:</strong> Data Analytics</td>
</tr>
<tr>
<td></td>
<td><strong>What:</strong> update a filtered data status of the selected analysis</td>
</tr>
<tr>
<td></td>
<td><strong>Why:</strong> so that this data is not returned again in get Alerts method</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any component can get the historical of analysis filtered data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any component gets all filtered data for the selected analysis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Once a filtered data is verified it is no longer displayed in the interface</td>
</tr>
</tbody>
</table>
Figure 147: Create Analytics Sequence Diagram

Figure 148: Read Analytics Sequence Diagram
Figure 149: Update Analytics Sequence Diagram

Figure 150: Delete Analytics Sequence Diagram
Figure 151: Create Query Sequence Diagram

Figure 152: Read Query Sequence Diagram
Figure 153: Execute and Visualise Analytics Query Sequence Diagram

Figure 154: Execute and Visualise Data Extraction Query Sequence Diagram
D2.2: Functional Specifications & Mockups - Vs: 1.0 - Public

The UIs for accessing the re-use of analytics, including the queries, are as follows:
Figure 157: Stored Queries UI Mockup

Figure 158: Build Query UI Mockup
5.2.3.2.2 Visual Analytics
This story deals with the steps to create, store and, thus, execute visual analytics operations in the form of graphs.

The main steps / functionalities are as follows:

- Select type of graph
- Select parameters
- Select time span
- Visualised graph
The UI for accessing the design of graphs is as follows:

Figure 160: Design Graph Sequence Diagram

Figure 161: Design Graph UI Mockup
5.2.3.2.3 Analytics
This story deals with the steps to create, and execute analytics algorithms categorised in regression, machine learning (ML) and classification and clustering algorithms.

The main steps/functionality are as follows:

- **Regression Analytics:**
  - Select dataset
  - Select Regression type (eg, linear, least-squares, polynomial, …)
  - Select grade of equation (in case of polynomial)
  - Show results of Regression
  - Forecasting

- **Machine Learning Analytics:**
  - Select dataset
  - Split dataset in train and test
  - Select ML method (eg, Random Forest, Decision Tree, …)
  - Train ML method
  - Test ML method
  - Show results of ML
  - Forecasting

- **Classification Analytics:**
  - Select dataset
  - Split dataset in train and test
  - Select Classification method (eg, K-Means, KNN, …)
  - Train Classification method
  - Test Classification method
  - Show results of Classification
  - Forecasting
Figure 162: Regression Sequence Diagram
Figure 163: Machine Learning Sequence Diagram
Figure 164: Classification Sequence Diagram
The UIs for configuring and executing these analytics operations are as follows:

![Figure 165: Regression UI Mockup](image1)

![Figure 166: Machine Learning UI Mockup](image2)
5.2.3.2.4 Configure Datasources
This story deals with the preparation steps prior to executing any algorithm, i.e., the configuration of the datasources depending on the type of analytics to be executed.

The main steps / functionalities are as follows:
• Configure Stream Sourcing:
  • Connect to stream source

• Configure Historic Sourcing:
  • Connect to vf-OS Storage
  • ETL

• Configure Batch Sourcing:
  • Connect to batch source
  • ETL

Figure 169: Configure Datasources Sequence Diagram

5.2.3.2.5 Stream Analysis Modules
This feature provides the capability to manage the stream analysis modules. A module is an independent set of stream data sources, thresholds, and sentences. Each module has its own life cycle.

The main functionalities are:
• Create an analysis module
• Read an analysis module
• Get analyses modules
• Verify an analysis module
• Drop an Analysis module
• Activate an analysis module
Deactivate an analysis module

Figure 170: Stream Analysis Modules Sequence Diagrams (I)
Figure 171: Stream Analysis Modules Sequence Diagrams (II)

The UI for Stream analysis modules is as follows:
5.2.3.2.6 **Stream Data Subscription**

This feature provides the capability to manage the Stream Data Sources analysed into a module.

The main functionalities are:

- Get stream data
- Subscribe to stream data
- Update data subscription
- Drop data subscription
The UI for Stream Data Subscription is as follows:

Figure 173: Stream Data Subscription Sequence Diagrams

Figure 174: Stream Data Subscription UI Mockup
5.2.3.2.7 Stream Analysis Contents

This feature provides the capability to manage the Stream Analysis Contents used in a module. The content of a module is a set consisting of: thresholds, alert sentences, and filtering sentences.

The main functionalities are:

- **Thresholds:**
  - Create threshold
  - Get thresholds
  - Update threshold
  - Drop threshold

- **Sentences:**
  - Create sentence
  - Get sentences
  - Update sentence
  - Drop sentence
  - Activate sentence
  - Deactivate sentence

Figure 175: Stream Analysis Contents Thresholds Sequence Diagrams
Figure 176: Stream Analysis Contents Sentences Sequence Diagrams
The UI for Stream Analysis Contents Sentences is as follows:

![UI for Stream Analysis Contents Thresholds UI Mockup](image)

Figure 177: UI for Stream Analysis Contents Thresholds UI Mockup
The UI for Stream Analysis Contents Sentences is as follows:

![UI for Stream Analysis Contents Sentences](image)

Figure 178: UI for Stream Analysis Contents Sentences UI Mockup

**5.2.3.2.8 Stream Analysis Results**

This feature provides the capability to manage the Stream Analysis Results produced in a module. The results of a module can be filtering data and / or alerts.

The main functionalities are:

- **Alerts:**
  - Publish an alert
  - Persist an alert
  - Get alerts
  - Update alert status

- **Filtered Data:**
  - Publish a filtered data
  - Persist a filtered data
  - Get filtered data
- Update filtered data status

Figure 179: Stream Analysis Results Alerts Sequence Diagrams

Figure 180: Stream Analysis Results – Filtered Data Sequence Diagrams
The UI for Stream Analysis Results Alerts is as follows:

Figure 181: UI for Stream Analysis Results Alerts UI Mockup
The UI for Stream Analysis Results Alerts is as follows:

![Figure 182: UI for Stream Analysis Results – Filtered Data UI Mockup](image)

5.2.3.3 Interaction description

From the previous description of the functionality covered by the Data Analytics component, a deeper level of detail regarding the main modules of the component and the interaction between those modules and other vf-OS components emerges. Whilst next Figure 183 shows the Architecture diagram, as presented in D2.1, the accompanying text focuses on the interactions and data exchange between the Data Analytics and other vf-OS components.
The main interactions of Data Analytics modules with other components are:

- **Data Sourcing**: it is the module in charge of offering the functionality of sourcing data from the different devices, other vf-OS components or software applications. This
module interacts with the different sources of information, meaning the Pub/Sub and Messaging (via the Stream API), the Storage (via the Historic API), the Data Warehouse (internal connection), and the Transformation Service (via the Transform API). The main information flows are:

- It receives datasets (derived from streamData) feeding from the vApps/software applications/devices (interaction with Messaging and Pub/Sub components)
- It receives datasets (derived from histData) feeding from the Historic Data repository (interaction with Storage component). This data is provided by sensors/devices/software applications that want to have some data stored for further analysis
- It sends source data needed for carrying out the ETL activities receiving the transformed data (interaction with Transformation Service)

- Query Manager: it is the module in charge of managing the Analytics’ queries and the Graph’s queries. It interacts directly with the main Analytics UI and also the Query Builder UI. It stores these queries in the vf-OS Storage for further re-use. The main information flows exchanged with external components are:
  - It sends, via the Query Builder UI, the builtQuery to be persisted for a later re-use (interaction with the Storage component)
  - It receives the query for execution (interaction with the Storage component)
  - It sends the reportingData after executing an analytics query (interaction with the Storage component)
  - It sends, via the Query Builder UI, inData for analysing its semantic relevance receiving the analysedData (interaction with the Mapping component)
  - It sends, via the Analytics UI, the analytics to be deployed as libraries for later re-use (interaction with SDK and Studio components)
  - It receives from the Analytics Engine, the OLAP Engine, and the Data Extraction modules the analysedData, processedDataSets, and extractedData, respectively, after they have been processed
5.3 I/O Toolkit

5.3.1 Enablers Framework

The Enablers Framework component provides a solution for integration of different enablers into a single service based component to provide uniform access to functionalities provided by enablers. The enablers specifically FIWARE enablers and Manufacturing enablers expose heterogeneous service interfaces posting the need to understand and implement diverse functionalities by the vApp developer to access their functionalities. Enablers’ framework intends to solve these issues by providing vApp developers with a single uniform access interface for accessing functionalities of the enablers. In the scope of the Enablers’ Framework component, the vApp developers are also provided with a client library that they can use in their implementation for easy
connection to the Enabler’s Framework and eventually to the enablers. Additionally, this component also exposes enablers to clients in a controlled manner without exposing the actual endpoint of the services and provides scalability through distributed synchronisation, and services grouping among multiple instances of enablers to handle request fluxes.

5.3.1.1 Behaviour and Functionality

Enablers Framework component provides a set of functionalities that could be grouped on the following features:

- **Registration and Configuration of Enablers**: All the enablers that are going to be integrated through the enablers’ framework must be registered in the enablers’ framework component. The main purpose of the registration process is to provide the details regarding the services that the enabler provides and the protocols that need to be followed for accessing those services. Additionally, the registration process also requires defining the details of messages that will be exchanged during method invocation. This feature will provide the IT admin user with the ability to configure different parameters that need to be provided for the correct functioning of the enabler. Note that due to the diverse nature of the enablers it might also be necessary for admin users to configure the enablers individually for the parameters not encapsulated by the framework. And also note that if the enablers are not freely available and have cost model associated with them then they first need to be registered in the Marketplace to provide their business model details. Then they can be further registered in the enabler’s framework with marketplace registration id. Enabler’s without marketplace registration proof will be served for free all the time else will be traced for consumption to generate necessary billing details.

- **Enabler’s lookup**: This feature is to provide a mechanism for finding enablers based on the functionalities that they will provide and other technical details like communication protocols supported, service interface model, QoS parameters etc. This lookup service will be internally used for utilising correct service proxy by service invocation request handling feature. And this feature can be used externally by the marketplace and/or SDK to help developers find the enabler that will meet their needs for developing vApps.

- **Service Invocation Request Handling**: This is one of the core features of this component and provides functional implementations for routing the service invocation requests to the right enabler. In doing so it is necessary to implement a proxy for services that will provide protocol bindings to those that are supported by the enablers thus providing the necessary protocol conversion. Additionally, this feature also encapsulates the functionality for load distribution over multiple instances of the same enabler to handle request influx.

- **Enablers’ Lifecycle Management**: this feature is intended to provide continuous integration of the enablers into the enablers’ framework especially for the vf-OS enablers who are expected to evolve over time. The enablers might expose new methods with newer releases and needs to be correctly integrated and served by the framework. Optionally, this feature also needs to encapsulate the mechanism to handle the case when the enablers can become obsolete and can impact the working of vApps.

- **Enablers’ Performance Monitoring**: This feature is dedicated to monitoring the performance and tracing the errors during run-time. This feature not only monitors its own performance but also collects errors and performance metrics from the enablers.
that are integrated to provide aggregated performance details. Sometimes, these metrics can also play an important part for the developer in making the choice for one enabler over another.

Follows is a story map where the main features, epics and user stories for the Enablers Framework components have been identified (see Figure 185).

**VF-OS ENABLERS’ FRAMEWORK**

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EFUS001. CRUD framework’s configurations</strong></td>
<td>Description</td>
</tr>
<tr>
<td>Who: vf-OS IT manager</td>
<td></td>
</tr>
<tr>
<td>What: Perform creation, viewing, updating and deleting configuration parameters that are applicable for enablers framework component</td>
<td></td>
</tr>
<tr>
<td>Why: so that admin users can manage the configuration details</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Make sure that all the admin users can configure the framework easily</td>
</tr>
<tr>
<td><strong>EFUS002. CRUD enablers’ configurations</strong></td>
<td>Description</td>
</tr>
<tr>
<td>Who: vf-OS IT manager</td>
<td></td>
</tr>
<tr>
<td>What: Perform creation, viewing, updating and deleting configuration parameters for the enablers that are registered and served in/by the framework</td>
<td></td>
</tr>
<tr>
<td>Why: so that admin users can manage the configuration details</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Make sure that all the admin users can configure the enablers easily</td>
</tr>
<tr>
<td><strong>EFUS003.</strong></td>
<td>Description</td>
</tr>
</tbody>
</table>
| **Provide enablers details for registration** | **Who:** vf-OS IT Admin  
**What:** Registration of enabler into the framework  
**Why:** so that the admin user can register the enabler and provide different parameters that will be used for describing the enabler. Some mandatory parameters are service endpoint, communication protocol and service interface type. Additionally it can include metadata for further describing the enabler such as textual description, keywords, multitenancy support etc. which are useful for refining results of lookup service.  
**Acceptance Criteria**  
Enabler is successfully registered and can be found through EFUS009 |
| **EFUS004. Provide enablers services details** | **Description**  
**Who:** vf-OS IT Admin  
**What:** Provide details of the functionalities that are provided by the enabler through methods in service interface.  
**Why:** so that admin user can define the methods accessible through the service interface and associated messages schemas for method invocation. Optionally this can also include textual description of the method for refining results of lookup service.  
**Acceptance Criteria**  
Methods exposed by enabler are successfully registered and can also be found through EFUS009 |
| **EFUS005. Provide enablers business model details** | **Description**  
**Who:** vf-OS IT Admin  
**What:** Provide details regarding business model associated with enabler.  
**Why:** so that admin user can provide a business model for the enabler and will be used for keeping track of accessibility and billing.  
**Acceptance Criteria**  
Business model for enabler are persisted and can also be found through EFUS009 |
| **EFUS006. Provide QoS details of the enabler** | **Description**  
**Who:** vf-OS IT Admin  
**What:** Provide QoS details associated with the enabler.  
**Why:** so that admin user can provide QoS related details that will be offered by the enabler. This will also be used for refining results of lookup service.  
**Acceptance Criteria**  
QoS parameters for enabler are successfully stored and can also be found through EFUS009 |
| **EFUS007. Install enabler** | **Description**  
**Who:** vf-OS IT Admin  
**What:** Install required enabler  
**Why:** so that the functionalities provided by the enabler can be accessed through the enablers framework.  
**Acceptance Criteria**  
Make sure the installation was successful.  
The communication protocol is implemented  
All desirable methods should be defined on the framework |
| **EFUS008. Provide continuous integration of the enablers into the framework** | **Description**  
**Who:** vf-OS Enabler Framework  
**What:** Latest development in the enabler can be integrated into the enabler’s framework through continuous integration.  
**Why:** so that the vAPPs can be provided with improved and/or additional functionalities through automated build, verification and deployment.  
**Acceptance Criteria**  
The continuous integration process doesn’t disrupt the functioning of the vAPPs |
| **EFUS009.** | **Description** |
### Find Enabler details from registry

**Who:** vf-OS Enablers Framework  
**What:** receive enabler’s methods and functionalities that each app can use  
**Why:** so that the details for invoking methods of enabler can be found.

#### Acceptance Criteria

The details of the enabler’s functionalities are provided and can be invoked by the enablers framework. This requires the implementation of EFUS003 and EFUS004.

#### Description

**Who:** vf-OS Enablers Framework  
**What:** receive enabler’s methods and functionalities that each app can use  
**Why:** so that the details for invoking methods of enabler can be found.

### EFUS010. Provide search over registered enablers

**Who:** vf-OS Enablers Framework  
**What:** search enablers based on users’ search criteria  
**Why:** so that the end user can be provided with enablers that will match the search criteria. In this use case the meta-data of the enablers is also used to get better search results.

#### Acceptance Criteria

The search over registered enablers must get all details of the enabler which includes available functionalities, QoS parameters, business models, access criteria etc. whatever is defined for the matching enablers. This requires the implementation of EFUS003 EFUS006.

#### Description

**Who:** vf-OS Enablers Framework  
**What:** search enablers based on users’ search criteria  
**Why:** so that the end user can be provided with enablers that will match the search criteria. In this use case the meta-data of the enablers is also used to get better search results.

### EFUS011. Access framework by it’s interface

**Who:** vf-OS vApp  
**What:** Request to use functionality provided by enablers through interface from framework  
**Why:** so that the vApps will have one uniform access interface to invoke the desirable enabler functionality.

#### Acceptance Criteria

The interface is well defined, supporting library (for JS and Java) for accessing enabler is provided and the messages schema are defined.

#### Description

**Who:** vf-OS vApp  
**What:** Request to use functionality provided by enablers through interface from framework  
**Why:** so that the vApps will have one uniform access interface to invoke the desirable enabler functionality.

### EFUS012. Invoke enabler’s functionality

**Who:** vf-OS Enablers Framework  
**What:** Enabler’s framework makes invocation of the functionality provided by selected enabler  
**Why:** so that the enabler framework can route the requests that it will receive at its interface (EFUS011), to the right enabler with necessary protocol and data adaptation.

#### Acceptance Criteria

The requested functionality is correctly invoked and a response is provided to the framework which will then be forwarded to the requesting vApp.

#### Description

**Who:** vf-OS Enablers Framework  
**What:** Enabler’s framework makes invocation of the functionality provided by selected enabler  
**Why:** so that the enabler framework can route the requests that it will receive at its interface (EFUS011), to the right enabler with necessary protocol and data adaptation.

### EFUS013. Synchronous response from enabler

**Who:** vf-OS Enablers Framework  
**What:** Enabler’s framework makes one enabler invocation and the framework waits for the response  
**Why:** To have a synchronous request/response.

#### Acceptance Criteria

The latency between request and response cycle is minimal

#### Description

**Who:** vf-OS Enablers Framework  
**What:** Enabler’s framework makes one enabler invocation and the framework waits for the response  
**Why:** To have a synchronous request/response.

### EFUS014 Handle request flux

**Who:** vf-OS Enablers Framework  
**What:** Management of request flux when there are a number of requests to the same enabler  
**Why:** So that many concurrent requests can be handled and provide acceptable response time for all the vApps invoking functionalities from enablers.

#### Acceptance Criteria

All the requests are handled with minimal latency and the different instances of same enablers are synchronised

#### Description

**Who:** vf-OS Enablers Framework  
**What:** Management of request flux when there are a number of requests to the same enabler  
**Why:** So that many concurrent requests can be handled and provide acceptable response time for all the vApps invoking functionalities from enablers.

### EFUS015. Asynchronous

**Who:** vf-OS Enablers Framework
response from enabler | What: enablers framework also serves asynchronous methods exposed by the enablers
Why: To have an asynchronous request/response functionalities between vApps and Enablers.

Acceptance Criteria
The asynchronous request response completes the cycle and is not clogged in between other requests.

EFUS016. Track and log message flows and related performance parameters | Description
Who: vf-OS Enablers framework
What: Track and log the performances of itself and also collect them from enablers served by the framework.
Why: For monitoring the working of the components and provide traceability of the errors for troubleshooting purposes. Additionally this functionality will also provide mechanisms for detection of errors and generate events that will be useful for detecting abnormalities.

Acceptance Criteria
All the invocation of the functionalities through the framework are logged and errors are correctly reported.

EFUS017. Provide Performance Details for a specific enabler | Description
Who: vf-OS System Dashboard
What: Provide performance details for the selected enabler
Why: To analyze performance and errors for each of the enablers for diagnostic purpose.

Acceptance Criteria
The enablers keep track of their run-time performances and errors and provide them to the enablers framework which can filter and aggregate based on the requested details.

EFUS018. Provide periodic performance report | Description
Who: vf-OS System Dashboard
What: Provide periodic performance report to the system dashboard.
Why: This will enable system admin to monitor the performance and health of the enabler’s framework and enablers that are served through the framework. The admin user can use this information to make plans for optimisation and or scaling the component in case of lower performance.

Acceptance Criteria
Performance reports are sent to the system dashboard on periodic basis as defined by the system dashboard admin.

5.3.1.2 UI mockups and Sequence Diagrams

5.3.1.2.1 Manage Configurations
This feature provides the capability to manage configurations for the enablers framework component and enablers that are integrated into the framework.

The main steps/functionality are:

- Perform CRUD operations on configurations of the framework
- Perform CRUD operations on configurations of the enablers integrated into the framework

The associated sequence diagram is as shown in Figure 186
Figure 186 Sequence Diagram for Managing Configurations for Enablers Framework Component and Enablers Integrated into the Framework
The associated UI mockups for managing configurations are as follows:

Figure 187: List, View Details, Edit and Delete Configurations for Framework UI Mockup

Figure 188: List, View Details, Edit and Delete Configurations for enablers UI Mockup
Figure 189 Add configurations for framework UI Mockup

Figure 190: Add configurations for enabler UI Mockup
5.3.1.2.2 Manage Registrations
This feature provides the capability to register the enablers that will be integrated into the enablers framework component and can be accessed through the interface provided by the framework.

The main steps/functionality are:

- Providing details for defining the enabler that will be integrated in the framework
- Providing business model associated with the enablers

The associated sequence diagram is as shown in Figure 191

![Sequence Diagram for Managing Registration of Enablers into the Enablers Framework](image)

Figure 191 Sequence Diagram for Managing Registration of Enablers into the Enablers Framework

The associated UI mockups for registration of enablers are:
Figure 192 List, View Details, Edit Descriptions and Delete Enablers Registered in the Enablers framework UI Mockup

Figure 193 Registration of new Enabler in the Framework UI Mockup
5.3.1.2.3 Enablers’ Lifecycle Management
This feature provides the capability to install and deploy the latest version of the enablers that are integrated into the framework.

The main steps/functionality are:
- Installation of enabler in the platform by fetching the binaries provided by the enabler.
- Perform continuous integration of the latest builds of the enablers as they evolve over time.

The associated sequence diagram is as shown in Figure 194.

5.3.1.2.4 Enablers’ Lookup
This feature provides a way for finding the enablers that are registered in the framework and associated details for using the enabler. The main step/functionality is:
- Look into the enablers registry and find the one that satisfies the search criteria.

The associated sequence diagram is as shown in Figure 195.
5.3.1.2.5 Service Invocation Request Handling
This feature provides the functional implementations for routing the service invocation requests to the right enabler based on the request made by the vApp.

The main step/functionality is:

- Collecting the request from the vAPP and finding the details of the enabler and its method that the request needs to be directed to.
- Instantiation of suitable service proxy to access the method of the enabler.
- Support both synchronous and asynchronous types of request-response patterns
- Provide load balancing and synchronisation between different instances of enablers to handle requests influx.

The associated sequence diagram is as shown in Figure 196.
5.3.1.2.6 Performance Monitoring

This feature provides the capability for monitoring the performance and tracing of errors during the run-time execution of the enablers framework and enablers. The main steps/functionality are:

- Collect the errors that will occur between the processes of requests.
- The error logs are to proactively find potential problems and publish notifications via dashboard for system admin.
- Keep track of the performance metrics such as down time, response time etc to find correlations between performance and framework service. The performance matrices are provided for system admin via system dashboard for performance optimisation and scalability.

The associated sequence diagram is as shown in Figure 197.
5.3.1.3 Interaction Description

Based on the description of the functionality covered by the enablers framework component we can observe a number of interactions that the component will have with other vf-OS components. Presented in this section is a detailed representation of the interactions with other vf-OS components and also some internal interactions between sub-components of the enablers framework component. The following figure shows the flow of information between the internal sub-components and other vf-OS components.
In order to clarify the interactions between components the main interactions of the messaging component with other components are as explained below:

- Configurations and Registrations management: This provides necessary interaction with the vf-OS storage component and is used for storing configuration details of the framework and associated enablers. Additionally, it also involves interactions for the management of registration of enablers. The main information flows are:
  - Put/Get configuration details into/from the storage
  - Put/Get registered enablers in the framework and the details provided during registration
• Request Handler: This provides necessary interaction with the vAPPs and enablers (FI-WARE Generic Enablers, Manufacturing Enablers and vf-OS Enablers). vAPPS interact to access the functionalities provide by the enablers via this subcomponent. And in turn, this component translates the requests from the vApps to the respective enabler with necessary message and protocol transformation to invoke the functionalities in the native standard of enablers. In this process, the request handler utilises service proxy factory sub-component to establish the correct service invocation mechanism towards the enabler. The main information flows are:

  • vApp sends request to invoke method of the enabler with necessary input parameters
  • RequestHandler finds the details of the enabler and specified method from the enablers registry
  • Enablers definition is provided to the service proxy factory that will provide the instance of service proxy that will establish a connection with the enabler and invoke the stated method
  • When the enabler finishes computing it returns back the result of the method invocation which is forwarded as a response to vApp

• Lookup Services: This provides necessary interactions towards marketplace to facilitate enablers search based on the needs of the end-user. The main information flows exchanged with external components are:

  • End user provides necessary search criteria
  • Search is performed in the enablers registry and is responded with a list of enablers that satisfy the search criteria

• Monitoring: This provides necessary interaction with the system dashboard component to publish performance, errors and events during execution. The main information flows exchanged with system dashboard are:

  • Enablers Framework component can publish errors and event logs with optional criticality tag
  • Enablers Framework can publish periodic performance metrics collected during the execution of the messaging component
5.3.2 Drivers

The Drivers component provides a collection of reference implementations to collect data from, and send commands to, industrial automation devices, such as PLCs, smart sensors, RFID readers, etc., directly or through protocol communication gateways. Drivers are based on a common architecture which is described in this section.

5.3.2.1 Behaviour and Functionality

Drivers component provides a set of functionalities that could be group on the following features:

- **Drivers management**: where the installed drivers are shown. Drivers are software classes that support the communication of specific physical devices through specific open or close protocols.
- **Device Management**: where physical devices are registered into a specific vf-OS Platform as sources of information and/or receivers of actuation commands to
interact with the physical world. The registration of devices implies the specification of parameters and the usage of installed drivers.

- **Devices’ data reading**: where a range of functionality is provided regarding the different mechanisms to read devices and their sensors. Drivers can implement synchronous and asynchronous read methods. Optionally, drivers can support short-term historic data read methods and edge computing.

- **Devices controlling**: A set of devices will provide actuators (components for moving or controlling them). In this case, and whenever the driver’s associated driver supports its control, vApps will be able to act on physical devices through the drivers component.

Follows is a story map where the main features, epics and user stories for the drivers components have been identified (see Figure 200).

![VF-OS DRIVERS](Image)

Figure 200: Drivers Story Map

The textual description of each user story is as follows:
### Subtask: DRUS001
**Receive asynchronous data from device**

**Description**

Who: vf-OS Drivers  
What: receive data pushed by a sensor based on an event configured on the sensor  
Why: so that event data can be pushed to subscribed vApps

**Acceptance Criteria**

Make sure that the event is generated according to the configuration parameters provided by the driver

### Subtask: DRUS002
**Push data on pub/sub**

**Description**

Who: vf-OS Drivers  
What: push data to the pub/sub component  
Why: so that data subscribers receive the data asynchronously

**Acceptance Criteria**

Make sure that the data is pushed according to the pub/sub configuration provided

### Subtask: DRUS003
**Store data in short-term historic**

**Description**

Who: vApps  
What: can query short-term historic data in driver local storage  
Why: so that they can retrieve data when in diverse scenarios such as when disconnection with the cloud platform is produced

**Acceptance Criteria**

Make sure that at least the last 100 measurements are stored in short-term historic data

### Subtask: DRUS004
**Polling process for reading sensors**

**Description**

Who: vf-OS Drivers  
What: periodically read data from sensors through devices that do not have a push-based mechanism  
Why: so that vApps can configure sensor events

**Acceptance Criteria**

Make sure the sensor configuration stores the time interval between sensor readings  
Make sure that a background process retrieves applying the time interval seconds from sensors/devices  
Make sure US006 is already developed

### Subtask: DRUS005
**Edge computing**

**Description**

Who: vf-OS Drivers  
What: apply mathematical or statistical processing to sensor data readings  
Why: so that only relevant information is sent to vApps / higher level data storage management

**Acceptance Criteria**

Make sure that arithmetic computation and rules can be applied before storing and pushing data asynchronously to pub/sub subscribed vApps or before storing it on local short-term storage

### Subtask: DRUS006
**Query devices / sensors**

**Description**

Who: vf-OS Drivers  
What: will query devices/sensors using specific drivers and will retrieve their data in a standardised way  
Why: so that data could be processed and sent to a vApp

**Acceptance Criteria**

Make sure that a range of specific physical devices with their sensors can be queried

### Subtask: DRUS007
**Offer API for**

**Description**
| **DRUS008. Read short-term data filtering by range of dates** | **Who**: vf-OS Drivers  
**What**: will provide methods to read short-term historic data through an API with a Messaging component  
**Why**: so that vApps can read on demand and historic short-term data  

**Acceptance Criteria**  
Make sure that the API is released and accorded with Messaging component  

**Description**  
Who: vf-OS Drivers  
What: will be able to send command on specific private protocols to devices  
Why: so that vApps can be enrich with functionality to act on devices according to certain criteria  

**Acceptance Criteria**  
Devices will support commands, will be configured as command receivers, drivers implementation will support actions on specific protocols and actions are carried out when sent by vf-OS driver module |
| **DRUS009. Send command to device** | **Who**: vf-OS Drivers  
**What**: will provide methods to interact with Messaging component  
**Why**: so that vApps can read synchronous data on demand and historic short-term data  

**Acceptance Criteria**  
Make sure that the API is released and accorded with Messaging component  

**Description**  
Who: vf-OS Drivers  
What: will provide an API to interact with Messaging component  
Why: so that vApps can read synchronous data on demand and historic short-term data |
| **DRUS010. Register device on vf-OS** | **Who**: vf-OS manufacturing and logistics provider  
**What**: will set-up an existing physical device as a source of information  
**Why**: so that vApps could receive, query and send commands to devices and their sensors  

**Acceptance Criteria**  
Only IT managers/administrators could register devices on the platform for a specific company  
Devices will be configured according to a driver and will make use of the driver’s specific development for reading devices/sensors and sending commands |
| **DRUS011. Register asynchronous sensors of device** | **Who**: vf-OS manufacturing and logistics provider  
**What**: will register sensors of a given device as asynchronous capable  
**Why**: so that vApps could receive data from devices under certain events and time intervals  

**Acceptance Criteria**  
Only IT managers/administrators could register devices on the platform for a specific company  
Devices will be configured according to a driver and will make use of the driver’s specific development for reading devices/sensors and sending commands |
| **DRUS012. Register synchronous sensors of device** | **Who**: vf-OS manufacturing and logistics provider  
**What**: will register sensors of a given device as synchronous capable  
**Why**: so that vApps could receive data from devices when requesting proactively  

**Acceptance Criteria**  
Only IT managers/administrators could register devices on the platform for a specific company  
Devices will be configured according to a driver and will make use of the driver’s specific development for reading devices/sensors and sending commands |
| **DRUS013. Configure device for short term historic database** | **Who**: vf-OS manufacturing and logistics provider  
**What**: will configure a sensor as a historic short-term sensor  
**Why**: so that vApps could query the data of a given sensor between two dates and the data push by the sensors is not lost when arise connectivity problems  

**Acceptance Criteria**  
Only IT managers/administrators could register devices on the platform for a specific company  
Devices will be configured according to a driver and will make use of the driver’s specific development for reading devices/sensors and sending commands |
<table>
<thead>
<tr>
<th>DRUS014. Configure computation of sensor data</th>
<th>with the vf-OS platform.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Only IT managers/administrators could switch on historic short-term storage on devices/sensors. The data will be stored on a short-term historic database handled by the vf-OS driver.</td>
</tr>
</tbody>
</table>
| **Description** | Who: vf-OS manufacturing and logistics provider  
What: will configure a data computation applied to the data captured by a device's sensor  
Why: so that vApps could obtain processed data with low level computation that could address scenarios such as the correction of a measuring error done by a sensor, or a homogenisation of units, etc. |
| **Acceptance Criteria** | Only IT managers/administrators will introduce the formula applied by the computation in set-up time  
The data will be run against a given computation and return to the vf-OS driver module so that he can store or propagate the data to other components until the vApp. |

<table>
<thead>
<tr>
<th>DRUS015. Configure device as command receiver</th>
<th>Description</th>
<th></th>
</tr>
</thead>
</table>
| **Who**: vf-OS manufacturing and logistics provider  
**What**: will configure a device as a command receiver  
**Why**: so that vApps could send commands according to their needs. | **Who**: vf-OS manufacturing and logistics provider  
**What**: will configure a device as a command receiver  
**Why**: so that vApps could send commands according to their needs. | **Description** |
| **Acceptance Criteria** | Only IT managers/administrators could switch on the command receiver according to the capabilities of the driver. The implementation of the driver will offer the list of commands that could be asked for by the driver module under the vApp query. | **Acceptance Criteria** |

<table>
<thead>
<tr>
<th>DRUS016. List existing devices already configured</th>
<th>Description</th>
<th></th>
</tr>
</thead>
</table>
| **Who**: vf-OS manufacturing and logistics provider  
**What**: will list existing registered and configured devices  
**Why**: so that they can acknowledge the sources of information (devices) that could provide data to specific vApps | **Who**: vf-OS manufacturing and logistics provider  
**What**: will list existing registered and configured devices  
**Why**: so that they can acknowledge the sources of information (devices) that could provide data to specific vApps | **Description** |
| **Acceptance Criteria** | Configuration of devices should be carried about beforehand  
Devices will be shown along with some attributes on a table | **Acceptance Criteria** |

<table>
<thead>
<tr>
<th>DRUS017. Filter list of devices according to name and type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
</table>
| **Who**: vf-OS manufacturing and logistics provider  
**What**: will be able to filter the configured list of devices by name and type  
**Why**: so that they detect what are the sources of information configured | **Who**: vf-OS manufacturing and logistics provider  
**What**: will be able to filter the configured list of devices by name and type  
**Why**: so that they detect what are the sources of information configured | **Description** |
| **Acceptance Criteria** | A search box will allow the filtering of the list of devices by name and type | **Acceptance Criteria** |

<table>
<thead>
<tr>
<th>DRUS018. Sort list of devices by columns</th>
<th>Description</th>
<th></th>
</tr>
</thead>
</table>
| **Who**: vf-OS manufacturing and logistics provider  
**What**: will be able to sort the configured list of devices by any of the columns  
**Why**: so that a better understanding of configured devices could be achieved | **Who**: vf-OS manufacturing and logistics provider  
**What**: will be able to sort the configured list of devices by any of the columns  
**Why**: so that a better understanding of configured devices could be achieved | **Description** |
| **Acceptance Criteria** | Arrows beside each column will allow sorting of the table of devices in ascending or descending order | **Acceptance Criteria** |

<table>
<thead>
<tr>
<th>DRUS019. Check device status</th>
<th>Description</th>
<th></th>
</tr>
</thead>
</table>
| **Who**: vf-OS manufacturing and logistics provider  
**What**: will be able to view the current status of the devices  
**Why**: so that problems with devices could be identified and solved | **Who**: vf-OS manufacturing and logistics provider  
**What**: will be able to view the current status of the devices  
**Why**: so that problems with devices could be identified and solved | **Description** |
<p>| <strong>Acceptance Criteria</strong> | | <strong>Acceptance Criteria</strong> |</p>
<table>
<thead>
<tr>
<th>DRUS020. Check sensor status</th>
<th>Each device on a list of devices should add a new column showing the status of the device</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vf-OS manufacturing and logistics provider</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will be able view the current status of each device's sensor</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> so that problems with sensors could be identified and solved</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Each device could be queried about its sensors and a list of those with a new column showing the status of the sensor should be provided</td>
<td></td>
</tr>
<tr>
<td>DRUS021. List existing drivers of a specific vf-OS instance installation</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vf-OS manufacturing and logistics provider</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will list existing installed drivers (installed from the vf-OS store module) getting the detail on version, etc</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> so that they can acknowledge which devices will be able to be configured according to driver/version/protocol</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Installation of drivers will be carried out by vf-OS store Drivers will be tagged with version and protocol List of drivers will be in a table and show name, protocol, version</td>
<td></td>
</tr>
<tr>
<td>DRUS022. Filter list of drivers by Name</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vf-OS manufacturing and logistics provider</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will be able to filter the existing installed drivers by name</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> so that it can be detect if a specific driver is installed properly</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Installation of drivers will be carried out by vf-OS store Drivers will be tagged with version and protocol A search box will allow the filtering of the list of drivers</td>
<td></td>
</tr>
<tr>
<td>DRUS023. Filter sensor readings according to their value</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vf-OS Drivers</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> filter readings according to their value (data value or timestamp)</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> so that I can control the value range and the minimum interval between sensor data publication events</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>A minimum value for sensor data publish messages can be defined A maximum value for sensor data publish messages can be defined A minimum time interval between data publish messages can be defined</td>
<td></td>
</tr>
<tr>
<td>DRUS024. Log messages from drivers component</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vf-OS Drivers</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will be able send logs on information, warning and errors to vf-OS platform</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> so that vf-OS platform can provide a unified log dashboard to vf-OS users</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Drivers should have logs with messages to be shown At least three levels of logs agreed with platform (information, warning error) A list of logs shown according to level of logs</td>
<td></td>
</tr>
<tr>
<td>DRUS025. Manifest reading</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vf-OS Drivers</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> will return a manifest file from an installed driver</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> so that vf-OS enablers can query drivers on their usage</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Existing files with drivers must be installed previously by the vf-OS Marketplace Manifest interface retrieves a json file associated to a given driver required in a request</td>
<td></td>
</tr>
</tbody>
</table>
5.3.2.2 UI Mockups and Sequence Diagrams

The following sub-sections describe the UI mockups and sequence diagrams describing the interaction.

5.3.2.2.1 List Existing Drivers

This feature provides the capability to list driver types that have been installed on vf-OS. The main steps / functionality are:

- List existing drivers of a specific vf-OS instance installation
- Filter list of drivers by Name

![Sequence Diagram](image)

Figure 201: List Existing Drivers Sequence Diagrams

5.3.2.2.2 Logs Listing

This feature provides the capability to log messages of the drivers component and list them to the vf-OS platform. Logging will be according to categories established by the platform (at least three, information, warning, and error).

The main steps / functionality are:

- Log messages from drivers component
- Send log messages to the vf-OS platform

![Sequence Diagram](image)
5.3.2.2.3 Manifest Reading
This feature provides the capability to the enablers to retrieve the manifest of drivers installed on the vf-OS platform so that enablers can query registered devices on a given vf-OS instance. The main steps / functionality are:

- Retrieve a manifest file
- Send it to the Enablers framework

5.3.2.2.4 Add/Configure New Device
This feature provides the capability to add an existing device on a shopfloor so that vApps can interact with compatible devices.

Registering a device implies indicating the type of driver/protocol the device is using (that should have been previously installed), register the set of sensors a device uses to capture data, define for each sensor the supported modes (synchronous, asynchronous), the
computation rules, the need to store short-term historic data, and/or if the device can be controlled.

The main steps / functionality are:

- Register device on vf-OS
- Register asynchronous sensors of device
- Register synchronous sensors of device
- Configure device for short term historic database
- Configure computation of sensor data
- Configure device as command receiver
Figure 204: Add/Configure New Device Sequence Diagram
The UIs for adding and configuring new devices is as follows:

Figure 205: Add/Configure New Device UI Mockup

5.3.2.2.5 List Existing Devices
The feature provides the capability to list devices that have been registered on vf-OS and filter the list according to their Name.

The main steps/functionality are:
- List existing devices already configured
- Filter list of devices according to name and type
- Sort list of devices by columns
Figure 206: List Existing Devices Sequence Diagram
The UI for listing existing devices is as follows:

![Figure 207: List Existing Devices UI Mockup](image)

5.3.2.2.6 Check Status

The feature provides the capability to check the status of a device (ok or fail) and its sensors (ok or fail).

The main steps / functionality are:

- Check device status
- Check sensor status
Figure 208: Check Status Sequence Diagram

The UIs for checking the status of devices and sensors are as follows:

Figure 209: Check Status UI Mockup

5.3.2.2.7 Receive Asynchronous Data from Device

This feature allows the vf-OS Drivers to get data from a physical device that pushes info according to a timer or event on the shopfloor.

For the range of devices that does not provide this proactive functionality, an own timer for reading the data is provided.

The main steps/functionality are:

- Receive asynchronous data from device
- Push data on pub/sub
- Store data in short-term historic
- Polling process for reading sensors
5.3.2.2.8 Read Synchronous Data from Device

This feature allows the vf-OS Drivers to receive queries under demand from any other component through the messaging component.

The main steps/functionalities are:

- Query devices / sensors
- Offer API for reading low latency stream data
5.3.2.2.9 Read Short Term Historic Data
This feature allows the vf-OS Drivers to read the data that has been stored on a database to be retrieve by range of dates by any other component through the messaging component.

The main steps/functionalities are:
- Read short-term data filtering by range of dates

5.3.2.2.10 Send Command to Device
This feature allows any vApp to send command to devices on a shopfloor, to do that commandSender will use proprietary developments of specific devices.

The main steps/functionality are:
- Send command to device
5.3.2.3 Interaction Description

From the previous description of the functionality covered by the drivers module, a deeper level of detail regarding the main components of the component and the interaction between those driver’s subcomponents and other vf-OS components emerges. Following, there is a picture showing the flow of information exchange between the drivers subcomponents and vf-OS components.
In order to clarify the interactions between components, the information exchanged between drivers subcomponents and other components has been detailed (see figure below). The emphasis is in the messages exchanged between interfaces (purple boxes) and external components (blue boxes):

- **Messages to/from PubSub, Messaging, vf-OS Assets**: the vf-OS Drivers component provides three different ways to provide the information regarding devices and sensor and to get commands from them. The information exchange is:
  - Drivers component returns sensorData when reading synchronous data from sensors
  - Drivers component returns a list of sensorMeasures when returning a list of short-term measures by range
  - Drivers component receives cmdDevice that is a command for a device

- **Messages to/from the vf-OS Platform**: providing information to the platform as a central point to manage the vf-OS instance. The information exchange is:
  - Drivers component sends logging data generated by its execution so that the vf-OS platform can show a unique logging mechanism to detect errors and evaluate the health of the platform
  - Drivers component sends metadata such as drivers installed, version and communication protocol to the vf-OS platform so that the unique configuration dashboard can show it to administrators

- **Messages to/from the Enablers framework**: providing the manifest files that are used by vf-OS Assets to load a driver automatically. The information exchange is:
  - Drivers component sends the manifest driver after a query from the enablers framework component

- **Messages to/from physical devices or gateway**: those are not vf-OS components but external components. The information exchange is device-proprietary formatted. The information exchange is:
  - Drivers component receives proprietaryDeviceDataObject with the data in a specific format according to the specific device
  - Drivers component receives statusData with in a specific format according to the specific device
  - Drivers component receives and sends configData with the configuration in a specific format according to the specific device
  - Drivers component sends cmdDevice with a command instruction in a specific format according to the specific device
5.3.3 APIs

In the perspective of an open enterprise application, APIs allow to leverage and incorporate functionalities in and from other applications (ERPs, CRM, etc.) as needed. APIs should be defined at the right level of complexity (ie granularity, security, etc.) and with enough flexibility to support the evolution during application lifecycle management. APIs support the communication at several levels. In the context of vf-OS, APIs are primarily for access external software applications and other components (eg vApps-to-backend, frontend, or services). Although they could also potentially allow vApp-to-vApp interoperability.

5.3.3.1 Behaviour and Functionality

API component provides a set of functionalities that could be grouped into the following modules:

- **API management**: Where a range of functionality is provided to add and configure an API, list existing APIs, to check feedbacks/ratings of other users and subscribe to APIs
- **API Lifecycle management**: This module provides functionalities to publish API on the API Gateway and check API statistics
- **API User management**: This module manages users by providing functionalities to create users, create and assign role (developer, provider, admin) to user
Following, there is a story map where the main features and user stories for the APIs components have been identified (see Figure 217).

**VF-OS APIs**

- Manage APIs
- List existing APIs
- Subscription
- Check feedbacks and ratings
- Publish APIs
- Monitoring Services
- Add/Remove User
- Manage Users
- Add/Configure API

**Release R1 (M24)**
- APUS001. Create API
- APUS006. Register Application
- APUS010. Publish API
- APUS007. Subscribe to API
- APUS008. Generate APIs access tokens

**Release R2 (M30)**
- APUS004. List APIs
- APUS009. Consult feedbacks and rating
- APUS012. Create user
- APUS013. Assign role

**Release R3 (M33)**
- APUS003. Create new API version
- APUS011. API Monitoring

**Figure 217: APIs Story Map**

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
</table>
| APUS001 Create API | **Description**  
**Who:** API Developer  
**What:** Will set up the API by setting all technical information and documentation needed  
**Why:** To publish the API on the API Gateway  
**Acceptance Criteria**  
Only API Developer could create APIs on the platform  
Set the endpoint Type (REST/SOAP), Name, Version, Visibility, Description, Tags, URL Pattern  
Set Production Endpoint of the API  
Set API parameters  
Set number of allowed requests  |
| APUS002 Update API Specifications | **Description**  
**Who:** API Developer  
**What:** Will update API specification |
| **APUS003** Create new API version | **Description**
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who</strong>: API Developer</td>
<td>Will create new version of an API</td>
</tr>
<tr>
<td><strong>What</strong>: Will create new version of an API</td>
<td>In order to update, fix bugs or offer a better service</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Only API Developer could update APIs specifications</td>
</tr>
<tr>
<td></td>
<td>Old versions become deprecated</td>
</tr>
<tr>
<td></td>
<td>The latest version of the API is published to the API Gateway</td>
</tr>
</tbody>
</table>

| **APUS004** List APIs | **Description**
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who</strong>: API users</td>
<td>Will list APIs</td>
</tr>
<tr>
<td><strong>What</strong>: Will list APIs</td>
<td>To get information of existing APIs</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>The user will search by Name, Tag or provider</td>
</tr>
</tbody>
</table>

| **APUS005** Check API Documentation | **Description**
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who</strong>: API users</td>
<td>Will check the documentation of an API</td>
</tr>
<tr>
<td><strong>What</strong>: Will check the documentation of an API</td>
<td>In order to get information about technical or functional</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>API information is available</td>
</tr>
</tbody>
</table>

| **APUS006** Register Application | **Description**
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who</strong>: API users</td>
<td>Subscribe to APIs through an application</td>
</tr>
<tr>
<td><strong>What</strong>: Subscribe to APIs through an application</td>
<td>To invoke the APIs</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Any kind of user could register Applications (Admin, developer, Provider)</td>
</tr>
<tr>
<td></td>
<td>Set Name and Description of the application</td>
</tr>
</tbody>
</table>

| **APUS007** Subscribe to API | **Description**
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who</strong>: API users</td>
<td>Subscribe to APIs through an application</td>
</tr>
<tr>
<td><strong>What</strong>: Subscribe to APIs through an application</td>
<td>To use the wanted APIs</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Any kind of user could subscribe to APIs (Admin, developer, Provider)</td>
</tr>
</tbody>
</table>

| **APUS008** Generate APIs access tokens | **Description**
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who</strong>: API users</td>
<td>Generate APIs access tokens</td>
</tr>
<tr>
<td><strong>What</strong>: Generate APIs access tokens</td>
<td>To ensure better security (e.g., prevent DoS attacks)</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td>Any kind of user could generate access tokens to APIs (Admin, developer, Provider)</td>
</tr>
<tr>
<td></td>
<td>Select application for which you want to generate an access token</td>
</tr>
</tbody>
</table>

| **APUS009** Consult feedbacks and rating | **Description**
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who</strong>: API Developer</td>
<td>Consult feedbacks and rating provided by API users</td>
</tr>
<tr>
<td><strong>What</strong>: Consult feedbacks and rating provided by API users</td>
<td>To get feedback about its usability, and offer a better service by updating or developing a new version</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
</tbody>
</table>
5.3.3.2 UI mockups and Sequence Diagrams

The following sub-sections describe the UI mockups and sequence diagrams describing the interaction.

5.3.3.2.1 Add/Configure API

The feature provides a capability to add and configure an API. Registering an API implies indicating the type of the API (REST, SOAP) and all information needed on the design phase, implementation phase and Management phase: Name of the API, URI context path, version, visibility (public, private), description of the API, tags, URL Patterns and their related parameters, end point type (http, https,…), production endpoint, endpoint authentication type (public, private), authentication credentials (username, password), number of request authorized, and other information about the business owner of the API (name, mail, technical owner…).
Figure 218 Add/Configure API Sequence Diagram
The UI to add and configure an API is as follows:

![Add/Configure New API UI Mockup (1)](image)

Figure 219 Add/Configure New API UI Mockup (1)
5.3.3.2.2 List existing APIs

The feature provides a capability to list all existing APIs that have been installed on vf-OS. The main functionalities are:

- List existing APIs
- Filter list of APIs by Name, Tag or provider.
- Select an API from the list to show related specifications
Figure 221 List Existing APIs Sequence Diagram
5.3.3.2.3 **Subscription**
The feature provides a capability to:

- Register an application which is a logical collection of APIs. An application is primarily used to decouple the consumer from the APIs.
- Subscribe to APIs through the application which enables you to invoke APIs
- Generate access token to authenticate API users and applications when invoking APIs to ensure better security.
The UIs for registering an application, subscribe to APIs and generating access are as follows:
Figure 224 Subscription Mockup (1)
5.3.3.2.4 Check feedbacks and ratings
The feature provides a capability to check comments and feedbacks of an API that have been installed on vf-OS.
The UIs for checking feedbacks and ratings of an API is as follows:

Figure 226 List existing APIs Sequence Diagram

The UIs for checking feedbacks and ratings of an API is as follows:
5.3.3.2.5 Publish APIs
The feature provides a capability to publish an API on vf-OS. Only API Providers can change the visibility of an API by selecting one of these options: Created, Published, Deprecated, Blocked:

- Created: Not visible to users yet (by default option)
- Published: Available in the API Gateway
- Deprecated: The API is still deployed in the API Gateway but not visible to new users
- Blocked: Access to the API is temporarily blocked

Figure 227: Publish APIs Sequence Diagram

The UIs for publishing an API is as follows:
5.3.3.2.6 Monitoring Services
The feature provides performance details and usage statistics on APIs. The provider could find some information like number of subscriptions for every version of the API, number of calls, API Latency Time, etc.

![Publish APIs Mockup](image)

Figure 228 Publish APIs Mockup

![Monitoring Services Sequence Diagram](image)

Figure 229 Monitoring Services Sequence Diagram

The UIs for getting access to monitor service is as follows:
5.3.3.2.7 Create/Assign Role

The feature provides to the administrator of the API platform functionalities to create, assign and manage roles. The default roles existing on the platform are "Developer", "Manager or Provider" and "subscriber".

A user with a “Developer” role is typically a person in a technical role who designs, implements and understands the technical specifications of the API (interfaces, documentation, versions etc.) and uses the API publisher to provision APIs into the API store but cannot manage their lifecycle.

A user with "Manager or Provider" role is a person in a managerial role and manage’s a set of APIs across the enterprise and controls the API lifecycle, subscriptions and monetisation aspects. The "Manager or Provider" is also interested in usage patterns for APIs and has access to all API statistics.

A “Subscriber” is a user or an application developer who searches the API store to discover APIs and use them. He reads the documentation, rates/comments on the APIs, subscribes to APIs, obtains access tokens and invokes the APIs.
The UIs for creating roles is as follows:

5.3.3.2.8 Add/Remove User

Users are consumers who interact with APIs. These users can be persons, devices or applications. Since these users interact with internal systems and access data, the need to define which user is allowed to do what is important.

The feature provides to the administrator functionalities to create users, manage and assign roles.
The UIs for creating users and assigning roles is as follows:

Figure 233 Add/Remove User Sequence Diagram

Figure 234 Add/Remove User Mockup
5.3.3.3 Interaction description

In order to clarify the interactions between components, the information exchanged between API subcomponents has been detailed (Figure 236). The main interactions of API component’s classes are:

- **API Manager**: It is in charge of managing the API Platform. The main information flows are:
  - It receives API usage statistics from API Monitoring
  - It returns permissions and authorise access to users
  - It receives a list of APIs from API Registry
- **API Store Manager**: It is in charge of managing applications and Store (by publishing APIs on API Gateway). It receives generated access token from Key Manager to authenticate API users and applications.

- **API Gateway**: It is in charge of managing the API Gateway by deploying APIs and to be accessible to users and managing their lifecycle (deploy, deprecate, block, ...)

---

**Figure 236 API Component Classes and Information Exchanged**

---

D2.2: Functional Specifications & Mockups - Vs: 1.0 - Public
5.3.4 External Service Provision

The vf-OS External Service Provision component provides a standard approach to use external services within vf-OS vApps. It provides documentation about external APIs as well as access (wrapper and support) libraries for these APIs.

5.3.4.1 Behaviour and Functionality

The following main functionalities are offered by the vf-OS External Service Provision component:

- **Library Use**: Mechanisms for developers to download and use external service wrapper and support libraries from their respective repositories inside vApps, including the provision of documentation
- **Library Development**: Mechanisms for developers to add and delete wrapper and support libraries to/from their respective repositories, again including documentation
- **Library Function Execution**: Means for users of the vf-OS platform to use vApps which on their turn use of external services

External Service Provision works closely with the Marketplace for its repository use and management. The Developer Engagement Hub is the key integration point for the external service documentation. Finally, functions inside the libraries themselves can make use of the vf-OS Pub/Sub and Messaging components for communicating with external services, in as far as the platform policies allow for this.

An overview of activities, tasks and stories related to the External Service Provision component is shown in Figure 237.
The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESUS001 download ESP library from marketplace</td>
<td><strong>Description</strong>&lt;br&gt;Who: vApp Developer&lt;br&gt;What: download an ESP wrapper or support library from the marketplace&lt;br&gt;Why: to use the library inside a newly developed vApp</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;The library is downloaded from the marketplace and available to the developer for integration inside the vApp.</td>
</tr>
<tr>
<td>ESUS002 integrate ESP library in vApp</td>
<td><strong>Description</strong>&lt;br&gt;Who: vApp Developer&lt;br&gt;What: use an ESP wrapper or support library inside a vApp&lt;br&gt;Why: to make use of the external service within the vApp</td>
</tr>
<tr>
<td></td>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;vApp is able to use the external service via the integrated wrapper or support library.</td>
</tr>
<tr>
<td>ESUS003 read documentation</td>
<td><strong>Description</strong>&lt;br&gt;Who: vApp Developer&lt;br&gt;What: use the documentation provided in the ESP documentation repository&lt;br&gt;Why: to get more information about the way in which a certain wrapper or support library can be used</td>
</tr>
</tbody>
</table>
### Acceptance Criteria

The developer is able to find and read the documentation via the Developer Engagement Hub.

### ESUS004  
**execute ESP library functions**

**Description**

- **Who:** vApp user
- **What:** use a vApp which makes use of certain ESP wrapper or support libraries
- **Why:** to use the functionality provided by the vApp

**Acceptance Criteria**

vApp is able to make use of the external services whilst providing its functionality.

### ESUS005  
**add library to repository**

**Description**

- **Who:** Library Developer
- **What:** add newly developed ESP wrapper or support libraries to the respective ESP repository
- **Why:** to provide new support for external services

**Acceptance Criteria**

Newly developed ESP wrapper or support library is added to the respective ESP repository.

### ESUS006  
**delete library from repository**

**Description**

- **Who:** Library Developer
- **What:** delete existing ESP wrapper or support library from the respective ESP repository
- **Why:** to discontinue the use of the library by developers

**Acceptance Criteria**

Wrapper or support library is removed from the respective ESP repository.

### ESUS007  
**add documentation to ESP documentation repository**

**Description**

- **Who:** Library Developer
- **What:** add documentation about an ESP wrapper or support library to the ESP documentation repository
- **Why:** to provide information about the usage of the library to developers

**Acceptance Criteria**

The added documentation is accessible via the Developer Engagement Hub.

### ESUS008  
**delete documentation from ESP documentation repository**

**Description**

- **Who:** Library Developer
- **What:** delete documentation about an ESP wrapper or support library from the ESP documentation repository
- **Why:** to discontinue the use of the documentation about the library by developers

**Acceptance Criteria**

The documentation is removed from the ESP documentation repository and no longer accessible via the Developer Engagement Hub.

### 5.3.4.2  UI mockups and Sequence Diagrams

The following sub-sections show sequence diagrams to clarify the stories sketched above and the vf-OS internal interactions related to them. As the External Service Provision is a completely internal component without any direct user interaction, no UI Mock-ups have been provided for it.

#### 5.3.4.2.1  Download Wrapper and Support Libraries

Figure 238 shows the sequence diagram related to the downloading of wrapper and support libraries from the library repositories. The Marketplace frontend plays a vital role in this process: it completely manages the interaction with the end user, which is, in fact, agnostic about the very existence of the external service provision repositories as such. For collecting its list of available items, it gets the repository items from the repositories. As
soon as a user puts one of the libraries in its cart, the actual downloading of the items (after payment) is managed by the Marketplace.

The main steps/functionality are:
- Download ESP Library from Marketplace

5.3.4.2.2 Integrate Wrapper and Support Library in a vApp
Developers can integrate the libraries in their vApps by means of the Studio. This functionality is described in the Studio section under user stories STUS052 and STUS104. The marketplace plays a role in this process as well.

The main steps/functionality are:
- Integrate ESP Library in vApp

5.3.4.2.3 Use Wrapper and Support Library Functions inside vApps
A vApp with a built-in wrapper or support library for external service provision can be used by end users in the same way as vApps that do not use such libraries. The vApp may just call the functions within the library as if it were completely internal functions. Note that these functions can be synchronous or asynchronous (with callbacks), depending on the nature of the interaction with the external service. Importantly, the configuration of the
vApp inside the portal must allow for calls to external services (eg via HTTP or otherwise). The sequence diagram for external service calls inside a vApp is shown in Figure 239.

The main steps/functionalities are:

- Execute ESP Library Functions

5.3.4.2.4 Manage Wrapper and Support Library Repositories

The management of the external service library repositories takes place via the Marketplace backend. After processing requests for adding new items or deleting existing ones, these items are added to resp. deleted from the appropriate repository. Figure 240 shows the sequence diagram for this functionality.

The main steps/functionalities are:

- Add Library to Repository
5.3.4.2.5 Use and Manage External Service Documentation
For the ESP documentation repository, a simple integration with the Developer Engagement Hub is foreseen. This Hub can directly retrieve documentation from the
repository, as well as add and delete documentation from it. The related sequence diagram is shown in Figure 241.

The main steps/functionalities are:

- Read Documentation
- Add Documentation to ESP documentation repository
- Delete Documentation from ESP documentation repository

![Notifications Execution Sequence Diagram](image)

**Figure 241: Notifications Execution Sequence Diagram**

### 5.3.4.3 Interaction description

An Interaction Diagram for the External Service Provision is shown in Figure 242. The Marketplace and Developer Engagement Hub interact with the repositories, while vApps can use specific libraries from these repositories for providing their functionality. The libraries themselves may directly or indirectly (via Pub/Sub or Messaging) use the external services. The exchanged data is straightforward:

- Between Marketplace and library repositories: lists of libraries, handles to download or upload libraries (new versions of existing libraries are seen as new libraries)
- Between Developer Engagement Hub and documentation repository: lists of documentation artefacts, as well as documentation artefacts themselves
- Between library functions and Pub/Sub: subscription requests and external service publications
- Between library functions and Messaging: messages with an unspecified payload to be exchanged with external services and vice versa

![I/O Toolkit: (T7.3) Service Provision Diagram]

Figure 242: I/O Toolkit Interaction Diagram

5.4 Control

5.4.1 System Dashboard

The System Dashboard offers a control panel for the vf-OS platform by integrating the various dashboards of individual platform components and vApps into a coherent whole. It acts as the main entry point for the configuration of all assets installed on the platform, including the platform itself.
5.4.1.1 Behaviour and Functionality

The vf-OS System Dashboard provides the following main functionalities:

- **Unified Dashboard Entrypoint:** it provides an overview of dashboards of installed components and vApps
- **Uniform Dashboard UI:** It provides a coherent user interface and look-and-feel for each of the dashboards of components and vApps
- **Component and vApp Statistics:** It provides functionality to store and aggregate status control data about and from components and vApps, as well as ways to visualise the actual and historical data
- **Notifications:** It provides mechanisms for managing automated notifications about the status of vApps
- **Data Maintenance:** It provides some basic maintenance functionality for the System Dashboard itself

An overview of activities, tasks and stories related to the System Dashboard is shown in Figure 243.
The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
</table>
| SHUS001       | Description: Who: platform user/admin  
What: open the main dashboard UI to get an overview of dashboards that can be accessed by this user  
Why: in order to access a specific dashboard for monitoring and configuration |
## SHUS002 use dashboard

**Description**
- **Who**: platform user/admin
- **What**: open a component dashboard or vApp dashboard
- **Why**: in order to monitor and configure the component or vApp

**Acceptance Criteria**
User clicks dashboard in vf-OS portal and gets an overview screen with dashboard icons in line with the access control rights assigned to this user.

## SHUS003 use aggregated status info visualisations

**Description**
- **Who**: platform user/admin
- **What**: inspect aggregated status information inside a component or vApp dashboard
- **Why**: in order to get insight into the control status of the component or vApp (e.g., usage of resources, log reports) over time

**Acceptance Criteria**
From the dashboard overview screen, user clicks a dashboard icon of a component or vApp. The contents of the dashboard for this component or vApp are presented to the user. Modifications to the settings of a component or vApp are propagated to the respective component or vApp.

## SHUS004 view notification list

**Description**
- **Who**: platform user/admin
- **What**: inspect an overview of automated notifications configured in the platform
- **Why**: in order to check or edit settings, to create new notifications or delete existing ones

**Acceptance Criteria**
User clicks notifications management in the dashboard and gets presented a list of notifications in line with the access control rights assigned to this user.

## SHUS005 create notification

**Description**
- **Who**: platform user/admin
- **What**: create a new automated notification
- **Why**: in order to automatically receive notifications as soon as certain status information for a component or vApp reach a configured threshold

**Acceptance Criteria**
User clicks create notification icon and are able to provide the appropriate information and settings for the new notification. After confirmation, the notification is visible on the notification list.

## SHUS006 edit notification settings

**Description**
- **Who**: platform user/admin
- **What**: edit an existing automated notification
- **Why**: in order to change the settings, parameters, thresholds for this notification

**Acceptance Criteria**
The user is able to modify the settings for a notification. After confirmation, the settings are updated.

## SHUS007 delete notification

**Description**
- **Who**: platform user/admin
- **What**: delete an automated notification
- **Why**: in order to stop the notification from happening

**Acceptance Criteria**
The user is able to delete a notification. After confirmation, the notification is removed from the notification list.

## SHUS008 receive asset status

**Description**
- **Who**: dashboard component
<table>
<thead>
<tr>
<th>Description</th>
<th>Who:</th>
<th>What:</th>
<th>Why:</th>
</tr>
</thead>
</table>
| **What:** periodically receive status control info from vf-OS components and vApps  
**Why:** in order to provide to the user with both past and present status control information  
**Acceptance Criteria**  
Component or vApp sends status control info to the System Dashboard Monitoring & Control Interface of the dashboard. Dashboard receives the information and stores it in the Data Aggregation Module. |
| **SHUS009 aggregate status information** | dashboard component | aggregate status control data from components and vApps, either periodically or upon demand | in order to present aggregated data to platform users/admins and/or to trigger notifications  
**Acceptance Criteria**  
Upon request, platform users/admins are able to inspect aggregated data in a component or vApp dashboard. |
| **SHUS010 send notification** | dashboard component | send a notification as soon as a threshold for a certain automated notification is reached | in order to inform platform users/admins about critical statuses of components or vApps  
**Acceptance Criteria**  
The user configures automated notification for a component or vApp. Component or vApp provides status control info to the dashboard component, either upon its own initiative or upon request. The info is automatically aggregated. Once certain (raw or aggregated) status control info reaches the configured threshold, the user receives a notification via a pre-configured communication channel. |
| **SHUS011 view aggregated data store status** | platform user/admin | get insight into the status of the aggregated data store within the Data Aggregation Module of the dashboard | to check whether the store should be cleaned up  
**Acceptance Criteria**  
The user is able to inspect the status of the aggregated data store (amount of data, total period, etc). |
| **SHUS012 cleanup aggregated data store** | platform user/admin | cleanup the aggregated data store within the Data Aggregation Module of the dashboard | to free space or to delete historical information for technical or legal reasons  
**Acceptance Criteria**  
The user cleans up the aggregated data store by removing data eg older than a certain date. The data is completely removed from the aggregated data store. The removed data is no longer shown in aggregated data overviews of the related dashboards. |

### 5.4.1.2 UI mockups and Sequence Diagrams

The following sub-sections show sequence diagrams and UI mockups to clarify the stories sketched above and the vf-OS internal interactions related to them.

#### 5.4.1.2.1 Dashboard Overview
Figure 238 shows the sequence diagram related to the provision of the Dashboard Overview. Initiation takes place at the Platform. The System Dashboard UI retrieves the list of dashboards from installed components and vApps and shows them as icons on the main overview.

Figure 244: Dashboard Overview Sequence Diagram

The main steps/functionalities are:

- Use Overview of Dashboards

The user interface is shown in Figure 245.
5.4.1.2.2 Use Component and vApp Dashboards

If a user clicks one of the icons in the main dashboard overview, the particular dashboard for that component or vApp is opened and its contents are shown. The specifics of the contents are determined by the component / vApp, while the rendering of the user interface is being done by the System Dashboard. The sequence diagram for this activity is shown in Figure 246.

The main steps/functionalities are:

- Use Dashboard
- Use Aggregated Status Info Visualisations
D2.2: Functional Specifications & Mockups - Vs: 1.0 - Public

An example user interface for part of the dashboard of the Platform is shown in Figure 247. Each dashboard has a fixed layout: on the left a list of categories, on the right two sections, one with statistics and one with settings. The specific content inside each of the sections is determined by the component / vApp to which the dashboard belongs. Visualisations of statistics are generated by the System Dashboard, as their contents are based on the contents of the Data Aggregation Module inside the System Dashboard.
5.4.1.2.3 Notifications Management

Another functionality offered by the System Dashboard is to manage and execute automated notifications based on certain status control data collected from vApps. Figure 248 shows the sequence diagram for performing CRUD actions on the list of notifications.

The main steps/functionalities are:

- View Notification List
- Create Notification
- Edit Notification Settings
- Delete Notification
The UI mock-up for this functionality is shown in Figure 249. A layout is chosen similar to the standard dashboard layout. On the left, the installed vApps for which notifications can be configured are shown. On the right, a list of notifications is shown which is configured for the selected vApp. Notifications can be added and deleted by clicking on the respective rows in the list. The button in the right lower corner enables users to add new notifications. The types of notifications available are determined by the vApps themselves. They need to provide the appropriate meta-information to the System Dashboard for configuration purposes.
5.4.1.2.4 Notifications Execution

The execution of notifications, ie the sending of messages to users via a diversity of channels based on triggers upon the particular status about or identified by a vApp, is part of the System Dashboard as well. The related sequence diagram is shown in Figure 250.

The main steps/functionality are:

- Receive Asset Status
- Aggregate Status Information
- Send Notification

This functionality is entirely executed in the background. No user interface is involved in it. Therefore, no mock-up is provided.
5.4.1.2.5 Maintenance

Finally, basic functionality is provided for maintenance purposes, i.e. for cleaning up the data aggregation module in the System Dashboard. This module collects status data and keeps it for aggregation purposes. Once in a while, administrators might want to purge historical data. The sequence diagram for this functionality is shown in Figure 251.

The main steps/functionabilities are:

- View Aggregated Data Store Status
- Clean-up Aggregated Data Store

Figure 252 shows the UI mock-up for the maintenance functionality. On the upper part, statistics are shown for each of the vApps and components for which historical status data is kept. The lower part enables the user to purge historical data.
Figure 251: Maintenance Sequence Diagram
An Interaction Diagram for the System Dashboard is shown in Figure 253. The System Dashboard interacts with all installed platform components and vApps in order to render and present their dashboards to the end user. The data exchanged comprises:

- Status Control Information. This comprises two different elements:
  
  - information about the status of a component or vApp, which is being collected in the Data Aggregation Module for providing statistics about the component or vApp to the end user
  
  - metadata describing fields and values which can be altered by the end user to change the settings for this component or vApp. The value types are often expected to be simple, but may as well become quite complex and different rendering mechanisms may be used to present the different dashboard elements to the end user (eg tables, lists, dropdown boxes, graphs with nodes,
The System Dashboard provides the means to build the user interface around this information, the components and vApps themselves determine which fields and possible values are presented to the user.

- **Actions to be performed.** This information is completely based on the metadata provided by the component or vApp itself. For example, if the metadata describes that a vApp requires the configuration of a particular type of temperature sensor, it specifies the field “temperature sensor” and supported values for this field. The end user of the System Dashboard is able to configure the value of the “temperature sensor” field. As soon as the field is changed, an action to be performed will be sent to the vApp to effectuate the configuration change.

Furthermore, there is a connection with External Messaging infrastructure. The minimal setup of the System Dashboard requires the availability of an email server. Different servers for messaging or SMS may be attached to it as well.

Figure 253: System Dashboard Interaction Diagram
6 Application-Deployment (Use) Components

6.1 Marketplace Services

6.1.1 vf-OS Marketplace

6.1.1.1 Behaviour and Functionality

The purpose of the marketplace is to provide a central point for users and developers to receive and offer (respectively) vf-OS Assets of different kinds such as vApps or Device Enablers.

The main part of the Marketplace is the vf-Store. It will be the core component for additional services that go beyond the core functionalities of vf-OS. This way the vf-Store can be seen as a general app store like the Apple App Store or Google Play Store but with additional functionalities. Users will be able to browse existing vf-OS Assets based on characteristics of the vf-OS Assets such as their price or category. After that, the user will be able to purchase vf-OS Assets as, for example, a one-time fee or on a pay-per-use basis, and acquire those. To support pay-per-use, interfaces will be provided to keep track of the remaining usages of a vf-OS Assets. In a general manner, it is foreseen to provide a public API to extend the vf-Store with external functionalities.
Figure 254: Marketplace Story Map (1)

The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
</table>
| VMUS001 Receive information from external service provider | Description  
Who: vf-OS Store  
What: receive data from external service provider  
Why: so that order can be marked as paid  

Acceptance Criteria  
Make sure the data (payment status paid) is saved correctly in order in database |
| VMUS002 Change information in order record accordingly | Description  
Who: vf-OS Store  
What: change information in order record  
Why: to represent the new status  

Acceptance Criteria  
Make sure the data (status) is saved in order in a database |
| VMUS003 Send information about payment to customer | Description  
Who: vf-OS Store  
What: send notification to customer that payment was received  
Why: so that the customer knows everything went well  

Acceptance Criteria  
Notification is sent to customer |
| VMUS004 Receive information from external service provider | Description  
Who: vf-OS Store  
What: receive data from external service provider  
Why: so that order can be marked as error  

Acceptance Criteria  
Make sure the data (payment status error) is saved correctly in order in database |
<p>| VMUS005 | Description |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Who</th>
<th>What</th>
<th>Why</th>
<th>Acceptance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change information in order record accordingly</td>
<td></td>
<td>vf-OS Store</td>
<td>change information in order record</td>
<td>to represent the new status</td>
<td>Make sure the data (status) is saved correctly in order in database</td>
</tr>
<tr>
<td>VMUS006 Send information about error to customer</td>
<td>Description</td>
<td>vf-OS Store</td>
<td>send notification to customer</td>
<td>so the customer knows an error occurred</td>
<td>Notification is sent to customer</td>
</tr>
<tr>
<td>VMUS007 Receive information from external provider</td>
<td></td>
<td>vf-OS Store</td>
<td>receive data from external service provider</td>
<td>so that order can be marked as cancelled</td>
<td>Make sure the data (payment status cancelled) is saved in order in the database</td>
</tr>
<tr>
<td>VMUS008 Change information in order record accordingly</td>
<td>Description</td>
<td>vf-OS Store</td>
<td>change information in order record</td>
<td>to represent the new status</td>
<td>Make sure the data (status) is saved in order in the database</td>
</tr>
<tr>
<td>VMUS009 Send information about cancellation to customer</td>
<td>Description</td>
<td>vf-OS Store</td>
<td>send notification to customer</td>
<td>so the customer knows the payment was cancelled</td>
<td>Notification is sent to customer</td>
</tr>
<tr>
<td>VMUS010 Receive information from external provider</td>
<td></td>
<td>vf-OS Store</td>
<td>receive data from external service provider</td>
<td>so that the order can be marked as cancelled</td>
<td>Make sure the data (payment status cancelled) is saved in order in the database</td>
</tr>
<tr>
<td>VMUS011 Send information about cancellation to customer</td>
<td>Description</td>
<td>vf-OS Store</td>
<td>change information in order record</td>
<td>to represent the new status</td>
<td>Make sure the data (status) is saved in order in the database</td>
</tr>
<tr>
<td>VMUS012 Add wanted items to cart</td>
<td>Description</td>
<td>End user</td>
<td>adds requested items to the cart</td>
<td>to be able to purchase multiple items at once</td>
<td>The requested item has been added successfully and is displayed on the &quot;current cart&quot; overview.</td>
</tr>
<tr>
<td>VMUS013 Enter information on checkout page</td>
<td>Description</td>
<td>End user</td>
<td>enters the needed information in a form</td>
<td>to make them available to the vf-Store after sending the form</td>
<td></td>
</tr>
</tbody>
</table>
### Acceptance Criteria

- **The information is entered correctly.** This includes various checks eg for a correct email address, correct credit card information.

### VMUS014 Save order in database

**Description**

- **Who:** vf-Store
- **What:** saves order record in the database
- **Why:** to process the order

**Acceptance Criteria**

The order has been saved. An order number has been generated for the order.

### VMUS015 Authorise payment

**Description**

- **Who:** End user
- **What:** authorises the payment by logging into its PayPal account or similar
- **Why:** to be able to purchase multiple items at once

**Acceptance Criteria**

The user is redirected back to the vf-Store. The most important aspect in the handling of payment information is the security of the data. Therefore all connections must be secured eg by using encryption.

### VMUS016 Send invoice to customer

**Description**

- **Who:** vf-Store
- **What:** sends an invoice to the end user
- **Why:** for legal reasons

**Acceptance Criteria**

The invoice has been sent successfully.

### VMUS017 Customer logs in to personal account

**Description**

- **Who:** End user
- **What:** logs in to personal account
- **Why:** to see information about own account

**Acceptance Criteria**

The end user entered the correct password and is redirected to account.

### VMUS018 Download invoice for order

**Description**

- **Who:** End user
- **What:** chooses invoice to download
- **Why:** to download invoice

**Acceptance Criteria**

Download starts.

### VMUS019 Customer care staff logs in to vf-Store Administration UI

**Description**

- **Who:** Customer care staff
- **What:** logs in to vf-Store Administration UI
- **Why:** to access restricted resources

**Acceptance Criteria**

User is logged in and redirected to dashboard.

### VMUS020 Search for order

**Description**

- **Who:** Customer care staff
- **What:** search for an order based on various criteria
- **Why:** to find a specific order

**Acceptance Criteria**

Order is displayed.

### VMUS021 Change attributes

**Description**

- **Who:** Customer care staff
- **What:** changes attributes of the order
- **Why:** to change the order

**Acceptance Criteria**

The information of the order is changed.
<table>
<thead>
<tr>
<th>VMUS022</th>
<th>Save order in database</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vf-Store</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> saves order record in the database</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to keep updated information</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The order has been saved.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS023</th>
<th>Customer care staff logs in to vf-Store Administration UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Customer care staff</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> logs into vf-Store Administration UI</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to access restricted resources</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>User is logged in and redirected to dashboard</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS024</th>
<th>Search for order</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Customer care staff</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> search for an order based on various criteria</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to find a specific order</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Order is displayed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS025</th>
<th>Define refund amount</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Customer care staff</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> defines the amount that will be refunded</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to give money back to the buyer</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>The defined amount is accepted. Eg the amount is not higher than the previously paid amount and the currency is correct.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS026</th>
<th>Initiate refund amount</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Customer care staff</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> initiates refund process</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to give money back to the buyer</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Refund process initiated</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS027</th>
<th>Send request for refund to external payment providers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> vf-Store</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> sends request to external payment provider</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to initiate refund process</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Request accepted by external payment provider</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS028</th>
<th>Customer care staff logs in to vf-Store Administration UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Customer care staff</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> logs into vf-Store Administration UI</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to access restricted resources</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>User is logged in and redirected to dashboard</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS029</th>
<th>Access dashboard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Customer care staff</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> access the dashboard</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to see an overview of order statistics</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>Dashboard is displayed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS030</th>
<th>Log in to vf-Store Administration UI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Who:</strong> Developer</td>
<td></td>
</tr>
<tr>
<td><strong>What:</strong> logs into vf-Store Administration UI</td>
<td></td>
</tr>
<tr>
<td><strong>Why:</strong> to access restricted resources</td>
<td></td>
</tr>
<tr>
<td>VMUS031</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Choose vf-OS Asset to be updated</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
<td></td>
</tr>
<tr>
<td>User is logged in and redirected to dashboard</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td>Who: Developer</td>
<td></td>
</tr>
<tr>
<td>What: chooses from a list a vf-OS Asset</td>
<td></td>
</tr>
<tr>
<td>Why: to access detailed information</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS032</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Upload new version</strong></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td>Information about the vf-OS Asset are displayed</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: Developer</td>
</tr>
<tr>
<td>What: uploads a new version of the vf-OS Asset and enters detailed information</td>
</tr>
<tr>
<td>Why: to provide new vf-OS Asset version</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS033</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Log in to vf-Store Administration UI</strong></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td>User is logged in and redirected to dashboard</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: Developer</td>
</tr>
<tr>
<td>What: logs into vf-Store Administration UI</td>
</tr>
<tr>
<td>Why: to access restricted resources</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS034</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create new vf-OS Asset</strong></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td>Wizard to create new vf-OS Asset is displayed</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: Developer</td>
</tr>
<tr>
<td>What: chooses option to create a new vf-OS Asset</td>
</tr>
<tr>
<td>Why: to access wizard to add information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS035</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add information about new vf-OS Asset</strong></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td>Wizard is completed and new vf-OS Asset is created</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: Developer</td>
</tr>
<tr>
<td>What: enters information about newly added vf-OS Asset</td>
</tr>
<tr>
<td>Why: to complete wizard</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS036</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Log in to vf-Store Administration UI</strong></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td>User is logged in and redirected to dashboard</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: Developer</td>
</tr>
<tr>
<td>What: logs into vf-Store Administration UI</td>
</tr>
<tr>
<td>Why: to access restricted resources</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS037</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Choose vf-OS Asset to be updated</strong></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td>Information about the vf-OS Asset is displayed</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: Developer</td>
</tr>
<tr>
<td>What: chooses from a list a vf-OS Asset</td>
</tr>
<tr>
<td>Why: to access detailed information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VMUS038</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enter pricing information</strong></td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong></td>
</tr>
<tr>
<td>Information about the vf-OS Asset is saved. Pricing models can vary from pay-per-use fees to a one-time fee. Furthermore, the user must be able to enter prices for different currencies.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Who: Developer</td>
</tr>
<tr>
<td>What: enters updated information</td>
</tr>
<tr>
<td>Why: to change information</td>
</tr>
</tbody>
</table>
### Log in to vf-Store Administration UI

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What: logs into vf-Store Administration UI</td>
<td></td>
</tr>
<tr>
<td>Why: to access restricted resources</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**
User is logged in and redirected to dashboard

### VMUS040 Choose vf-OS Asset to view error reports for

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What: chooses from a list a vf-OS Asset</td>
<td></td>
</tr>
<tr>
<td>Why: to access detailed information</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**
Information about the vf-OS Asset is displayed

### VMUS041 Choose error report

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What: chooses error report from a list</td>
<td></td>
</tr>
<tr>
<td>Why: to get detailed information about the error</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**
Information about the vf-OS Asset are displayed

### VMUS042 Receive usage statistics from vf-P

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: vf-OS Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>What: receive data from vf-P</td>
<td></td>
</tr>
<tr>
<td>Why: so the vf-Store learns how vf-OS Assets are used</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**
Usage statistics are saved to the database. These must include, among others, a user identification to map the information to the correct user.

### VMUS043 Receive error report from vf-P

<table>
<thead>
<tr>
<th>Description</th>
<th>Who: vf-OS Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>What: receive data from vf-P</td>
<td></td>
</tr>
<tr>
<td>Why: so the vf-Store learns when a problem with a vf-OS Asset occurs</td>
<td></td>
</tr>
</tbody>
</table>

**Acceptance Criteria**
Error reports are saved to the database. All information provided must be saved in a way that makes them searchable by various criteria.

### 6.1.1.2 UI mockups and Sequence Diagrams

The following subsections describe the UI mockups and sequence diagrams describing the interaction.

### 6.1.1.2.1 Receive information on paid order from external payment providers

This feature enables the marketplace to receive information on paid orders. This information is provided by an external payment provider.

The main steps/functionalities are:
- Receive information from external service provider
- Change information in order record accordingly
- Send information about payment to customer
6.1.1.2.2 Receive Information on errors in payment process from external payment providers

This feature enables the marketplace to receive information on errors in the payment process. This information is provided by an external payment provider.

The main steps(functionalities are):

- Receive information from external service provider
- Change information in order record accordingly
- Send information about error to customer
6.1.1.2.3 Receive Information on cancelled payment from external payment providers

This feature enables the marketplace to receive information on cancelled payments. This information is provided by an external payment provider.

The main steps/functionalities are:

- Receive information from external service provider
- Change information in order record accordingly
- Send information about cancellation to customer
6.1.1.2.4 Receive Information on refund from external payment providers

This feature enables the marketplace to receive information on a refund. This information is provided by an external payment provider.

The main steps/functionalities are:

- Receive information from external service provider
- Change information in order record accordingly
Figure 259: Receive Information on Refund from External Payment Providers Diagram

6.1.1.2.5 **Place new order**

This feature enables users to place new orders that are then handled and processed by the marketplace.

The main steps/functionalities are:

- Add wanted items to cart
- Enter information on checkout page
- Save order in database
- Authorise payment
- Send invoice to customer
The UI for placing new orders is as follows:

![Figure 261: Place New Order UI Mockup](image)

### 6.1.1.2.6 Download invoice

This feature enables users to download invoices for placed orders.

The main steps/functionalities are:

- Customer logs into personal account
- Download invoice for order
The UIs for downloading invoices are as follows:
6.1.1.2.7 Change attributes of an order

This feature enables users to change attributes of placed orders.

The main steps/functionalities are:

- Customer care staff logs in to vf-Store Administration UI
- Search for order
- Change attributes
- Save order in database
Figure 265: Change Attributes of an Order Diagram

The UIs for changing attributes of an order are as follows:
Figure 266: Change Attributes of an Order Login to Account UI Mockup

Figure 267: Change Attributes of an Order search UI Mockup
6.1.1.2.8 **Refund payment**

This feature enables the refund of payments of placed orders.

The main steps/functionalities are:
- Customer care staff logs in to vf-Store Administration UI
- Search for order
- Define refund amount
- Initiate refund amount
- Send request for refund to external payment providers
The UIs for refunding payments are as follows:
Figure 270: Refund Payment Login to Account UI Mockup

Figure 271: Refund Payment Order Search UI Mockup
6.1.1.2.9 Get statistics about orders
This feature enables users to get statistics placed orders.

The main steps/functionalities are:

- Customer care staff logs in to vf-Store Administration UI
- Access dashboard
The UIs for getting statistics about orders are as follows:
Figure 275: Get Statistics about Orders Login to Account UI Mockup

Figure 276: Get Statistics about Orders Dashboard UI Mockup

6.1.1.2.10 **Upload new version of vf-OS Asset**

This feature enables developers to upload new versions of vf-OS Assets.

The main steps/functionality are:

- Log in to vf-Store Administration UI
- Choose vf-OS Asset to be updated
- Upload new version
The UIs for uploading new versions of vf-OS Assets are as follows:

Figure 277: Upload New Version of vf-OS Asset Diagram
Figure 278: Upload New Version of vf-OS Asset Login to Account UI Mockup

Figure 279: Upload New Version of vf-OS Asset Asset List UI Mockup
6.1.1.2.11 Create new vf-OS Asset
This feature enables developers to create new vf-OS Assets.

The main steps/functionalities are:

- Log into vf-Store Administration UI
- Create new vf-OS Asset
- Add information about new vf-OS Asset

Figure 280: Upload New Version of vf-OS Asset Detail UI Mockup

Figure 281: Create New vf-OS Asset Diagram
The UIs for creating new vf-OS Assets are as follows:

![vf-Store Backend UI](image)

**Figure 282: Create New vf-OS Asset Login to Account UI Mockup**

![vf-OS Backend UI](image)

**Figure 283: Create New vf-OS Asset Detail UI Mockup**

### 6.1.1.2.12 Change pricing model of an existing vf-OS Asset

This feature enables developers to change the pricing model of existing vf-OS Assets.

The main steps=functionalities are:

- Log in to vf-Store Administration UI
- Choose vf-OS Asset to be updated
- Enter pricing information
The UIs for changing the pricing model of an existing vf-OS Asset are as follows:
Figure 285: Change Pricing Model of an Existing vf-OS Asset Login to Account UI Mockup

Figure 286: Change Pricing Model of an Existing vf-OS Asset Choose Asset UI Mockup
6.1.1.2.13 **View current error reports for vf-OS Asset**

This feature enables developers to view error reports of vf-OS Assets.

The main steps/functionalities are:

- Log in to vf-Store Administration UI
- Choose vf-OS Asset to view error reports for
- Choose error report

![Diagram of error reports](image)

Figure 287: Change Pricing Model of an Existing vf-OS Asset Detail UI Mockup

Figure 288: View Current Error Reports for vf-OS Asset Diagram
The UIs for viewing error reports for vf-OS Assets are as follows:

Figure 289: View Current Error Reports for vf-OS Asset Login to Account UI Mockup
Figure 290: View Current Error Reports for vf-OS Asset Choose Asset UI Mockup

Figure 291: View Current Error Reports for vf-OS Asset Detail UI Mockup
6.1.1.2.14 **Receive usage statistics from vf-P**
This feature enables the marketplace to receive usage statistics about vf-OS Assets.
The main steps/functionalities are:
- Receive usage statistics from vf-P

![Diagram showing usage statistics](image1)

6.1.1.2.15 **Receive error report from vf-P**
This feature enables the marketplace to receive error reports about vf-OS Assets.
The main steps/functionalities are:
- Receive error report from vf-P

![Diagram showing error report](image2)
6.1.1.3 Interaction description

The following diagram (Figure 295) was taken from the global architecture definition (D2.1) as there have been no changes since.
Figure 295: Marketplace Component Interactions

The information exchange with other components is as follows:
Figure 296: Marketplace Class Diagram 1
6.2 vf-OS Assets

6.2.1 vf-OS Enablers

6.2.1.1 Behaviour and Functionality

As defined in the story maps, the vf-OS Enablers will be connected to the vf-OS Enablers Framework on the I/O Toolkit and based on that connection, the Enabler’s internal logic will receive a configuration defined on the framework side, and apply it to shape the behaviour according to that configuration.

Besides the configuration parameters that can be received, the Enabler’s internal logic will be able to upload data to the vf-OS platform in two ways: Synchronously and Asynchronously. However, this behaviour will be affected to the internal functionality to address the needs of each pilot as well as the entire vf-OS platform.

Based on the way each Enabler will generate data, the Monitoring Services will create aside bridge between the Enabler itself and the data destination to analyse that data and create some performance analysis.

Following, there is a story map where the main features, epics and user stories for the vf-OS Specific Enablers components have been identified (see Figure 298).
The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VEUS001</td>
<td><strong>Apply configuration</strong></td>
</tr>
</tbody>
</table>
| **Description** | Who: vf-OS Specific Enabler  
What: Receive and apply configuration  
Why: to be configured as expected by the end user |
| **Acceptance Criteria** | Specific Enabler must be able to receive the configuration and configure itself to start working as expected |
| VEUS004       | **Notify configuration status**                                                      |
| **Description** | Who: EV  
What: notify EF  
Why: to notify if the configuration file was loaded properly or not |
| **Acceptance Criteria** | Specific Enabler must notify the configuration status towards the EF |
| VEUS002       | **Provide synchronous generated**                                                   |
| **Description** | Who: vf-OS Specific Enabler |

Figure 298: vf-OS Specific Enabler Story map
<table>
<thead>
<tr>
<th><strong>data</strong></th>
<th><strong>Description</strong></th>
<th><strong>Acceptance Criteria</strong></th>
</tr>
</thead>
</table>
| **VEUS003** Provide asynchronous generated data | **Who:** vf-OS Specific Enabler  
**What:** Send data to vf-OS Enablers' Framework  
**Why:** to be used by the vf-OS Enablers' Framework requester | The request owner (EF) should receive generated data upon each data request sent |
| **VEUS005** Provide synchronous data towards Monitoring Services | **Who:** vf-OS Specific Enabler  
**What:** Send data to Monitoring Services & vf-OS Enablers' Framework  
**Why:** to monitor the data flow | Specific enabler should send generated data towards Monitoring services upon each new data request coming from EF. The monitoring services should receive the generated data. The EF should receive the generated data as a response from the data request |
| **VEUS006** Provide asynchronous data towards Monitoring Services | **Who:** vf-OS Specific Enabler  
**What:** Send data to Monitoring Services & vf-OS Enablers' Framework  
**Why:** to be monitoring the data flow asynchronously | The monitoring services should receive the generated data. The EF should receive the generated data |
| **VEUS007** Analyse received data | **Who:** Monitor Services  
**What:** receive data and monitor it  
**Why:** to monitor the data flow asynchronously | Monitoring services should receive the generated data and analyse it accordingly |
| **VEUS008** Send performance reports towards “Provides Data” interface | **Who:** Monitor Services  
**What:** upload PM and FM to EF  
**Why:** to enable EF to monitor the EV | Monitoring Services should upload the PM/FM to EF accordingly to the granularity defined |
| **VEUS009** Provide generated data towards “Provides Data Interface” | **Who:** Monitor Services  
**What:** upload data to EF  
**Why:** to enable a monitoring data proxy towards EF | Monitoring Services should act as a proxy between Specific Enabler and EF. Incoming data must be equal to the outgoing data |

### 6.2.1.2 UI mockups and Sequence Diagrams

The following sub-sections describe the sequence diagrams describing the interaction.
6.2.1.2.1 Manage Configuration
This feature provides the capability to receive the configuration file where the developer of each enabler will be able to configure how each enabler should work. The Enabler should be able to apply the configuration and after being applied, the Enabler should work standalone in line with the configuration received previously. The configuration will be sent by the Enabler’s Framework and each Specific Enabler, after applying the configuration, should emit an acknowledgement notifying the Enabler’s Framework regarding the status of that configuration received, by marking the configuration as applied or not.

![Diagram of Manage Configuration](image)

Figure 299: Manage Configuration

6.2.1.2.2 Generate Data
This feature represents the internal specific logic developments oriented to different types of use, either to be integrated into other enablers and thus to fulfil the pilots and project need as a whole, or they can be used to cover specific functionality needs of the vf-OS platform itself. This internal feature provides cross-backend services to Assets whenever they need it. The generated data will be routed by the Enabler’s Framework whenever the Asset needs it. Thus this need will be handled by the Enabler’s Framework by invoking the vf-OS Specific Enabler’s in order to receive the needed generated data and then it should route the data to who should receive it. vf-OS Specific Enablers can generate data and upload it to the Enabler’s Framework by their own (Asynchronously) if this behaviour is set on the configuration file received previously. Whenever this is the case, the Enabler’s Framework should notify the vf-OS Specific Enabler that the generated data was received properly to fulfil the quality of the service provided by each Enabler.
Figure 300: Generate Data
6.2.1.2.3 Monitoring Services

Monitoring Services acts as an internal Enabler’s component providing performance management (PM) and fault management (FM) to the Enabler’s Framework. This feature should provide reports to the Enabler’s Framework related to the performance of each Enabler. Apart from this behaviour, the Monitoring Services can provide to each Enabler a proxy service where he can forward the generated data to the Enabler’s Framework. Acting as a proxy, the Monitoring Services will analyse the generated data and only after it will forward the data towards the Enabler’s Framework.
6.2.1.3 Interaction Description

From the previous description of the functionality covered by the vf-OS Specific Enabler module, a deeper level of detail regarding the main components of the Specific Enabler and the interaction between the Enabler's Framework and those functional development of the vf-OS Specific Enablers. Following there is a picture showing the flow of information exchange between the Specific Enablers subcomponents and vf-OS components.
Figure 302: vf-OS Specific Enablers Component Interaction Diagram

The messages exchange are:
6.2.2 FIWARE Manufacturing Enablers

The purpose of this module is to provide from FIWARE Manufacturing Enablers core functionalities such as Collaborative Asset Management, Advanced Management of Collaborative Assets, Shopfloor Data Collection...

The enablers will form the basis for developing applications by integrating existing manufacturing enablers serving different functional requirements. Enablers are of interest, also to reuse solutions in the domain that are not addressed specifically in vf-OS, such as IoT enablers, Supply Chain enablers, Collaborative Enablers and Data Analytics Enablers. These enablers contribute to form the core of the vf-SK and will be distributed as packages to be installed as services in the selected run-time environment of vf-OS.

6.2.2.1 Behaviour and Functionality

Manufacturing Enablers component provide a set of functionalities that could be grouped on the following features:

- **Configuration management**: Where a range of functionality is provided to list existing Enablers, add or update a configuration of a Manufacturing Enabler.

- **Enabler Services**: This module provides APIs to get access to invoke functionalities of the Manufacturing Enablers, and check statistics of Exposed Manufacturing Enablers.

Following, there is a story map where the main features and user stories for the Manufacturing Enablers components have been identified (see Figure 304).
The textual description of each user story is as follows:

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Subtask description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMUS001 Configuration service</td>
<td><strong>Description</strong>&lt;br&gt;Who: Manufacturing Enabler&lt;br&gt;What: Receive configurations from configuration's user interface&lt;br&gt;Why: In order to configure the Manufacturing Enabler&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;Make sure all configuration validations are valid</td>
</tr>
<tr>
<td>FMUS002 Configuration Manager UI</td>
<td><strong>Description</strong>&lt;br&gt;Who: Manufacturing Enabler configuration's user interface&lt;br&gt;What: Provides the necessary functionalities for Admin user&lt;br&gt;Why: to define and update the configuration parameters of the Manufacturing Enablers&lt;br&gt;<strong>Acceptance Criteria</strong>&lt;br&gt;Make sure configurations are in the same types and have the correct limits</td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FMUS003</td>
<td><strong>List enabler configuration</strong></td>
</tr>
<tr>
<td>FMUS004</td>
<td><strong>Invoke enabler APIs</strong></td>
</tr>
<tr>
<td>FMUS005</td>
<td><strong>Asynchronous response data</strong></td>
</tr>
<tr>
<td>FMUS006</td>
<td><strong>Analyse received data</strong></td>
</tr>
<tr>
<td>FMUS007</td>
<td><strong>Provide performance details</strong></td>
</tr>
<tr>
<td>FMUS008</td>
<td><strong>Qualification Service</strong></td>
</tr>
<tr>
<td>FMUS009</td>
<td><strong>Qualification Manager UI</strong></td>
</tr>
<tr>
<td>FMUS010</td>
<td><strong>Adaptation Service</strong></td>
</tr>
<tr>
<td>FMUS011</td>
<td><strong>Adaptation Manager UI</strong></td>
</tr>
</tbody>
</table>
6.2.2.2 UI mockups and Sequence Diagrams

The following sub-sections describe the UI mockups and sequence diagrams describing the interactions.

6.2.2.2.1 Add/Update Configuration

The feature provides a capability for admin users to define and update the configuration parameters of the FITMAN Specific Enablers. The parameters include database configurations, security constraints, server configurations or other business constraints as necessary. The configuration details are stored in the internal repository as support for enablers’ implementation.

![Sequence Diagram](image)

Figure 305 Add/Update Configuration Sequence Diagram

The UI to configure a Manufacturing Enablers is as follows:
6.2.2.2.2 List configuration
The feature provides a capability to list all configurations of a Manufacturing Enabler.

Figure 306 Add/Update Configuration UI Mockup
The UI to list configuration of a Manufacturing Enablers is as follows:

Figure 307 List configuration Sequence Diagram
Figure 308 List Configuration UI Mockup

6.2.2.2.3 Enabler APIs
6.2.2.2.4 Monitoring services
The feature provides performance details and usage statistics on Manufacturing Enabler’s services. These data are accessible from the API Manager and the admin could find some information like number of subscriptions for every version of the API, number of calls, latency Time etc.
The UI of the API Manager Monitoring services is as follows:
6.2.2.2.5 Enabler Qualification
The feature provides classification of Manufacturing Enablers in terms of exposed services, protocols, reliability in run-time, etc.

![Enabler Qualification Sequence Diagram](image-url)

The UI to get qualifications of a Manufacturing Enablers is as follows:
6.2.2.2.6 Enabler Adaptation
This feature provides a capability to add minor adaptations to improve the interoperability of exposed functionalities or services.
The UI to adapt the configuration of a Manufacturing Enablers is as follows:

Figure 315 Enabler Adaptation UI Mockup
6.2.2.3 Interaction description

Figure 316 Manufacturing Enablers Interaction Diagram

D2.2: Functional Specifications & Mockups - Vs: 1.0 - Public
6.2.3 FI-WARE Generic Enablers

FI-WARE Generic Enablers component is composed of various generic enablers that have been developed in the FI-WARE project, which provide diverse functionalities and have been developed by using diverse technologies. The functionalities provided by these enablers can be used for the realisation of some of the core features of vApps to
implement their business requirements. In the scope of vf-OS FIWARE enablers are of interest, also to reuse solutions in the domain that are not addressed specifically in vf-OS and the initial selection of the enablers with their functional descriptions will be provided in D3.1. In this section, we will provide the generic functionalities that will be covered by all the enablers. Note that for integration and uniform access purpose various functionalities from enablers must be accessed via the Enablers’ Framework component.

6.2.3.1 Behaviour and Functionality

Enablers Framework component provides a set of functionalities that could be grouped on the following features:

- **Configuration and Adaptation of Enablers**: Even though the enablers framework provides necessary functionalities for configuration of the enablers, it is important to have configuration related functionalities in the FI-WARE Generic Enablers. This is mainly because of the diverse nature of the enablers and the generic encapsulate configuration management in Enablers Framework might not be enough for all enablers. This feature will provide the IT admin user with the ability to configure different additional Generic Enabler specific parameters that need to be provided for the correct functioning of the enabler. While the adaptation feature is utilised for wrapping the functionalities of some of the enablers which do not have any standardised method invocation interfaces.

- **Service Invocation**: The functionalities provided by the enablers which are encapsulated into respective functional implementations are invoked by the vApps via Enablers Framework. This feature is thus the collection of various functionalities provided by enablers which can include methods providing both synchronous and asynchronous responses. Also, note that the methods of generic enablers might need to be wrapped during adaptation process to enable service invocations.

- **Enablers’ Performance Monitoring**: This feature is dedicated to monitoring the performance and tracing the errors during runtime of all the generic enablers. These metrics can play an important part for the developer in making the choice for one enabler over other and IT Admin to adopt necessary performance optimisation strategy.

Following, there is a story map where the main features, epics and user stories for the FIWARE Generic Enablers components have been identified (see Figure 318). Note that one important factor for different release is also set of fl-ware generic enablers that will be selected in the scope of vf-OS. Even though this will impact the set of functionalities that will be provided it is not quite possible to represent in story maps and hence is not depicted in the story map.
### Subtask Description

<table>
<thead>
<tr>
<th>Subtask</th>
<th>Description</th>
</tr>
</thead>
</table>
| **FGUS001. Set enabler’s configurations** | **Who:** vf-OS IT manager  
**What:** receive configurations from configuration’s interface in order to configure framework  
**Why:** so that it is possible to configure enabler  
**Acceptance Criteria**  
Enabler is configured |
| **FGUS002. List all enablers configurations** | **Who:** vf-OS IT manager  
**What:** collect configurations from configuration’s repository and return them to user’s interface  
**Why:** so that it is possible to return the configurations  
**Acceptance Criteria**  
Make sure all configuration validations are returned |
| **FGUS003. Adaptation Service** | **Who:** Generic Enabler  
**What:** Receive adaptations from adaptation’s user interface  
**Why:** in order to provide adaptations to improve the interoperability of exposed functionalities or services  
**Acceptance Criteria**  
Make sure all adaptations applied are valid and don’t affect the functionalities of the enabler |
<table>
<thead>
<tr>
<th>FGUS004. Invoke enabler functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: Generic Enabler</td>
<td></td>
</tr>
<tr>
<td>What: will invoke the method specified by vApp</td>
<td></td>
</tr>
<tr>
<td>Why: so that the functional implementation of the method can be invoked</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FGUS005. Send enabler’s response (synchronous)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: Generic Enabler</td>
<td></td>
</tr>
<tr>
<td>What: will have the synchronous response to the provided methods.</td>
<td></td>
</tr>
<tr>
<td>Why: so that all the methods providing synchronous request-response patterns can be invoked</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FGUS006. Send enabler’s response (Asynchronous)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: Generic Enabler</td>
<td></td>
</tr>
<tr>
<td>What: Enabler will provide response asynchronously.</td>
<td></td>
</tr>
<tr>
<td>Why: so that all the methods providing asynchronous request-response patterns can be invoked</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FGUS007. Provide performance details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who: Generic Enabler</td>
<td></td>
</tr>
<tr>
<td>What: Keep traces of performance metrics and errors</td>
<td></td>
</tr>
<tr>
<td>Why: so that the performance of generic enablers can be analysed so performance and errors can be traced and resolved.</td>
<td></td>
</tr>
</tbody>
</table>

Acceptance Criteria

The return code from the invocation service must be a success and can have response data. In case of error a proper error code and description is provided.

Acceptance Criteria

The response must be within the latency limit and acknowledgement is received.

Acceptance Criteria

The response is successfully delivered.

6.2.3.2 UI mockups and Sequence Diagrams

6.2.3.2.1 Manage Configurations

This feature provides the capability to manage configurations for the enablers framework component and enablers that are integrated into the framework.

The main steps/functionality are:

- Set configurations as required by the Generic Enabler
- List all the configurations that have been applied to a specified generic enabler

The associated sequence diagram is as shown in Figure 186.
The UI for the generic enablers are provided by themselves and are not very uniform to provide the UI mocks. They will be provided in detail in the scope of D3.2.

6.2.3.2.2 Service Invocation and Performance Tracing

This feature provides the functional implementations accessing the functionalities provided by Generic Enabler.

The main step/functionality is:

- Invoke the selected functionality provided by the generic enabler with necessary input parameters
- Support both synchronous and asynchronous types of request-response patterns
- Provide performance details to the enabler framework when the service invocation cycle ends

The associated sequence diagram is as shown in Figure 320. Note that during service invocation the generic enabler can access their repositories to get/put data when necessary (not shown in the sequence diagram because it might not be the case for all service invocations).
6.2.3.3 Interaction Description

Based on the description of the functionality covered by the FI-WARE Generic Enabler framework component we can observe a number of interactions that the component will have with other vf-OS components. Presented in this section is a detailed representation of interactions with other vf-OS components and also some internal interactions between sub-components. The following figure shows the flow of information between the internal subcomponents and other vf-OS components.
In order to clarify the interactions between components the main interactions of messaging component with other components are as explained below:

- Configurations and Adaptation management: This provides necessary interaction with the vf-OS storage component and is used for storing specific configuration.
details of enablers. Additionally, it also involves interactions for adaptation of enablers. The main information flows are:

- **Put/Get configuration details into/from the storage**
- **Provide and apply adaptation mechanism**

- **Functional Invocation:** This provides necessary interaction with the Enablers Framework component to access the specified functionality provided by the generic enabler. The main information flows are:
  
  - Enabler Framework sends a request to invoke a method of the enabler with necessary input parameters through the native interface of the generic enabler
  - Method is invoked and executed as defined in the functional implementation
  - Functional implementation can interact with the storage to put/get data
  - During the entire invocation, process performance is monitored and when it ends the performance details are provided to the Enablers Framework

---

**Figure 322 Classes of FI-WARE Generic Enablers with External Interactions and Model of Information Exchanged**
7 Compliance with software requirements

Describes the mapping between software requirements and provided functional features.

7.1 Software Requirements Mapping

Software requirement mapping to user stories is provided in an annexed Excel file. This Functional Specifications specify the scope of vf-OS, and it is a unique document presenting the functionality provided by the vf-OS components to end users of vf-OS and the interfaces these users will interact with. The functionality provided by any platform should address user needs and requirements. As such, this annex maps requirements of vf-OS identified on T1.5 to the functionality provided by the vf-OS components.

The methodology followed has been as follows: The T1.5 requirements where filtered according to their scope selecting only the vf-OS requirements (and not the vApps). Then were filtered only the functional requirements (as far as, non-functional requirements are within the scope of the T1.3 Technical Specification). Finally, only the requirements were selected with an importance 1 and 2, keeping out nice-to-have and make-up requirements that could drive complexity into the components without a valuable impact.

Those requirements have been mapped to user stories. User stories are the functionality identified in the story maps of the different components. Then, if there was a match between a filtered requirement and a user story, it can be confirmed that the vf-OS platform is addressing the requirements in a given way.

The following figure show a brief if the results generated. The full result is in an excel file.

<table>
<thead>
<tr>
<th>rid</th>
<th>actor</th>
<th>description</th>
<th>User Stories</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ_0004</td>
<td>vf-IO</td>
<td>vf-OS must provide means to connect the system to both virtual (software) and physical factory assets</td>
<td>DRUS010</td>
<td>Drivers</td>
</tr>
<tr>
<td>RQ_0010</td>
<td>vf-OAK</td>
<td>vf-OAK must provide a Front End Environment to generalise the UI of vApps</td>
<td>FEUS014</td>
<td>Frontend Environment</td>
</tr>
<tr>
<td>RQ_0017</td>
<td>vf-P</td>
<td>vf-OS OAK must provide security mechanisms on a role-model base on represent real world access rights</td>
<td>SCUS013</td>
<td>Security</td>
</tr>
<tr>
<td>RQ_0018</td>
<td>vf-SK</td>
<td>vf-SK must facilitate the creation of new enablers to allow the extension of the vf-OS ecosystem</td>
<td>EFUS003-008, EFUS12-15</td>
<td>Enablers Framework</td>
</tr>
<tr>
<td>RQ_0019</td>
<td>vf-SK</td>
<td>vf-SK must be able to fit existing FI-WARE generic enablers</td>
<td>EFUS003-008, EFUS12-15</td>
<td>Enablers Framework</td>
</tr>
<tr>
<td>RQ_0020</td>
<td>vf-SK</td>
<td>vf-SK must be able to run in any OS platform (eg Linux, Win, Mac OS)</td>
<td>DAUS021-DAUS027; DAUS001-DAUS004</td>
<td>Data Analytics</td>
</tr>
<tr>
<td>RQ_0022</td>
<td>vf-MW</td>
<td>vf-MM must provide offline data analytics methods with machine learning algorithms</td>
<td>DAUS010-DAUS011</td>
<td>Data Analytics</td>
</tr>
<tr>
<td>RQ_0023</td>
<td>vf-MW</td>
<td>vf-MM must provide a scalable data storage able of storing real-time data</td>
<td>DSUS010-DSUS117</td>
<td>Storage</td>
</tr>
</tbody>
</table>
8 Potential Risks and Open Issues

8.1 Potential Risks

Functional specifications are the main document where the functionality is described and they are the main document driving the development of the software solutions. They are used to be the “contractual” document where users can see the functionality that the software will provide to them and the kind of interfaces they will have available. As such, they have a lot of risks, because the technical specification and the development tasks can limit some of the functionality that was agreed and can change and restrict the interfaces agreed. Correspondingly the consortium will carefully monitor this possible misalignment and other issues during the technical work packages (WP3 to WP7) and during the T2.3 Technical Specifications.

The following table depicts some early risks detected and how the consortium intends to handle them during the project. The main risks were collaboratively identified during consortium discussions.

<table>
<thead>
<tr>
<th>Risk</th>
<th>Description</th>
<th>Contingency Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Risks</td>
<td>The selected technology for creating the components could not allow implementation of all aspects of the security component</td>
<td>A specific and separate technology from the rest of the components will be selected for implementing all security features</td>
</tr>
<tr>
<td></td>
<td>Malware can be developed with the platform</td>
<td>All developers must be authenticated and located. The code will be scanned by using proven and standard security technologies</td>
</tr>
<tr>
<td>Technical and development misalignment</td>
<td>It could be difficult to dynamically inspect message content within Pub/Sub functionality</td>
<td>A specific technology will be selected that allows the inspection of content for the creation of Pub/Sub topics and/or channels respecting security and privacy concerns</td>
</tr>
<tr>
<td></td>
<td>Functional Specification can see their interfaces and functionality compromised by the technical availability and limitations</td>
<td>The technical specification has to consider the functional specification when selecting libraries and providing APIs. The different technical WPs (from 3 to 7) will need to consider the functional specification when designing and developing the final software solution</td>
</tr>
<tr>
<td>Integration complexity</td>
<td>The different components identify and their functionality provided are the result of the coordination among vf-OS components. Additionally, the development language of each component can be different</td>
<td>The functional specification has made an effort detailing in the interaction among components and specification the structure of the information exchanged. That coordination should be managed carefully on the technical WP according to the contracts identified. Addressing the Multilanguage development of vf-OS, REST services will be provided in T.2.3</td>
</tr>
<tr>
<td>Platform Unique Coordination</td>
<td>Every vf-OS component provides management capabilities that accessible from a unique end-point, the Platform component that forces the component to have a unique style and coherent functionality and usability</td>
<td>Technical Specification and Technical Work packages will need to coordinate their management functionality in conjunction and together with the Platform component</td>
</tr>
</tbody>
</table>
8.2 Open Issues

Although the intensive work carried out by the vf-OS consortium has concluded in the current functional specification, there are still some open issues/uncertainties that need to be resolved by the Technical Specifications. The following table (Figure 324) identifies and summarises these open issues allowing an easy track and resolution procedure.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Next Steps</th>
<th>Lead (Rationale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage of the Messaging component</td>
<td>The I/O Toolkit components should not directly communicate with other components, eg Storage or Transformation, but utilise the Messaging component instead. This way, there will be a single central point for defining all data flow</td>
<td>I/O Toolkit components to discuss with other</td>
<td>Messaging: This data flow must be clear in the project</td>
</tr>
<tr>
<td>Security at the Platform</td>
<td>From the diagrams of this document, it can be deduced that there is no security foreseen on the execution side. Currently, this is being seen as an intrinsic part of the components that are being executed. From a security point of view, if an interaction between two (or more) components is not declared, then the Security component will not allow such interaction</td>
<td>Platform, and Security to discuss how to address this issue</td>
<td>Platform: The Security component has to be informed about interactions between components to allow them</td>
</tr>
<tr>
<td>Permissions within vApps or vf-OS Assets</td>
<td>Upon design/runtime/deployment use, the individual vf-OS Asset or Component that needs to connect with the access control list to determine what is possible for that user. However, an application cannot come with a manifest of people who can use it and the roles they have so it is a mixture of both</td>
<td>Security, and Platform to discuss how to address this issue</td>
<td>Security: the vApp is the one that has permissions, not the user.</td>
</tr>
</tbody>
</table>

Figure 324: Open Issues/Uncertainties Identified
9 FAQs

This section contains a list of FAQs raised during the technical discussions for future reference and clarity. These FAQs will be extended in the next documents.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why don’t all of the sequence diagrams have mockups associated?</td>
<td>Only the sequence diagrams where user interfaces are involved have mockups associated. It is usually the case of functionality launched by users’ interaction with the software system. In case of functionality invoked by another component, that is, initiated automatically by another piece of software, there will be no UI present and therefore no mockup associated.</td>
</tr>
<tr>
<td>Why are there no information models detailing information structure within each component?</td>
<td>The information models provided in the present document are representing the structure of the information exchanged between the different components in vf-OS. The information models detailing the internals of the components are detailed in each specific technical workpackage along with the implementation.</td>
</tr>
</tbody>
</table>

Figure 325: FAQs
10 Conclusions

This document has presented the vf-OS functional specification, and therefore is the main specification to understand the functionality provided by vf-OS and delimiting its scope. The document is a core to:

- Understand what is offered or not in the scope of the vf-OS platform
- How the functionality assures the vf-OS will satisfy the requirements agreed with different vf-OS stakeholders
- How the functionality will be carried out through the aggregation of functionality provided by different vf-OS components composing the vf-OS platform.
- What will be the user interface (UI) experience that the different vf-OS users will have when interacting with the software
- How the D2.1 Global architecture is concretised in specific functionality
- What are the features and main components that developers must address and develop when coding the different components in their specific technical workpackages (that is, WP3 to WP7)

The document has followed an agile methodology for introducing the different explanation of components and baselines of functionality have been defined, defining what functionality will be developed in each iteration of the vf-OS environment.

This document is complemented by D2.3 Technical Specifications vf-OS.
# Annex A: History

## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1:</td>
<td>First Draft produced by Editor</td>
</tr>
<tr>
<td>V0.2:</td>
<td>Modification of structure of the document</td>
</tr>
<tr>
<td>V0.3:</td>
<td>Drivers story maps</td>
</tr>
<tr>
<td>V0.4:</td>
<td>Drivers sequence and mockups</td>
</tr>
<tr>
<td>V0.5:</td>
<td>Drivers Interaction diagrams</td>
</tr>
<tr>
<td>V0.6:</td>
<td>Comments on content</td>
</tr>
<tr>
<td>V0.7:</td>
<td>Written section 0, 1 and 2</td>
</tr>
<tr>
<td></td>
<td>First sketch of section 7, 8 and 9</td>
</tr>
<tr>
<td>V0.10:</td>
<td>Unification of sections</td>
</tr>
<tr>
<td></td>
<td>Executive summary and conclusions</td>
</tr>
<tr>
<td>V0.12:</td>
<td>1&lt;sup&gt;st&lt;/sup&gt; Internal review</td>
</tr>
<tr>
<td>V0.13:</td>
<td>Amendments after 1&lt;sup&gt;st&lt;/sup&gt; internal review</td>
</tr>
<tr>
<td></td>
<td>External Service Provision, System Dashboard, Platform, OAK SDK, OAK Studio, Developer Engagement Hub</td>
</tr>
<tr>
<td>V0.14:</td>
<td>Review by coordinator</td>
</tr>
<tr>
<td>V0.15:</td>
<td>Amendment after review by coordinator</td>
</tr>
<tr>
<td>V1.0:</td>
<td>Version submitted to EC</td>
</tr>
</tbody>
</table>

## Versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1:</td>
<td>First Draft produced by Editor</td>
</tr>
<tr>
<td>V0.2:</td>
<td>Modification of structure of the document</td>
</tr>
<tr>
<td>V0.3:</td>
<td>Drivers story maps</td>
</tr>
<tr>
<td>V0.4:</td>
<td>Drivers sequence and mockups</td>
</tr>
<tr>
<td>V0.5:</td>
<td>Drivers Interaction diagrams</td>
</tr>
<tr>
<td>V0.6:</td>
<td>Comments on content</td>
</tr>
<tr>
<td>V0.7:</td>
<td>Written section 0, 1 and 2</td>
</tr>
<tr>
<td></td>
<td>First sketch of section 7, 8 and 9</td>
</tr>
<tr>
<td>V0.10:</td>
<td>Unification of sections</td>
</tr>
<tr>
<td></td>
<td>Executive summary and conclusions</td>
</tr>
<tr>
<td>V0.12:</td>
<td>1&lt;sup&gt;st&lt;/sup&gt; Internal review</td>
</tr>
<tr>
<td>V0.13:</td>
<td>Amendments after 1&lt;sup&gt;st&lt;/sup&gt; internal review</td>
</tr>
<tr>
<td></td>
<td>External Service Provision, System Dashboard, Platform, OAK SDK, OAK Studio, Developer Engagement Hub</td>
</tr>
<tr>
<td>V0.14:</td>
<td>Review by coordinator</td>
</tr>
<tr>
<td>V0.15:</td>
<td>Amendment after review by coordinator</td>
</tr>
<tr>
<td>V1.0:</td>
<td>Version submitted to EC</td>
</tr>
</tbody>
</table>

## Contributions

**ICE:**
- Oscar García – Input to sections 4.1.4, 4.1.5, 5.1.1, 5.2.2, 5.2.3 and Annex C
- Stuart Campbell – Input to sections 4.1.4, 5.1.1, 5.2.2 and coordinator internal review
- Rebecca Campbell – Internal English review

**IKERLAN:**
- Oskar Saiz - Input to Section 5.2.1 and Annex C
- José Luis Flores - Input to Section 5.2.1 and Annex C

**UNINOVA:**
- Sudeep Ghimire - Input to Sections 5.1.2, 5.1.3, 5.3.1, 6.2.3 and Annex C

**UPV:**
- Raul Poler – Main Editor. First draft and input to all sections
- Víctor Anaya – Input to sections 1, 2, 5.3.2, 7, 8 and 10
- Francisco Fraile - Input to sections 2, 5.3.2 and Annex C

**CMS:**
- Carlos Coutinho - Input to Section 4.1.1, 4.1.2, and 4.2

**LYON2:**
- Néjib Moalla - Input to Sections 5.3.3, 6.2.2 and Annex C

**ASCORA:**
- Danny Pape - Input to Sections 4.1.3, 7 and Annex C
- Tobias Hinz - Input to Section 6.1.1 and Annex C

**ALMENDE:**
<table>
<thead>
<tr>
<th>KBZ:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Andries Stam - Input to Sections 3.1, 5.3.4, and 5.4.1</td>
</tr>
<tr>
<td>• David Aleixo - Input to Sections 6.2.1 and Annex C</td>
</tr>
</tbody>
</table>
Annex B: References

None