vf-OS: virtual factory Operating System

WP5: Virtual Factory Data and Connect

D5.1a: WP5 Umbrella Deliverable - Vs: 1.0

Deliverable Lead and Editor: Oscar Garcia, ICE
Contributing Partners: ICE, UNINOVA, IKER
Date: 2018-03
Dissemination: Public
Status: Consortium Approved

Short Abstract
This deliverable is the WP5 –Virtual Factory Data and Connect Umbrella Document. It provides the documentation of activities and progress of WP5 as defined in the vf-OS DOA. All of the tasks of WP5 (T5.1-T5.4) will produce software packages as deliverables hence, this document will be the base for understanding the development and availability status of different vf-OS components that constitute the virtual factory Middleware (vf-MW).
Document Status

<table>
<thead>
<tr>
<th>Deliverable Lead</th>
<th>Oscar Garcia, ICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Reviewer 1</td>
<td>Stuart Campbell, ICE</td>
</tr>
<tr>
<td>Type</td>
<td>Deliverable</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP5: Virtual Factory Data and Connect</td>
</tr>
<tr>
<td>ID</td>
<td>D5.1a: WP5 Umbrella Deliverable</td>
</tr>
<tr>
<td>Due Date</td>
<td>2018-03</td>
</tr>
<tr>
<td>Delivery Date</td>
<td>2018-03</td>
</tr>
<tr>
<td>Status</td>
<td>Consortium Approved</td>
</tr>
</tbody>
</table>

History
See Annex A.

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.
Project Partners:
Executive Summary

The main objective of the “WP5: Virtual Factory Data and Connect” is to create the v-OS data infrastructure including interoperability connectors, storage, transformation services, and analytics functionality. The deliverables of this work package are divided into software packages and document/reports. In terms of reporting:

- All software deliverables (type “OTHER”) are available from the vf-OS repository with access details in Section 0 and install instructions in the corresponding annex of this document. The outputs are software components: Data Infrastructure Middleware, Data Storage, Data Harmonisation, and Data Analytics.
- Since this work package has all its deliverables of type “OTHER” this Umbrella document, as identified in the DOA, reports the work accomplished in those deliverables.

WP5 has three deadlines for deliverables submission, which means that this document will also have three iterations, M18, M24, and M30. Consequently, this document is structured in four main sections in direct relation to the four WP5 tasks. Whilst from the reporting perspective each one describes planned activities, the progress, and the next activities for each of the mentioned periods, from the software perspective an annex is included to describe the software functionalities and provide pointers to the location of the software developed so far.

In this period, M12-M18, this document reports about the following tasks:

- T5.1 - Data Infrastructure Middleware
- T5.2 - Data Storage
- T5.3 - Data Harmonisation
- T5.4 - Data Analytics
# Table of Contents

0. Introduction................................................................................................................................. 1

1. Context ......................................................................................................................................... 5
   1.1 Positioning in overall vf-OS Architecture .............................................................................. 5
   1.2 Purpose of the vf-MW Components ......................................................................................... 6
   1.3 Implementation Strategy .......................................................................................................... 7
   1.4 Development Roadmaps ......................................................................................................... 7
      1.4.1 Data Infrastructure Middleware – Development Roadmap ........................................... 8
      1.4.2 Data Storage – Development Roadmap .......................................................................... 9
      1.4.3 Data Harmonisation – Development Roadmap .............................................................. 11
      1.4.4 Data Analytics – Development Roadmap ......................................................................... 12

2. Data Infrastructure Middleware (T5.1) ..................................................................................... 15
   2.1 Scope .................................................................................................................................... 15
   2.2 M18 Report ........................................................................................................................... 15
      2.2.1 Planned Activities ........................................................................................................... 15
      2.2.2 Progress .......................................................................................................................... 15
      2.2.3 Next Activities .................................................................................................................. 16
   2.3 M24 Report ........................................................................................................................... 16
   2.4 M30 Report ........................................................................................................................... 16

3. Data Storage (T5.2) ............................................................................................................... 17
   3.1 Scope .................................................................................................................................... 17
   3.2 M18 Report ........................................................................................................................... 17
      3.2.1 Planned Activities ........................................................................................................... 17
      3.2.2 Progress .......................................................................................................................... 17
      3.2.3 Next Activities .................................................................................................................. 18
   3.3 M24 Report ........................................................................................................................... 18
   3.4 M30 Report ........................................................................................................................... 18

4. Data Harmonisation (T5.3) ....................................................................................................... 19
   4.1 Scope .................................................................................................................................... 19
   4.2 M18 Report ........................................................................................................................... 19
      4.2.1 Planned Activities ........................................................................................................... 19
      4.2.2 Progress .......................................................................................................................... 19
      4.2.3 Next Activities .................................................................................................................. 20
   4.3 M24 Report ........................................................................................................................... 20
   4.4 M30 Report ........................................................................................................................... 20

5. Data Analytics (T5.4) ............................................................................................................... 21
   5.1 Scope .................................................................................................................................... 21
   5.2 M18 Report ........................................................................................................................... 21
      5.2.1 Planned Activities ........................................................................................................... 21
      5.2.2 Progress .......................................................................................................................... 22
      5.2.3 Next Activities .................................................................................................................. 23
   5.3 M24 Report ........................................................................................................................... 23
   5.4 M30 Report ........................................................................................................................... 23

6. Conclusions ............................................................................................................................. 24

Annex A: History ............................................................................................................................ 25
Annex B: References ....................................................................................................................... 26
Annex C: Data Infrastructure Middleware R1 Software Packages .............................................. 27
C.1. Messaging Software Package ............................................................................................... 27
C.1.1. Messaging Requirements ................................................................. 27
C.1.2. Messaging Installation ................................................................. 27
C.1.3. Messaging Execution ................................................................. 27
C.2. Pub/Sub Software Package ............................................................... 27
   C.2.1. Pub/Sub Requirements .............................................................. 28
   C.2.2. Pub/Sub Installation ................................................................. 28
   C.2.3. Pub/Sub Execution ................................................................. 28
Annex D: Data Storage R1 Software Packages .............................................. 29
   D.1. Relational Storage Software Package .................................................. 29
      D.1.1. Relational Storage Requirements .................................................. 29
      D.1.2. Relational Storage Installation .................................................... 29
      D.1.3. Relational Storage Execution ..................................................... 30
   D.2. Document-oriented Storage Software Package ................................. 30
   D.3. Time-series Storage Software Package ............................................ 30
   D.4. Triple-store Storage Software Package ............................................ 30
Annex E: Data Harmonisation R1 Software Packages .................................... 31
   E.1. Data Mapping Software Package ....................................................... 31
      E.1.1. Data Mapper Requirements ......................................................... 31
      E.1.2. Data Mapper Installation ............................................................. 31
      E.1.3. Data Mapper Execution ............................................................... 32
   E.2. Data Transformation Software Package ............................................ 41
Annex F: Data Analytics R1 Software Packages ............................................ 42
   F.1. Historical Data Analytics Software Package ....................................... 42
      F.1.1. Historical Data Analytics Requirements ....................................... 42
      F.1.2. Historical Data Analytics Installation .......................................... 42
      F.1.3. Historical Data Analytics Execution ............................................ 42
   F.2. Stream Data Analytics Software Package ......................................... 43
      F.2.1. Stream Data Analytics Requirements .......................................... 43
      F.2.2. Stream Data Analytics Installation ............................................. 43
      F.2.3. Stream Data Analytics Execution .............................................. 43
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 of 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 Logistic Users** which will explore the marketplace for already created solutions, ready to be run on the vf-OS
- **Manufacturing and Logistics Solutions 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. 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 manufacturing stages 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 “D5.1 WP5 Umbrella Deliverable” is to provide a centralised report for all the activities in the scope of WP5. The deliverables in WP5 tasks are software packages; hence this deliverable provides the overview of planned activities, progress and plans for different reporting periods, ie M18, M24, and M30. The main purposes of the deliverable are:

- Provide insight into the status and progress of development tasks of WP5
- Provide insight into the developed code, release cycles and pinpoint links to the source code, binary releases and documentation associated with the software packages.

0.3 Target Audience

Whilst primarily aimed at the project partners, this public deliverable can be useful for the scientific and industrial community. This includes other publicly funded projects, which may be interested in collaboration activities.

0.4 Deliverable Context

This document is reports across all the activities of WP5 and is mainly focused on reporting activities. However, the activities in WP5 are dependent on various other deliverables. Hence, the relationship of this deliverable to other deliverables is as follows:

- **SoA on Existing Solution (D1.4):** Wiki for providing current state of relevant technologies, including cloud computing, IOT, embedded systems, sensors and CPS components, data analytics, Big Data sources, and mobile data sources. Available now
- **Requirements Specification (D1.5):** Report for documenting the requirements of vf-OS divided into strategic, high level and technical requirements. Available now
- **Global Architecture Definition (D2.1):** Report providing the high-level global architecture of vf-OS with all the first level modules and different interactions, both at service calls and data exchange. Available now
- **Functional Specification & Mock-ups (D2.2):** Report for providing an in-depth definition of the functionalities and behaviour of all vf-OS components. Available now
- **Technical Specification (D2.3):** Report for providing an in-depth definition of the concrete interfaces between all vf-OS software components, protocols and
class/package structures, including definitions of methods, parameters, return values, and error handling for each component and interface. Available now

- **Holistic Security and Privacy Concept (D2.4):** Report for identifying suitable security technology and standards to be applied. Concrete guidelines have been generated which need to be taken into account throughout the vf-OS software development tasks. Available now

### 0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1:** Context: This section provides the overall positioning of the virtual factory Middleware (vf-MW) in the context of vf-OS technical architecture
- **Section 2:** Data Infrastructure Middleware (T5.1): This section provides the report for task T5.1 for its reporting periods M18, M24 and M30
- **Section 3:** Data Storage (T5.2): This section provides the report for task T5.2 for its reporting periods M18, M24 and M30
- **Section 4:** Data Harmonisation (T5.3): This section provides the report for task T5.3 for its reporting periods M18, M24 and M30
- **Section 5:** Data Analytics (T5.4): This section provides the report for task T5.4 for its reporting periods M18, M24 and M30
- **Section 6:** Conclusions: Conclusions of the document

- **Annexes:**
  - Annex A: Document History
  - Annex B: References
  - Annex C: Data Infrastructure Middleware R1 Software Packages
  - Annex D: Data Storage R1 Software Packages
  - Annex E: Data Harmonisation R1 Software Packages
  - Annex F: Data Analytics R1 Software Packages

### 0.6 Document Status

This document is Public since all of its related deliverables are listed in the Description of Action as “Public”.

### 0.7 Document Dependencies

This document is part of an iteration of living deliverables. This is the first iteration and covers the M18 reports for each of the other sections from 2 – 5.

### 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

All software deliverables (type “OTHER”) are available from the vf-OS repository with read-only access details as follows:

- URL: https://owncloud.ascora.de/index.php/s/smJN0KDHyEwjp9a
- Password: 123vfOS

0.10 Reading Notes

None
1 Context

vf-OS intends to be the reference system/platform for providing common services for the manufacturing industry. Similarly, to a regular OS, vf-OS has core functionalities but is mainly focused in a manufacturing environment. An analogy between some components of a regular Software OS and the vf-OS Manufacturing OS is presented in Figure 1 (as introduced in the vf-OS D1.1 Vision Consensus).

<table>
<thead>
<tr>
<th>Software OS Environment</th>
<th>Virtual Factory OS (vf-OS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Sub-component</td>
</tr>
<tr>
<td>Kernel</td>
<td>Processor, Memory, Internal Bus</td>
</tr>
<tr>
<td>I/O</td>
<td>Interfaces, Peripherals, Device Drivers, APIs</td>
</tr>
<tr>
<td>File and Data Handling</td>
<td>Interfaces</td>
</tr>
<tr>
<td>SDK</td>
<td>SDK, Application Development portal, System Monitor</td>
</tr>
<tr>
<td>Applications</td>
<td>ERPs, CRMs, MES</td>
</tr>
<tr>
<td>Environment</td>
<td>Developer Environment, App Store, Internet and Service Provision</td>
</tr>
</tbody>
</table>

The vf-OS Middleware (vf-MW), thus, represents the elements dealing with the data integration, processing, storing and analysing.

1.1 Positioning in overall vf-OS Architecture

The virtual factory Middleware (vf-MW) is one of the components of the Open Operating System for Virtual Factories as referenced in D2.1: Global Architecture. Technically, vf-MW is not one single software component but a collection of different vf-OS components specifically those that are developed in the scope of WP5.

Figure 2 is an overview of the high-level architecture of vf-OS providing a formal split of functional components, as already described in D2.1. As vf-MW is particularly focused on providing data infrastructure including interoperability connectors, storage, transformation services, and analytics functionality, this is achieved through the development of six key components, according to the Figure 2 below.
Thus, the vf-MW is formed of six components belonging to the following architectural building blocks:

- **Design time: Application-Development**: This block comprises the different vf-OS components that will be used for the development of the vf-OS Manufacturing Assets/vApps. Of particular interest is the Data Mapping, which is a component comprised within the Data Harmonisation task and which features are useful for analysts to carry out relevant activities when developing an Asset/vApp.

- **Runtime: Application Services & Middleware**: This block includes the vf-OS components that will be used by the Assets/vApps when they are executing or when they are trying to execute some of the vf-OS resources. Of the four modules that the Runtime is divided into, only two are the focus of the vf-MW, these being the Middleware and the Data Management modules. Whilst within the Middleware only T5.1 is involved by developing the PubSub and the Messaging components, within the Data Management module the other three tasks of WP5 are involved dealing with the Storage (T5.2), Transformation (T5.3) and Analysis (T5.4).

### 1.2 Purpose of the vf-MW Components

It is important to define the purpose of the vf-MW components to set up the scene where these components are being developed:

- **OAK Toolkit: Data Mapping**: To define the Manufacturing Maps that will allow the transformation and integration of data
• **Middleware: Messaging:** To provide a unified infrastructure for data (message) transport between vf-OS components

• **Middleware: Publish/Subscribe:** To provide a communication infrastructure based on a publish/subscribe pattern by creating topics or by dynamically inspecting message content to enable listening applications to subscribe to specific messages

• **Data Management: Data Storage:** To implement a scalable data storage system, capable of handling real-time sensor data and events, based on an underlying infrastructure that transparently absorbs very large amounts of data, as well as other types of non-real-time heterogeneous data

• **Data Management: Data Transformation:** 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

• **Data Management: Data Analytics:** To provide business analytic functionalities to derive events from stream and historic process data within the manufacturing domain

### 1.3 Implementation Strategy

The implementation strategy for the vf-MW is based on the principles of software reusability, ie “Do not reinvent the wheel” approach. As such, various components of the vf-MW are based on existing tools which need certain vf-OS specific features and/or improvements.

Planned reuse of software components can improve the productivity and quality in software development. This deliverable thus identifies the functional characteristics of the selected tools, as part of an extended exercise of what is presented in section 9 of D2.1. These characteristics are all guided by common requirements from different vf-OS requirements, whether they have been expressed by the industrial partners or not, where the development strategy can follow reusing existing open source solutions to extend their functionalities as required by vf-OS pilots.

Since vf-MW is not a unique software package but a distributable, installable, and deployable bundle of software components, each component will follow its own implementation strategy. As an example, the Data Mapping will be delivered as a standalone application, while the Data Transformation and the Data Analytics will be delivered as standalone libraries that will be published within the vf-Store. As a conclusion, the vf-MW components' implementation strategies have already been defined within D2.1 with a necessary selection of technologies.

### 1.4 Development Roadmaps

The development tasks performed in WP5 are based on the functional specification detailed in D2.2: Functional Specification & Mock-ups through user stories. The following sub-sections present the initial release plan and corresponding user stories for each of the software components in tasks T5.1 – T5.4.

The remainder of the document reviews the activities planned, achieved, and forecasted for the rest of the deliverable as follows:

- **Section 2:** Data Infrastructure Middleware (T5.1): This section provides the report for task T5.1 for its reporting periods M18, M24 and M30

- **Section 3:** Data Storage (T5.2): This section provides the report for task T5.2 for its reporting periods M18, M24 and M30
• **Section 4:** Data Harmonisation (T5.3): This section provides the report for task T5.3 for its reporting periods M18, M24 and M30

• **Section 5:** Data Analytics (T5.4): This section provides the report for task T5.4 for its reporting periods M18, M24 and M30

### 1.4.1 Data Infrastructure Middleware – Development Roadmap

The Data Infrastructure Middleware is divided into two software components: Messaging and Pub/Sub. These will be developed with three major release milestones, i.e., R1 (M18), R2 (M24), and R3 (M30). Release R1 will provide the first version for the core functionalities to send and receive messages, which will be enhanced with additional advanced functionalities in R2 and learns from the initial phase of implementation for the other components of WP5. In reference to the user scenarios provided in the vf-OS deliverable D2.2, the summary of R1 is:

#### Messaging
- MSUS001: List existing configurations
- MSUS002: Add/delete/update configuration(s)
- MSUS005: Determine message destination(s)
- MSUS006: Route messages to one or more destinations
- MSUS011: RESTful API to messaging using HTTP
- MSUS014: Historical storage of messages
- MSUS016: Track and log message flows and related performance parameters
- MSUS017: Notify events or errors

#### Pub/Sub
- PSUS004: Register publisher
- PSUS005: Unregister publisher
- PSUS006: Notify un-registration of publishers to subscribers
- PSUS007: Discover available messages for subscription
- PSUS008: Subscribe to messages
- PSUS009: Unsubscribe to messages
- PSUS011: Publish messages by message producers
- PSUS012: Send data to subscribers

The user stories for R2 are:

#### Messaging
- MSUS003: List/add/update/delete Business Rules
- MSUS007: Create communication channel based on the requirement of the receiver
- MSUS008: Perform message aggregation/decomposition
- MSUS010: Queuing and precedence management of messages
- MSUS018: Provide periodic performance report

#### Pub/Sub
- PSUS010: Notify availability of new messages available for subscription
- PSUS013: Classify messages based on contextual information

Finally, the user stories for R3 are:
• Messaging
  • MSUS004: Add advanced configurations
  • MSUS009: Perform data filtering and quality check
  • MSUS012: Messaging of continuous real-time message
  • MSUS013: Transactional messaging
  • MSUS015: Encryption and Decryption of the messages being persisted in an external repository

1.4.2 Data Storage – Development Roadmap

The Data Storage task has three iterations for M18, M24 and M30, and has the objective of implementing 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. The development is planned in three major release milestones ie R1 (M18), R2 (M24), and R3 (M30), each of them aiming at a storage type. Release R1 will provide Relational Storage, release R2 will provide the Document Oriented Storage and the Triple Store while release R3 will provide the Time Series Storage. In reference to the user scenarios provided in vf-OS deliverable D2.2, the summary of R1 is:

• DSUS001: Create SQL Database
• DSUS002: Drop SQL Database
• DSUS003: Describe SQL Database
• DSUS004: Catalog of SQL Database tables
• DSUS005: Catalog of SQL Database views
• DSUS006: Create SQL Table
• DSUS007: Drop SQL Table
• DSUS008: Alter SQL Table
• DSUS009: Create SQL View
• DSUS010: Drop SQL View
• DSUS011: Replace SQL View
• DSUS012: Create SQL Index
• DSUS013: Drop SQL Index
• DSUS014: Describe SQL Table
• DSUS015: Describe SQL View
• DSUS016: Insert Entity
• DSUS017: Delete Entity
• DSUS018: Merge entity
• DSUS019: Update entity
• DSUS020: Query entity
• DSUS021: Create role
• DSUS022: Drop role
• DSUS023: Query roles
• DSUS024: AddRoleMember
• DSUS025: DeleteRoleMember
• DSUS026: Create user
• DSUS027: Drop user
• DSUS028: Query users
• DSUS029: Grant permission
• DSUS030: Revoke permission
The user stories for R2 are:

- DSUS101: Create TimeSeries Database
- DSUS102: Drop TimeSeries Database
- DSUS103: List Metric names
- DSUS104: List tag names
- DSUS105: List tag values
- DSUS106: Delete Metric
- DSUS107: Insert DataPoint
- DSUS108: Delete DataPoint
- DSUS109: Query Metrics
- DSUS110: Query Metric Tags
- DSUS111: Create Rollup Task
- DSUS112: List Rollup Tasks
- DSUS113: Get Rollup Task
- DSUS114: Delete Rollup Tasks
- DSUS115: Update Rollup Tasks
- DSUS116: Create user
- DSUS117: Drop user

The user stories for R3 are:

- DSUS201: Create NoSQL Database
- DSUS202: Drop NoSQL Database
- DSUS203: Describe NoSQL Database
- DSUS204: Catalog of NoSQL Database tables
- DSUS206: Create NoSQL Table
- DSUS207: Drop NoSQL Table
- DSUS208: Alter NoSQL Table
- DSUS209: Create NoSQL View
- DSUS210: Drop NoSQL View
- DSUS211: Replace NoSQL View
- DSUS212: Create NoSQL Index
- DSUS213: Drop NoSQL Index
- DSUS214: Describe NoSQL Table
- DSUS215: Describe NoSQL View
- DSUS216: Insert Entity
- DSUS217: Delete Entity
- DSUS218: Merge entity
- DSUS220: Query entity
- DSUS221: Create role
- DSUS222: Drop role
- DSUS223: Query roles
- DSUS224: AddRoleMember
- DSUS225: DeleteRoleMember
- DSUS226: Create user
- DSUS227: Drop user
1.4.3 Data Harmonisation – Development Roadmap

The Data Harmonisation task has three iterations for M18, M24 and M30 with three major release milestones ie R1 (M18), R2 (M24), and R3 (M30). It has the objective to provide the means for developing and executing Manufacturing Maps. Each of these phases is covered by the two components of the task: Data Mapping and Data Transformation. While release R1 will provide the basic functionality of Data Mapping, release R2 will finish off this component. Then, Data Transformation will be kicked off in release R2 and finished off at the end of R3. In reference to the user scenarios provided in vf-OS deliverable D2.2, the summary of R1 is:

- **Data Mapping**
  - DMUS001: Connect to Datasource
  - DMUS002: Read XML
  - DMUS003: Read CSV
  - DMUS004: Read JSON
  - DMUS005: Read TXT
  - DMUS006: Read XLS
  - DMUS008: Display UI
  - DMUS009: Load Source Schema
  - DMUS010: Display Source Schema
  - DMUS011: Analyse Source Schema
  - DMUS014: Display UI
  - DMUS015: Load Target Schema
  - DMUS016: Display Target Schema
  - DMUS017: Analyse Target Schema
  - DMUS020: Connect to Storage
  - DMUS021: Read file
  - DMUS022: Connect to Storage
  - DMUS023: Search Map
  - DMUS026: Connect to Storage
  - DMUS027: Annotate Map
  - DMUS028: Serialise Map
  - DMUS029: Persist Map
  - DMUS030: Connect to Storage
  - DMUS031: Search Map
  - DMUS032: Delete Map
  - DMUS038: Get Linked Concepts
  - DMUS039: Get Link
  - DMUS040: Is Linked
  - DMUS041: Add Linked Concept
  - DMUS042: Add Link
  - DMUS045: Get Concept
  - DMUS046: Add Concept

The user stories for R2 are:
• **Data Mapping**
  • DMUS007: Read MySQL
  • DMUS012: Connect to ontology
  • DMUS013: Suggest Semantic Concepts for Source Schema
  • DMUS018: Connect to ontology
  • DMUS019: Suggest Semantic Concepts for Target Schema
  • DMUS024: Filtering in Storage
  • DMUS025: Preview Map
  • DMUS033: Annotate Service
  • DMUS034: Create Service
  • DMUS035: Deploy Service
  • DMUS036: Connect to vf-Store
  • DMUS037: Publish Deployed Map
  • DMUS043: Update Linked Concept
  • DMUS044: Update Link
  • DMUS047: Update Concept
  • DMUS048: Delete Concept
  • DMUS049: Reasoning

• **Data Transformation**
  • DTUS001: Get invocation
  • DTUS002: Connect to vf-Store
  • DTUS003: Unpack Routines
  • DTUS004: Read Data

The user stories for R3 are:

• **Data Transformation**
  • DTUS005: Transform
  • DTUS006: Push Transformed Data
  • DTUS007: Store Transformed Data
  • DTUS008: Submit Usage Data

1.4.4 **Data Analytics – Development Roadmap**

The Data Analytics task has three iterations for M18, M24, and M30, and its component provides the features for deriving events from stream and historic process data within the manufacturing domain. The development is planned in three major release milestones ie R1 (M18), R2 (M24), and R3 (M30). Release R1 will provide the first version of the functional solution of the Data Analytics, which will be enhanced with additional advanced functionalities in R2 and R3. In reference to the user scenarios provided in D2.2, the summary of R1 is:

• DAUS001: Create analytics
• DAUS002: Read analytics
• DAUS005: Create query
• DAUS006: Read query
• DAUS007: Execute query
• DAUS008: Specify type of query (eg OLAP, Data Extraction...)
• DAUS009: Visualise query
• DAUS012: Select type of graph
• DAUS013: Select parameters
• DAUS014: Select time span
• DAUS016: Select dataset
• DAUS017: Select regression type (linear, least squares, polynomial...)
• DAUS018: Select grade of equation (in case of polynomial)
• DAUS019: Show results of regression
• DAUS021: Select dataset
• DAUS022: Split dataset in train and test
• DAUS023: Select ML method (eg Random Forest, Decision Tree...)
• DAUS024: Train ML method
• DAUS025: Test ML method
• DAUS026: Show results of ML
• DAUS028: Select dataset
• DAUS029: Split dataset in train and test
• DAUS035: Connect to stream source
• DAUS036: Connect to vf-OS Storage
• DAUS037: ETL
• DAUS038: Connect to batch source
• DAUS039: ETL
• DAUS040: Create an analysis module
• DAUS041: Read an analysis module
• DAUS042: Get analyses modules
• DAUS047: Get stream data
• DAUS048: Subscribe to stream data
• DAUS051: Create threshold
• DAUS052: Get thresholds
• DAUS053: Create sentence
• DAUS054: Get sentences
• DAUS061: Execute an active analysis
• DAUS063: Publish Alerts
• DAUS064: Publish Filtered data
• DAUS065: Persists Alerts
• DAUS066: Get alerts
• DAUS067: Update an alert status

The user stories for R2 are:
• DAUS003: Update analytics
• DAUS004: Delete analytics
• DAUS010: Update query
• DAUS011: Delete query
• DAUS015: Visualise graph
• DAUS030: Select Classification method (eg K-means, KNN...)
• DAUS031: Train Classification method
• DAUS032: Test Classification method
• DAUS033: Show results of Classification
• DAUS043: Verify an analysis module
• DAUS044: Drop an Analysis module
• DAUS045: Activate an analysis module
• DAUS046: Deactivate an analysis module
• DAUS049: Update data subscription
• DAUS050: Drop data subscription
• DAUS055: Update threshold
• DAUS056: Drop threshold
• DAUS057: Update sentence
• DAUS058: Drop sentence
• DAUS059: Activate sentence
• DAUS060: Deactivate sentence
• DAUS062: Provide the mechanism to execute all active analysis
• DAUS068: Persists filtered data
• DAUS069: Get filtered data
• DAUS070: Update status of filtered data

The user stories for R3 are:
• DAUS020: Forecasting (at Regression)
• DAUS027: Forecasting (at Machine Learning)
• DAUS034: Forecasting (at Classification)
2 Data Infrastructure Middleware (T5.1)

2.1 Scope

The Messaging and Publish/Subscribe (Pub/Sub) components are used by the vf-OS components to exchange messages with each other. All messages use the AMQP\(^1\) protocol that provides reliable solutions for systems-interconnection based on messages exchange, using a combination of point-to-point messages, publish-and-subscribe or multicast communication channels. The Messaging and Pub/Sub components use a message broker\(^1\) to handle the communication between the message producer and message receiver. This broker is connected with all vf-OS components and it has the responsibility to send the messages to the right component.

To use the Messaging and Pub/Sub components, the vf-OS components will have access to one library that has the necessary functionalities to communicate with the broker using the AMQP protocol.

To simplify this task, it has consequently been split into two components according to these two different functionalities:

- **Messaging** is responsible for sending messages for all the components of the platform through their identification
- **Publish/Subscribe** is composed by a publisher that sends messages to one or more topics, and composed by a consumer, that receives messages from specific queues

2.2 M18 Report

2.2.1 Planned Activities

The main activities that were planned for this period were:

- Develop the functionalities to send messages for specific components
- Develop the functionalities to send messages to a specific topic
- Develop the functionalities to get messages from a specific topic
- Perform an analysis of what are the communication needs for the vf-OS components
- Establish the developments priorities to meet the dependencies of messaging and Pub/Sub components

2.2.2 Progress

In the current reporting month, the progress status of the task is:

- A basic Messaging and Pub/Sub functionalities has been developed. The different actions that this Messaging and Pub/Sub can already execute are, according to the nomenclature used in the Functional Specification (Annex C: Data Infrastructure Middleware R1 Software Packages):
  - Determine message destination(s)
  - Route messages to one or more of many destinations
  - AMQP to send and receive messages from the broker and vf-OS components

\(^1\) [http://158.42.105.151/mediawiki/index.php/Middleware#tab=Messaging-Pub_2FSub](http://158.42.105.151/mediawiki/index.php/Middleware#tab=Messaging-Pub_2FSub)
• Historical storage of messages
• Notify events or errors
• Register publisher
• Unregister publisher
• Notify un-registration of publishers to subscribers
• Discover available messages for subscription
• Subscribe to messages
• Unsubscribe to messages
• Publish messages by message producers
• Send data to subscribers.

2.2.3 Next Activities

The activities that are planned for the next period are as follows:
• Analyse the necessary configuration for Messaging and Pub/Sub components and develop those functionalities
• Integrate with security component to perform the necessary authorisation procedures
• Analyse how the broker can handle inter-platform communication
• Develop the UI that can have all necessary information either for Admin or for the vf-OS component
• Streamline the installation process of these components (pre-configurations) into the broker.

2.3 M24 Report

Content to be added at M24.

2.4 M30 Report

Content to be added at M30.
3 Data Storage (T5.2)

3.1 Scope
The Data Storage component allows other vf-OS components and assets to persist data efficiently. It provides different data storage services oriented to different data storage needs:

- **Relational Data storage**: Service designed to store, retrieve, and manage structured data
- **Document Oriented storage**: Service designed to store, retrieve, and manage semi-structured data. A typical use case of this kind of databases is storing XML or log files
- **Time Series storage**: Service designed to store, retrieve, and manage time series data. A typical use case of this kind of databases is real time sensor data storage
- **Triple store**: A service designed to store, retrieve, and manage triples (semantic facts) through semantic queries.

3.2 M18 Report

3.2.1 Planned Activities
The main activity that was planned for this period was to develop the Relational Data Storage Service. For that the following tasks were planned:

- Select the base Relational Database Management System (RDBMS) that will be used as the service core
- Search the existing REST API tools for the selected RDBMS
- If there is no existing one to be used off the shelf:
  - Select the development tools (language, frameworks, components...)
  - Detailed design
  - Development
  - Testing
  - Documentation
  - Deployment

3.2.2 Progress
In the current reporting month, the progress status of the task is:

- The Relational service is finished. It is a REST API which allow users to manage relational assets (databases, tables, views, indexes, rows, constraints...)
- PostgreSQL was selected as the basis RDBMS
- An existing REST API tool that covered all the requirements detected was not found but the three most likely candidates were:
  - **Dreamfactory**: Main Limitation: It does not allow the creation of views nor indexes
  - **PostREST**: Main Limitation: It needs to start from a previously defined database schema. Thus, it does not allow to create dynamic schemas, tables etc
• **restSQL**: Main Limitation: It allows only basic querying

• Thus, it has been decided to create a generic REST API for PostgreSQL from scratch

• The development tools selected were:
  • Swagger-OpenAPI: For API definition and documentation
  • Swagger codegen: For API skeleton code definition
  • Java: As the development language
  • Springboot: As the development framework
  • JDBC: As database access library
  • C3P0: As connection pool.

• With these tools, the development, documentation, testing, and deployment were performed. A description of this storage can be found in Annex D: Data Storage R1 Software Packages.

### 3.2.3 Next Activities

The main activity that is planned for the upcoming period relies on the development of the document-oriented storage service and the Triple store. The development plan will be similar to the one used for the relational storage service.

Besides, global security measures will also be applied to the developed components. This has not been applied yet to Relational Data Storage component as there are still issues to develop in the security component.

### 3.3 M24 Report

Content to be added at M24.

### 3.4 M30 Report

Content to be added at M30.
4 Data Harmonisation (T5.3)

4.1 Scope

The Data Harmonisation aims to (a) extract information from incoming data, and (b) prepare the data in a form/schema suitable for other components. These functionalities are split in two different phases: Prepare maps and execute maps. To simplify this task, it has consequently been split into two components according to these two different functionalities:

- **Data Mapping** is responsible for defining the Manufacturing Maps that will allow the transformation and integration of data
- **Data Transformation** is responsible for integrating data from concrete existing software systems by executing the Manufacturing Maps created, ie transforming data from its source format to its destination format

4.2 M18 Report

4.2.1 Planned Activities

The main activities that were planned for this period were:

- Perform an analysis of the pilots’ functional requirements to identify the workflow needs for the Data Harmonisation
- Establish the development priorities to meet the objectives of the task
- To have an initial and basic Data Mapper ready:
  - To deal with basic data formats, eg CSV, XML, JSON, etc
  - To be able to load source and target schemas
  - To generate a semantic map
  - To create the vf-OS Data Model and launch initial queries against it
- No specific targets were established for this period for the Data Transformation component as its development is expected to start after the Data Mapping is in a more stable and ready stage.

4.2.2 Progress

In the current reporting month, the progress status of the task is:

- A basic Data Mapper tool has been developed. The different actions that this Data Mapper can already execute are, according to the nomenclature used in the Functional Specification (Annex E: Data Harmonisation R1 Software Packages):
  - Connect to Data Source
  - Read CSV, TXT, XML and XLS files
  - Load source and target schemas
  - CRUD operations with the Manufacturing Maps (in a local storage)
  - Calls for the vf-OS Data Model.
4.2.3 Next Activities

The activities that are planned for the next period are as follows:

- The basic Data Mapper will be ready for the start of the Data Transformation component. The different additional actions that this Data Mapper would be able to execute are the following:
  - Credentials to connect to the vf-OS Storage and the vf-OS Store
  - Queries against the vf-OS Data Model
  - Read other data formats
  - Suggest semantic concepts when importing the schemas
  - Create, deploy and publish transformation services
  - Update of the vf-OS Data Model.

- An initial version of the Data Transformation will be developed. The different actions that this Data Transformation would be able to execute are the following:
  - Unpack service
  - Read input data
  - Connect to the vf-Store.

4.3 M24 Report

Content to be added at M24.

4.4 M30 Report

Content to be added at M30.
5 Data Analytics (T5.4)

5.1 Scope

This task covers the creation of the building blocks for off-line analytical processing of sensor inputs. This includes machine learning algorithms supporting supervised and unsupervised scenarios. The core of the analytic algorithms is based on the combination of modern statistical-machine-learning and linear-algebra based algorithms (e.g. SVM, CRF, LDA, Mixture-Models), and traditional data-mining algorithms (e.g. decision trees and rules, K-Means, association rules, etc.).

A broad and diverse range of intelligent sensors provide a massive amount of data as inputs for the Data Analytics component. The data from the sensors will be published in real-time as a Stream data and can also be stored in databases as Historical data. The Historical data is used by the Machine Learning (ML) algorithms to train models to be able to make predictions in real time on stream data. The result of the processing is provided in a form that is useful for decision makers and also for software systems able to act on the provided analytics.

The tasks required for processing streaming data include Anomalies detection, Prediction of the future evolution of the data, Prediction for systems control, and Interpretation of data and visualisations.

Figure 3 shows the relationship between the Data analytics components (and its sub-components) and the functionalities provided as one of the vf-OS components.

![Data Analytics Component and its functionalities](image)

5.2 M18 Report

5.2.1 Planned Activities

The main activities that were planned for this period were:

- Perform an analysis of pilots’ functional requirements to identify the workflow needs for the Data Analytics
- Establish the technologies to be (re-)used and the developments priorities to meet the objectives of the task
- Prioritise the development tasks to guide the compliment of the objectives
- Have an initial and basic development environment ready:
  - To deal with the different ML libraries to re-use
  - To Include ML algorithms supporting supervised and unsupervised scenarios
• To be able to execute an example of a ML algorithm against the deployed server
• To establish a methodology to deploy other ML algorithms
• To have a basic API prototype of the stream analysis sub-component, that interacts with an environment analysing the incoming streaming data against the stream analysis modules defined by vf-OS developers.

5.2.2 Progress

In the current reporting month, the Data Analytics component workflow is ready as illustrated in Figure 4, where the process starts with the connection plugins enabling access to the data sources to transform the data into one or more data frames. Once the data is on its data frames form, the execution of the selected ML algorithm takes place in an iterative way until obtaining a valid model (algorithm training). The final model could be exported as a POJO/MOJO file and utilised by the user (Developer) to build its own application.

![Figure 4. Data Analytics Component Workflow](image)

Regarding the identification of the ML technologies to be (re-)used, H2O was selected against Weka (See D2.1). The final selection was taken after observing that H2O presented a higher rate of predicting precision for the H2O models which included a lower execution time. It was also found that H2O provided better portability and interoperability features than Weka for the Data Analytics component (e.g., H2O can be used with Big Data frameworks such as Apache Spark and Hadoop). H2O combines the power of highly advanced algorithms, the freedom of open source, and scalable in-memory processing for Big Data on one or many nodes.

Regarding the Historical data analytics sub-component, a basic environment hosts an H2O server, and a base case of the K-Means algorithm was developed using the web-flow to import the historical data, create the training and test frames, execute the training process of the model and finally export it as POJO and MOJO files. These files can be used as libraries on its own Java application to validate the usability and portability as a first step of the Anomalies detection functionality.

The H2O web interface is able to create flows to produce models based on either supervised or unsupervised scenarios of ML algorithms. These models can be used also as libraries to combine algorithms and improve the Prediction task.

The Stream analysis sub-component has a basic environment to execute a module of stream analysis defined by a vf-OS developer and an early API prototype with basic functionality.

A deeper description of the Data Analytics component can be found in Annex F: Data Analytics R1 Software Packages.
5.2.3 Next Activities

The planned activities for the next period include the analysis and further development of the major functionalities included in the Data analytics component (Anomalies detection and Making of Predictions) and refinement of the Sub-components.

Once the Data Analysis is enabled with access to historical data, the component will be able to improve the Anomalies Detection functionality and identify and prioritise the activities which will be then the input for the Prediction functionality.

The stream analysis sub-component will be further developed with the integration of vf-OS components Data Storage (Relational storage), Data Infrastructure middleware (Pub/Sub) and security, and the addition of functionalities to obtain a complete definition of stream analysis modules by the API. Finally, it is also planned to develop the GUI to ease the definition of the stream analysis modules.

5.3 M24 Report

Content to be added at M24.

5.4 M30 Report

Content to be added at M30.
6 Conclusions

This document reports on the different WP5 tasks’ activities providing the current umbrella status at M18.

It first reiterates the definition of what the vf-MW is and what tasks and components compose it. Its main aim is to create the v-OS data infrastructure including interoperability connectors, storage, transformation services, and analytics functionality. It is composed of the following tasks and components:

- T5.1 - Data Infrastructure Middleware: Messaging and Pub/Sub
- T5.2 - Data Storage: Data Storage
- T5.3 - Data Harmonisation: Data Mapping and Data Transformation
- T5.4 - Data Analytics: Data Analytics

All these tasks are being closely monitored and evaluated against the user stories presented in D2.2 Functional Specification & Mock-ups. Every section of the document describes the achievements of each task/component and is being complemented by the software reports of the corresponding annexes.

Since this is the first iteration of the software reporting of WP5, the deliverable acts as the basis for further iterations, ie progresses and improvements, of the different vf-MW components.
## Annex A: History

<table>
<thead>
<tr>
<th>Document History</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Versions</strong></td>
</tr>
<tr>
<td>V0.1:</td>
</tr>
<tr>
<td>• First Structure of document produced by Oscar Garcia</td>
</tr>
<tr>
<td>V0.2:</td>
</tr>
<tr>
<td>• Input from the different WP5 tasks/components</td>
</tr>
<tr>
<td>V0.3:</td>
</tr>
<tr>
<td>• Internal Review 1</td>
</tr>
<tr>
<td>V0.4:</td>
</tr>
<tr>
<td>• Internal Review 2</td>
</tr>
<tr>
<td>V0.5:</td>
</tr>
<tr>
<td>• Amendments to Internal Review</td>
</tr>
<tr>
<td>V0.6:</td>
</tr>
<tr>
<td>• Coordinator’s review</td>
</tr>
<tr>
<td>V1.0:</td>
</tr>
<tr>
<td>• Amendments to Coordinator’s review and submission</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ICE:</strong></td>
</tr>
<tr>
<td>• Stuart Campbell – Document Structure and Final review</td>
</tr>
<tr>
<td>• Oscar Garcia – Main Editor, Document Structure, Sections 1, 4, 5 and related annexes</td>
</tr>
<tr>
<td>• Arturo Brotons – Section 4 and related annex</td>
</tr>
<tr>
<td>• Armando Aguayo – Section 5 and related annex</td>
</tr>
<tr>
<td><strong>UNINOVA</strong></td>
</tr>
<tr>
<td>• Joao Sarraipa – Section 2 and related annex</td>
</tr>
<tr>
<td>• Joao Gião – Section 2 and related annex</td>
</tr>
<tr>
<td><strong>IKER</strong></td>
</tr>
<tr>
<td>• Oskar Saiz – Section 3 and related annex</td>
</tr>
<tr>
<td>• Lurdes Hernandez – Section 5 and related annex</td>
</tr>
</tbody>
</table>
Annex B: References


Annex C: Data Infrastructure Middleware R1 Software Packages

This annex provides the detailed documentation of the R1 software packages developed for the Data Infrastructure Middleware task. As both are being developed in parallel, they are considered in the subsections below.

C.1. Messaging Software Package

This section comprises the instructions on how to download, deploy and execute the Messaging component.

The Messaging component is composed by two entities, the message broker service RabbitMQ and the JavaScript library lib-messaging-pub-sub-js that handles the communication between vf-OS components and the broker.

C.1.1. Messaging Requirements

In order to install the message broker, a system running the Docker Engine is required. Though the Docker Engine can be used on different families of operating systems, GNU/Linux is recommended as it is the native host for such type of container technology and it will allow the component to run without additional compatibility layers.

The JavaScript library, on the other hand, requires the NodeJS library amqplib to be installed and imported. The installation can be accomplished through the Node Package Manager (NPM) by running `npm install amqplib`.

C.1.2. Messaging Installation

The broker is released in the form of a Docker image named `rabbitmq`.

A container running RabbitMQ can be created by running the following command:

```
```

Installing the JavaScript library can be performed through NPM, by running the command:

```
npm install lib-messaging-pub-sub-js.
```

C.1.3. Messaging Execution

After installing the broker, it can be initialised by running `docker start vfos-broker` and the JavaScript library can be imported into the project component.

Upon importing the library, the project component will be extended with features that allow it to communicate with the broker, send and receive messages.

C.2. Pub/Sub Software Package

This section comprises the instructions on how to download, deploy and execute the Pub/Sub component.
C.2.1. Pub/Sub Requirements

The broker is released in the form of a Docker image named `rabbitmq`.

A container running RabbitMQ can be created by running the following command:

```
```

Installing the JavaScript library can be performed through NPM, by running the command:

```
npm install lib-messaging-pub-sub-js.
```

C.2.2. Pub/Sub Installation

After installing the broker, it can be initialised by running `docker start vfos-broker` and the JavaScript library can be imported into the project component.

Upon importing the library, the project component will be extended with features that allow it to communicate with the broker, send and receive messages.

C.2.3. Pub/Sub Execution

After installing the broker, it can be initialised by running `docker start vfos-broker` and the JavaScript library can be imported into the project component.

Upon importing the library, the project component will be extended with features that allow it to communicate with the broker, send and receive messages.
Annex D: Data Storage R1 Software Packages

This annex provides the detailed documentation of the R1 software packages developed for the Data Storage task. This task comprises four types of storages: relational, document-oriented, time-series and triple-stores.

D.1. Relational Storage Software Package

This section comprises the instructions on how to download, deploy and execute the Relational Storage component. The Relational Storage Software Package is composed of two Docker containers:

- PostgreSQL Database Management System
- REST API service that allows other vf-OS assets to use it

D.1.1. Relational Storage Requirements

The component is deployed using Docker. Therefore, in order to deploy this component, a platform running Docker is required. It is recommended to use any Linux 64 bits distribution (Ubuntu, CentOS...) as operating system.

The Linux host must have Docker Engine installed, version 1.13.1 or higher. Instructions for Docker Engine installation can be found at: https://docs.docker.com/cs-engine/1.13/

D.1.2. Relational Storage Installation

The installation of the service is as follows:

- Configure the physical place for the database to store the data (a disk volume). A folder where the PostgreSQL data will be stored should be created and given a name, eg myPostgressDataVolume
  Example: /home/usr1/postgressDataVolume
- Decide the ports:
  - The port where the PostgreSQL will be accessible (eg, portPostgresql)
    Example: 8082
  - The port where the Relational Storage Service will be accessible (eg, portRelStorage)
    Example: 8083
- Obtain the relational storage Docker file and configuration file. Use the following address and password:
  - Address: It can be found at the following route under the link provided in Section 0.9: M18 vf-OS Software Prototypes at Month 18\ WP5\T5.2 Data Storage\RelationalStorage\bin
  - Files:
    relational-storage-1.0.0.jar
    application.properties
  - Password: vf-OS_password
- Write the address of the PostgreSQL docker in the application.properties file. Edit that file and set the db.postgresqlUrl entry, using the IP and
Example:

db.postgresqlUrl=jdbc:postgresql://185.254.12.54:8082/postgres

Where:

- IP = 185.254.12.54
- portPostgresql = 8082

- Compile the Relational Storage Dockerfile to create the Docker image. The following command should be executed (Note there is a dot (.) at the end of the command):

  
  $ docker build -t relational-storage-1.0.0 .

### D.1.3. Relational Storage Execution

The following steps must be followed to run the service:

- Run the PostgreSQL Docker container by executing the command:

  

  
  Example:

  
  $ docker run -d -p 8082:5432 -v /home/usr1/postgressDataVolume:/var/lib/postgresql/data -e POSTGRES_PASSWORD=vfos postgres:9.6

- Run the Relational Storage Service Docker container by executing the command:

  
  $ docker run -d -p portRelStorage:8080 relational-storage-1.0.0

  
  Example:

  
  $ docker run -d -p 8083:8080 relational-storage-1.0.0

The services of the component should now be available at <server-address>:portRelStorage.

Security issues have not been yet included in the component. As such, for testing purposes, the following credentials should be used:

- usr:postgres
- pwd:vfos

### D.2. Document-oriented Storage Software Package

This storage is not due during this reporting period.

### D.3. Time-series Storage Software Package

This storage is not due during this reporting period.

### D.4. Triple-store Storage Software Package

This storage is not due during this reporting period.
Annex E: Data Harmonisation R1 Software Packages

This annex provides the detailed documentation of the R1 software packages developed for the Data Harmonisation task. Of the two vf-OS components that compose the Data Harmonisation, only the Data Mapping component is considered in this period.

E.1. Data Mapping Software Package

This section comprises the instructions on how to download, deploy and execute the Data Mapping component.

The Data Mapping component is divided in one main subcomponent responsible for the design of the manufacturing maps and some minor subcomponents responsible for other tasks, such as Linked Concepts and managing the maps.

E.1.1. Data Mapper Requirements

Some preparations are needed in order to install and run a stable instance of the Data Mapper. Even if the Data Mapper may run on other operating systems, it is recommended to use a Windows 7 (64-bit) or newer, because it has been the most tested scenario at this stage of development, ie R1.

The development of Data Mapper has been based on the latest version available of the ICE Data Gateway which, in turn is based on one of the latest versions of the Talend Open Studio for Data Integration [Tal17], TOS-DI. It is available as a single ZIP file ready to be extracted and executed. It may be needed to have full access rights to the extracted folder, as it is needed by the original TOS-DI.

In addition, another requirement for being able to run the Data Mapper is to install Java Runtime Environment (JRE). For this release, the official recommended version was Oracle’s JRE7, but as it is already discontinued, it may be preferred to install JRE 8, which can be found on Java’s official website.

After installing the JRE, it is needed to check that the JAVA_HOME System Variable is present and points to the actual Java installation folder, being typically C:\Program Files\Java\JRE_x.x.x; and the Path System Variable contains the sequence “%JAVA_HOME%\bin” (without quotes). More details can be found on this topic at Talend Help Center [Tal17-2].

E.1.2. Data Mapper Installation

The Data Mapper is provided as a ZIP file. It hosts a folder structure containing all the files needed for its execution, as well as the extra functionality developed during this first release (M13 to M18).

The installation ZIP file can be found at the following route under the link provided in Section 0.9: M18 vf-OS Software Prototypes at Month 18\WP5\T5.3 Data Harmonisation. Once downloaded, it should be extracted in a folder, eg D:\vfOS_DataMapping with full rights.
E.1.3. Data Mapper Execution

To run the Semantic Mapper, it is only needed to navigate to the root folder of TOS-DI and run one of the executables, TOS_DI-win32-x86.exe or TOS_DI-win-x86_64.exe, depending on the architecture of the computer where it is executed. If all the previous steps have been successfully performed, the ICE Data Gateway splash screen should appear (see Figure 5), followed by the User License Agreement (see Figure 6), and the selection of the first project (see Figure 7).

![ICE Data Gateway Splash Screen](image1.png)

Figure 5: ICE Data Gateway Splash Screen

![User License Agreement](image2.png)

Figure 6: ICE Data Gateway User License Agreement
At the selection of the first project step, using the “Manage Connections” button (see Figure 7), it is possible to insert the vf-OS settings of the current session such as the vf-OS username and password and different URLs of services used by the Data Mapper. However, this data will be rolled up in the upcoming release. Clicking “Finish” (Figure 7) will create an empty project and load the Data Mapper.

In order to create a manufacturing map, it is needed to create a workflow to host it. This is performed by opening the contextual menu on the “Workflows” item in the Data Harmonisation panel (see Figure 8).

Also input and output schemas need to be imported under the “Schemas” element. On the sequence of Figures 9 to 12, an example on how to import a schema can be found.
Figure 9: Importing a Schema in ICE Data Gateway

Figure 10: Selecting Source File for a new Schema in ICE Data Gateway
The recently created Schema can be then used on an input component such as tFileInputJSON. To do so, it is needed to open a workflow where it is going to be used and drag and drop from the palette a tFileInputJSON (see Figure 13).
Once all previous steps have been carried out, i.e., adding source and target components (e.g., `tFileInputJSON`, `tFileOutputXML`, etc.) to the canvas, a `tSemanticMap` component can be placed in between, which will be responsible for mapping the input schema to the output. By opening the advanced UI of the `tSemanticMap` with a double click, the interface will show a 5 stages transformation, from input schema fields, to input concepts, variables (which are optional), output concepts and output schema fields. On the bottom, once the Reasoner services are developed, suggested concepts will appear for non-assigned schema fields (see Figure 14). To assign a concept to the ‘unassigned’ input field, it is as easy as check the desired concept.
If none of the concepts suggested by the Reasoner engine satisfies the semantics of the field, a new concept can be created by opening the contextual menu on the field and choosing “Create Concept…” (see Figure 15).
The maps can also be organised on an advanced level by annotating them. This option can be found on the context menu by right clicking on a workflow (see Figure 16).

Afterwards, annotated maps can be found via the Map Manager, which will allow downloading a map from the vf-OS Storage and modifying it according to the mapping needs (see Figure 17).
In order to make an Annotated Map available to build a manufacturing process, maps need to be published to the vf-Store. This can be performed by opening the context menu of a Workflow with right click and then selecting Publish Map. This action will open a dialog (see Figure 18), similar to the Annotate Map one, which will allow publishing the manufacturing map to the vf-Store.

Before being able to publish a map, it is necessary to build it from the source files, making use of the button Build (it is also suggested when trying to click the Publish button). This
action will trigger TOS-DI Build Job routine, which has been adapted and simplified when called from a vf-OS Asset to show a minimal set of required information to generate the dockerised map (see Figure 19).

Figure 19: Build vf-OS Map Interface with Minimal Set of Fields

The publishing process is then handled by the DHS service, which will receive the generated file and process it to convert the map files into a Docker image, suitable to be published to the vf-Store (see Figure 20).
Figure 20: Successful Publishing Process

E.2. Data Transformation Software Package

This component is not due during this reporting period.
Annex F: Data Analytics R1 Software Packages

This annex provides the detailed documentation of the R1 software packages developed for the Data Analytics task. This component is divided in two main subcomponents each of them responsible for the historical data and for the stream data analysis.

The packages can be retrieved from the following route under the link provided in Section 0.9: M18 vf-OS Software Prototypes at Month 18\WP5\T5.4 Data Analytics.

F.1. Historical Data Analytics Software Package

This section comprises the instructions on how to download, deploy and execute the Historical Data Analytics component.

F.1.1. Historical Data Analytics Requirements

The following is the list of requirements for a successful deploy of the development environment of the Historical Data Analytics module:

- Java Virtual Machine version 7 or higher
- Docker Community Edition version 7.12.0 or higher
  - For prerequisites and OS requirements go to the next link: [https://www.docker.com/community-edition](https://www.docker.com/community-edition)
- H2O Docker container
- Install and Launch Docker

F.1.2. Historical Data Analytics Installation

The following is the list of steps for successfully installing the development environment of the Historical Data Analytics module:

- Install Java. See: [https://docs.docker.com/install/](https://docs.docker.com/install/)
- Install Docker container CE. See: [https://www.docker.com/community-edition](https://www.docker.com/community-edition)
- Pull a H2O docker image and make sure that the Docker daemon is running. To do this, enter all commands below in a terminal command:
  
  ```bash
  $ docker pull lkwg82/h2o-http2-server
  ```

F.1.3. Historical Data Analytics Execution

The following is the list of steps for executing the development environment of the Historical Data Analytics module:

- The easiest way is to run an H2O server using the following commands on a terminal:
  
  ```bash
  docker run -p "8080:80" -ti lkwg82/h2o-http2-server
  ```

- For testing, use the following commands on a terminal:
  
  ```bash
  $ curl http://localhost:8080/
  ```
F.2. Stream Data Analytics Software Package

This section comprises the instructions on how to download, deploy and execute the Stream Data Analytics component.

F.2.1. Stream Data Analytics Requirements

In order to deploy this component, a UNIX server host is required. It is recommended to use the Linux distribution Ubuntu 16.04 or above as the operating system. In any other operating system this subcomponent should work, but it cannot be guaranteed.

The subcomponent can be deployed by using Docker. Docker containers wrap the software in a complete file system that contains everything needed to run the application. This guarantees that the software will always run in the same environment, regardless of the host environment.

The Linux host must have Docker Engine installed, version 1.13.1 or higher. Instructions for Docker Engine installation can be found here: https://docs.docker.com/cs-engine/1.13/

- Docker engine version 1.13.1 or higher
- Docker image built for Stream Data analytics, it hosts the following packages:
  - Ubuntu:16.04
  - Oracle Java 8

F.2.2. Stream Data Analytics Installation

For installing this component API with Docker, a recent version of the subcomponent’ dockerfile is required. To get a recent version of it use the following route under the link provided in Section 0.9: M18 vf-OS Software Prototypes at Month 18\WP5\T5.4 Data Analytics\StreamAnalysis. The source code contains its corresponding dockerfile to build the Docker image. The following commands must be executed from the API folder to build and run the image of it.

    $ docker build -t stream-analysis

F.2.3. Stream Data Analytics Execution

When the Docker image of the subcomponent is built, the following commands need to be typed in order to launch the Docker image.

    $ docker run -d -p 31700:31700 stream-analysis

The services of the subcomponent should now be available at <server-address>:31700, where port 31700 should be an available port on the server.