

SEMANTICS FOR PERFORMANT AND SCALABLE INTEROPERABILITY OF MULTIMODAL TRANSPORT

D3.4 – Requirements Analysis and Design of Architecture, Testing Infrastructure, Test Cases and Benchmarks of the IF (F-REL)

Due date of deliverable: 30/06/2020

Actual submission date: 31/08/2020

Leader/Responsible of this Deliverable: CEF

Reviewed: Y

Document status		
Revision	Date	Description
0.1	19/05/2020	Initial table of contents
0.2	22/07/2020	Update document structure, add traceability matrix for requirements
0.3	30/07/2020	Added introduction and executive summary
1.0	04/08/2020	Document ready for review
2.0	31/08/2020	Final version after TMC approval

Project funded from the European Union's Horizon 2020 research and innovation programme		
Dissemination Level		
PU	Public	X
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

Start date of project: 01/12/2018

Duration: 25 months

EXECUTIVE SUMMARY

This deliverable describes the output of Tasks 3.2, 3.3 and 3.4 of the SPRINT project, for what concerns the F-REL milestone. This document reports the final set of requirements and user stories which drove the final release of the architecture of the SPRINT Interoperability Framework. Moreover, we describe the performance and scalability requirements for all of the SPRINT components, and we then explain all the tests that will be carried out on the SPRINT testing infrastructure to validate them.

ABBREVIATIONS AND ACRONYMS

Abbreviation	Description
ADMS	Asset Description Metadata Schema
AM	Asset Manager
CQRS	Command Query Responsibility Segregation
DCAT	Data Catalog Vocabulary
DCAT-AP	DCAT Application Profile
GTFS	General Transit Feed Specification
IF	Interoperability Framework
IT	Information Technology
OBDA	Ontology Based Data Access
OBDI	Ontology Based Data Integration
RDF	Resource Description Framework
RML	RDF Mapping Language
R2RML	RDB to RDF Mapping Language
xR2RML	eXtended R2RML
S2R	Shift2Rail
RDF	Resource Description Framework
WSDL	Web Services Description Language

TABLE OF CONTENTS

Executive Summary	2
Abbreviations and Acronyms	3
Table of Contents.....	4
List of Figures	6
List of Tables	7
1. Introduction	8
2. Stakeholder Identification	10
3. Interoperability Framework User Stories	12
3.1 Modified user stories	14
3.2 New user stories	17
4. Final design of the Interoperability Framework architecture.....	20
4.1 Review of requirement analysis.....	20
4.2 IF Architecture.....	23
4.2.1 Data and Service Abstraction.....	24
4.2.2 Asset	25
4.2.3 Asset Manager	26
4.2.4 User Management	30
4.2.5 Interoperability Services and Utilities	31
4.3 Deployment Strategies	32
4.3.1 Ready to use services, tools and utilities.....	32
4.3.2 Continues Development and Deployment.	33
5. Testing infrastructure	34
5.1 Defining Test Cases for the IF	34
5.2 Test checklist	35
5.3 Performance and scalability testing infrastructure	36
5.3.1 Scoping and non-functional requirements capture	37
5.3.2 Scripting performance test use cases	38
5.3.3 Validating performance test scenarios	38
5.3.4 Automatically executing the performance tests	38
5.3.5 Collecting data	39
5.3.6 Conducting the final analysis and reporting.....	40
5.4 Applying Testing Infrastructure Workflow	40
6. Performance and scalability requirements.....	52
6.1 Modified Requirements	54

6.1.1	Distributed SPARQL Endpoint	54
6.2	New Requirements.....	56
6.2.1	Asset Manager	56
6.2.2	Collaborative Ontology Management Tool	57
6.2.3	Automatic Generation of Ontologies from Non-ontological Sources (from XSD to Ontology)	58
6.2.4	Converter.....	60
7.	Performance and Scalability test cases.....	61
7.1	Collaborative ontology management	61
7.2	Automatic generation of ontologies from non-ontological sources	62
7.3	Mapping Tool	63
7.4	Asset Manager	64
7.5	Converter	65
7.6	Distributed SPARQL Endpoint.....	67
8.	Conclusions	69
9.	References	70

LIST OF FIGURES

Figure 1 Interoperability Framework and IP4 ecosystem.....	9
Figure 2 User Roles	10
Figure 3 Schematic IF Architecture	24
Figure 4 Asset domain model.....	26
Figure 5 IF Architecture.....	28
Figure 6 IF and External Identity Provider	30
Figure 7 IF Deployment Possibilities	32
Figure 8 Design of Testing Infrastructure	37
Figure 9 Distributed SPARQL query engine Architecture	42
Figure 10 Docker images for Virtuoso and Ontario.....	42
Figure 11 JMeter Client.....	43
Figure 12 Group creation	43
Figure 13 Selecting the number threads per second	44
Figure 14 Selecting HTTP Request.....	44
Figure 15 Showing configuration for an Ontario query	45
Figure 16 Adding a query.....	45
Figure 17 Adding listeners	46
Figure 18 Added listeners	47
Figure 19 Created Test Plan	48
Figure 20 Validating results in JMeter	48
Figure 21 Configuring a test in Jenkins	49
Figure 22 Executing a test plan in Jenkins	50
Figure 23 Collecting data results in Jenkins	50
Figure 24 Collecting data results in JMeter	50
Figure 25 Final analysis and reporting	51

LIST OF TABLES

Table 1 User story/Use Case Scenario Traceability Table	14
Table 2 Modified US-10	15
Table 3 Modified US-2 for Distributed SPARQL endpoints	16
Table 4 US-8 Generation of ontologies from non-ontological sources	17
Table 5 US-9 Collaborative Ontology Manager	18
Table 6 US-12 Asset Manager as metadata aggregator for multiple NAPs	18
Table 7 US-13 Asset Manager as NAP contributor	19
Table 8 Index of Data, Service and System Management Requirements	21
Table 9 An example of a Test Checklist	36
Table 10 – Requirements Traceability Table	54
Table 11 Performance requirement for Distributed SPARQL Endpoint	54
Table 12 Scalability requirement for Distributed SPARQL Endpoint	55
Table 13 Performance requirements for Exploration API: DCAT-AP metadata retrieval	56
Table 14 Scalability requirements for Exploration API: DCAT-AP metadata retrieval	57
Table 15 Performance requirement for Collaborative Ontology Manager	57
Table 16 Scalability requirement for Collaborative Ontology Manager	58
Table 17 Performance requirement for Generation of ontologies from non-ontological sources	59
Table 18 Scalability requirement for Generation of ontologies from non-ontological sources	59
Table 19 – Scalability requirement SR-4 for the Converter	60
Table 20 – Scalability requirement SR-5 for the Converter	60

1. INTRODUCTION

The aim of Shift2Rail IP4 is to provide a better experience to travelers in Europe through digital technologies. To achieve this goal, IP4 is promoting the creation of an open environment for traveler-centric services and is fostering the integration of services provided by different and competing transport operators related to the following features:

- Booking
- Ticketing
- Journey planning
- Issuing
- Trip tracking

Such macro-features are then accessed via Travel Companions, personal mobile applications by which travelers can obtain seamless access to all the transport services.

Interoperability in an open environment implies a multilateral scenario where multiple peers have the opposing needs of maintaining control and sovereignty over their systems and datasets while benefiting from other parties' information. Also, the transportation domain over the years produced a high number of standards and specifications, and forcing them to transition to newer standards may be not feasible in a short time. Integrating services provided by different transport operators, each one dealing with data conforming to different standards and specifications therefore means a large effort into solving several challenges, like:

- understanding similarities and differences in different standards and specifications pertaining to the same domain or sub-domain;
- creating upper level models, general enough to capture the information of multiple standards;
- governing how transport service providers can share information, and gain access to information provided by others.

The Interoperability Framework was born with the explicit role of providing the tools to support the creation of the Shift2Rail IP4 ecosystem. Since it provides tools and methodologies, the IP4 ecosystem is just one of the possible ecosystems that can be built using the SPRINT Interoperability Framework.

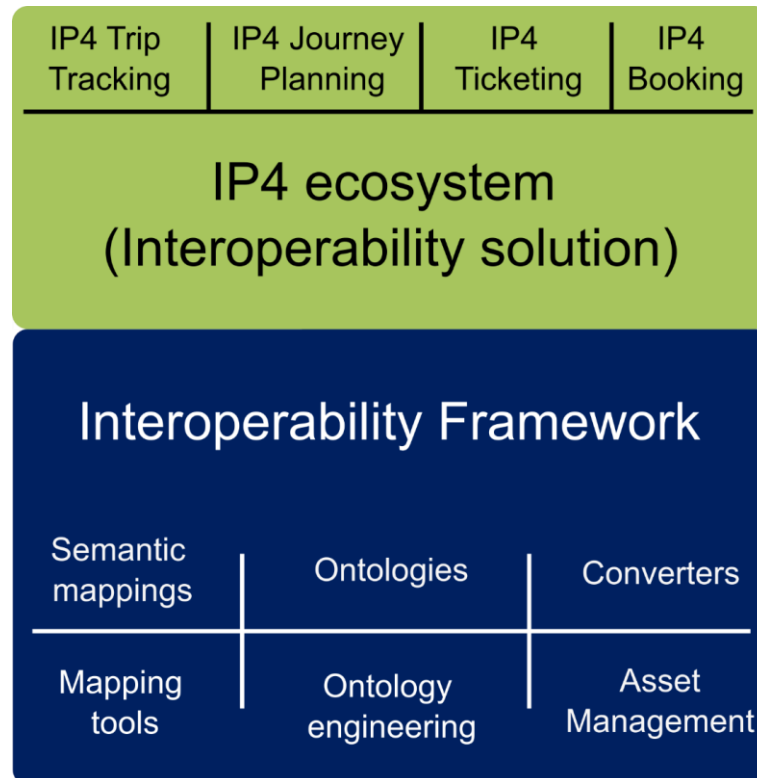


Figure 1 Interoperability Framework and IP4 ecosystem

The F-Rel version of the SPRINT Interoperability Framework provides, as depicted in Figure 1, a consistent set of tools and methodologies to support the creation of ontologies and mappings, to enact interoperability by building converters that can connect systems adopting different standards and specifications, and to share information according to a specific governance structure. Such tools and methodologies can be used to create an open ecosystem like the Shift2Rail IP4 ecosystem.

As F-Rel will be the final release of the SPRINT Interoperability Framework, this document updates and reviews its requirements, provides a refinement of its architecture and describes how all its components will be tested to evaluate its performances and scalability. We start with a review of the Interoperability Framework stakeholders in Chapter 0, which we updated according to our better knowledge of the ShiftRail IP4 landscape and related actors. Chapter 3 continues with the final set of user stories which we considered, and Section 4.1 describes how such user stories were translated into actual requirements for the IF. Section 4.2 provides the final architecture of the SPRINT IF, breaking it down into components and showing their roles and relationships.

Since the IF is a distributed system, it is important to define performance and scalability requirements, and to provide a consistent way to test them. In Chapter 5 we describe our testing infrastructure, and we provide guidelines for the implementation of performance and scalability tests. We proceed in Chapter 6 and 7 in reviewing and updating C-REL performance and scalability requirements, and we then describe test cases for each of the SPRINT IF components. Such tests will be executed on the F-Rel implementation which will be described in D5.5, and their results will be analyzed in D5.6.

2. STAKEHOLDER IDENTIFICATION

In Deliverable D3.3 we have a comprehensively identified potential stakeholders of IF. In this chapter we briefly overview a narrow-downed list that collects the stakeholders in which IF could apply major changes and improvements in different ways. It is also worth reminding that the stakeholders/users of IF would interact with system through two logical Roles: as Contributors (or Providers) and Consumers. Contributors are composed of a wide range of service/infrastructure providers in the transportation domain, including transport authorities, transport service providers, infrastructure managers, retailers and travel agency distributors [1]. Similarly, the IF consumers are also transportation actors such as travel service providers, social networks, and IT suppliers and software applications. Obviously, any actual corporation might interact with IF as both Consumer and Contributor. Table represents the primary IF stakeholders with their type. (For more details please refer to Deliverable SPRINT D3.2).

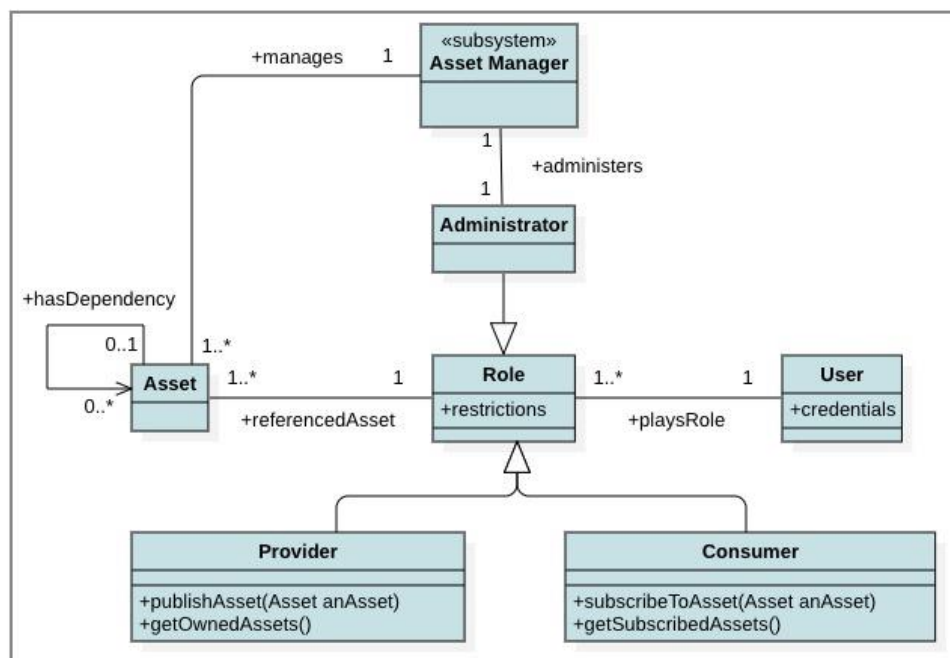


Figure 2 User Roles

Stakeholder(s)	<p>Transport service provider (TSP)</p> <p>Mobility as a Service (MaaS) Provider</p>
Major Interactions	<p>As Contributor:</p> <ul style="list-style-type: none"> To advertise its various types of services, data and assets (e.g., Ontologies) in IF

	<ul style="list-style-type: none"> • To securely let any interested users to utilize its service and data • To utilize available assets of IF to create new services • To utilize available assets to improve the its services • To automate various aspects of their services • To facilitate for users to engage and deploy its services • To collaborate with other parties and operators for development of assets such as ontologies • To understand which tools and services are used by which other TSP/MaaS provider, gaining a better knowledge about the status of the ecosystem
Stakeholder(s)	Retailer, Travel Agency, Distributor
Major Interactions	<p>As Consumer:</p> <ul style="list-style-type: none"> • (Distributors:) To discover various services offered by TSPs • (Retailers:) To discover various Distributors and their services • To access/query the web of transportation data <p>As Contributor:</p> <ul style="list-style-type: none"> • To advertise their various types of services, data and assets usually built upon services provided by TSPs • To cooperate and communicate with TSP, distributors and retailers in semantically interoperable manner • To utilize wide range of available assets, utilities and tools within IF to improve services • To automate various aspects of services and the process of building (mash-up) services • (Distributors:) To facilitate for users (Retailers) to engage and deploy its services
Stakeholder(s)	Travel service provider (TrSP), IT supplier and Software Application (ISA)
Major Interactions	<p>As Consumer:</p> <ul style="list-style-type: none"> • (Journey planning, offer building, Trip tricking, Travel companion, Travel Event Provider) To discover various types of services offered by TSPs and Distributors • To access/query the web of transportation data

3. INTEROPERABILITY FRAMEWORK USER STORIES

In the SPRINT project, we have referred to **User Story** as a generic description of communication of users and IF, and/or functionality of IF with respect to a particular identified challenge and requirement. **Use-Case Scenario** then extends the user stories and provide a more detailed description and step by step interaction of users with a distinct IF component(s) to realize one or more user stories.

The first group of User Stories have been previously defined in SPRINT Deliverable **D3.2** and complement with their respective Use-Case Scenarios in WP5. More specifically, we introduced the use-case scenarios in **D5.1** . Then the C-REL implementation of the modified version of those use-case scenarios, and, the validation results are reported in **D5.2** and **D5.3**^[1] respectively. Furthermore, in this deliverable, we are introducing **four** more new User Stories and present a modified version of **two** user stories.

For the sake of completeness and traceability, in Table 1 we list all the User Stories in SPRINT (including those defined in D3.2 and new stories introduced in current deliverable) along with the use-case scenarios of each user story (including those that have been implemented in D5.3 for C-REL, and, those that will be implemented in the forthcoming deliverables of WP5 for F-REL).

User Story ID	Reference to description of the User Story in WP3	Reference to description of the Use Case Scenario in WP5	Use-Case Scenario ID
US-1	Basic Scenario 1 in Deliverable D3.2 : Join and search user story	S1 in Deliverable D5.3 : Joining the IF Use case (Provider Registration)	UCS-1
		S2 in Deliverable D5.3 : Joining the if use case (user registration)	UCS-2
US-2	Basic Scenario 2 in Deliverable D3.2 , modified in D3.4 (Table 3) Distributed service/asset discovery	S3 in Deliverable D5.3 : Simple discovery	UCS-3
		S4 in Deliverable D5.4 : Distributed SPARQL endpoints	UCS-4
US-3	Basic Scenario 3 in Deliverable D3.2 : Batch Data Conversion	S5 in Deliverable D5.3 : Direct access use case for batch data conversion	UCS-5

US-4	Basic Scenario 4 in Deliverable D3.2: Runtime Data/Message Conversion	S6 in Deliverable D5.3: Direct download use case for runtime data/message conversion	UCS-6
US-5	Basic Scenario 5 in Deliverable D3.2: Fast adaptation to peaks	S9 in Deliverable D5.3: Fast adaptation to peaks	UCS-7
US-6	Basic Scenario 6 in Deliverable D3.2: Special purpose discovery package	S10 in Deliverable D5.3: Special purpose asset discovery package: Resolver user case	UCS-8
US-7	Basic Scenario 7 in Deliverable D3.2: Automated mapping process for the conversion user story	S7 in Deliverable D5.3: Automated mapping process for the conversion use case	UCS-9
US-8	Basic Scenario 8 in Deliverable D3.4: Generation of Ontologies from Non-ontological Sources	S1 in Deliverable D5.4: Generation of Ontologies from Non-ontological Sources in <u>Deliverable D5.4</u>	UCS-11
US-9	Basic Scenario 9 in Deliverable D3.4: Collaborative Ontology Manager	S2 in Deliverable D5.4: Collaborative Ontology Manager in <u>Deliverable D5.4</u>	UCS-12
US-10	Advanced Scenario 1 in Deliverable D3.2: Federated identification and access control	S1 and S2 in Deliverable D5.4: Joining the IF Use case (User Registration) in <u>Deliverable D5.4</u> Joining the IF Use case (Provider Registration) in <u>Deliverable D5.4</u>	UCS-1 UCS-2
US-11	Advanced Scenario 2 in Deliverable D3.2: Ad-hoc Converter creation	S8 in Deliverable D5.3: Automatic converter building use case	UCS-13

US-12	Asset Manager as NAPs aggregator	S13 in Deliverable D5.4 Asset Manager as National Access Points aggregator	UCS-14
US-13	Asset Manager as NAP contributor	S14 in Deliverable D5.4 Asset Manager as National Access Points aggregator	UCS-15

Table 1 User story/Use Case Scenario Traceability Table

3.1 MODIFIED USER STORIES

Federated identification and access control	
Type	Joining Phase
Actors	<p>X-com: <u>TrSP</u></p> <p>BE-Service: <u>TrSP</u> (Offer building service) for rail and road travel within central parts of Europe. Its front-end API is used by mobile and web applications (say, T-A-1 to T-A-10) and its back-end has access to, and, engaged with many train/bus operators (say, T-O-1 to T-O-20) in the covered zones.</p>
Story	<p>Alex is the head of IT department of X-com, which is one of the main rail operators of Country X. He recently has heard about the IF and he is interested to join the ecosystem to offer its travel services as well as various software and IT products/assets to other operators and travel applications, and to learn about their services and assets. However, X-com's services are accessible only under specific use-conditions. Similarly, the company supplies various types of data including ontologies, meta data, service descriptions, and so on. Though the access to some data is fully open, valued and sensitive data are available only to authorized users.</p> <p>The manager of BE-Service is also interested in being involved in the IF ecosystem in order to use its benefits, especially for easier/interoperable engagement with rail/road operators (see basic scenario 5) as well as to enjoy benefits of interoperability services such as converters and resolvers. However, BE-Service is already a client of X-com and some other operators, which means it already has gone through the registration process of such organizations.</p>

Challenge	<p>From the service provider viewpoint:</p> <p>In general, not only every single organization/ service provider such as X-com, but even each data source has its own well-established registration process and it applies its own policy and access control strategy, which is not necessarily the same as others.</p> <hr/> <p>From the service consumer viewpoint:</p> <p>The heterogeneity of access control mechanisms of service providers, plus the existence of intermediate organizations running their own user databases (which impose their own user registration mechanisms in addition to the destination organization) increases the complexity of the situation. As a result, a service consumer is required to repeatedly register to each and every service/data provider and follow a separate authentication process for each of the service/data requests it has.</p>
Goal	<p>X-com desires to define (in the IF) the access conditions to its resources according to its organization policy, which might be different from other organizations.</p> <hr/> <p>A typical service consumer desires to minimize the registration and authentication process and to be identified globally and in a cross-organization manner.</p>
Involved IF Components	Asset Manager

Table 2 Modified US-10

Distributed service/asset discovery	
Type	Engaging Phase
Actors	<p>B-Com: A Belgian-based TrSP which it is hosted by the (National Access Point) NAP of Belgium.</p> <p>S-com: A Spanish-based TSP named “S-com” already hosted by the NAP of Spain. S-com offers travel services within and beyond the Spain boundaries.</p> <p>V-com₁, V-com₂, ..., V-com₁₀: European TSPs which it is hosted by 10 (National Access Point) NAPs</p> <p>MyMobility: An Italian company providing transport services in many regions of Europe.</p>

Story	<p>Summer is approaching and MyMobility anticipates many travellers may target European towns. Accordingly, it would like to retrieve more details about catalogues describing data of public transport providers and information about the different public means of transport with European travel operators in order to tailor offers and facilities. Accordingly, services offered by S-com, B-com, V-com₁, V-com₂, ..., V-com₁₀ might be interesting for MyMobility.</p> <p>MyMobility searches to discover the desired service/data by querying the IF-node with which it has registered (which in turn federates the query across all IF-Nodes).</p>
Challenge	<p>The services provided by S-com/B-com/V-com₁/V-com₂/.../V-com₁₀ are stored in one instance of the IF, while MyMobility is initiating the discovery query from a geographically, computationally and administratively separate IF-Node. Hence, the challenge here is to perform a distributed service discovery and grant the maximum service visibility and discovery coverage while the number of nodes increases.</p> <p>In general, a TSP registers itself and its services on one instance IF-Node (e.g., the IF-Node hosted by the NAP of the TSP's country of origin) but its motivation to offer a service is mainly business-oriented and it does not limit to the boundary of a certain country. Accordingly, the potential users of such services might be distributed over various IF-Nodes across Europe.</p>
Goal	<p>It is not desirable and feasible for S-com/B-com/V-com₁/V-com₂/.../V-com₁₀ to register each of its services to the IF-Node that is more probable to be searched by a client. A generic service provider desires to register its services on the IF-Node in which it is known, and then such a service should be discoverable in any other instance of the IF.</p> <p>It is not desirable and feasible for MyMobility to query the IF-Node that is more probable to contain the desired service. A generic service consumer desires to initiate its search request in the IF-Node that it knows, but the result should include all the services matched with its request across all the IF-Nodes.</p>
Involved IF Components	Asset Manager, Distributed SPARQL endpoint

Table 3 Modified US-2 for Distributed SPARQL endpoints

3.2 NEW USER STORIES

Generation of Ontologies from Non-ontological Sources	
Type	Adaptation Phase
Actors	EU-Bus: A European TSP, in specific, a bus service provider that uses NeTex for exchanging Public Transport.
Story	EU-Bus currently has its own data model and uses NeTex XSD as the format for exchanging its data. However, EU-Bus wants to share and integrate its data with the IF node which has its own data model that shares many characteristics over the whole transport domain through an ontology that provides a global view of multiple data sources. XSD schemas are already part of the data lifecycle of EU-Bus, so EU-Bus searches to automate the ontology creation from non-ontological sources. In particular, EU-Bus will select one of the techniques to convert XSD schemas into ontologies, and it will develop mechanisms to then match XML data with the ontology.
Challenge	Currently, to transform XSD, the EU-Bus tool should be able to extract the shared concepts and map the terms in the source model (XSD) to the equivalent concepts in the target model (ontology).
Goal	EU-Bus would like to benefit from a more automated approach for the extraction of equivalent concepts in the non-ontological sources and ontological models.
Involved IF Components	Artefact Registry

Table 4 US-8 Generation of ontologies from non-ontological sources

Collaborative Ontology Manager	
Type	Adaptation Phase
Actors	EU-Tram: An European <u>TSP</u> , in specific, a tram service provider that aims to represent its data semantically.
Story	EU-Tram has generated ontologies to semantically represent its transport data. After generating its ontologies, EU-Tram wants to validate and document them. As part of the lifecycle of the ontology, EU-Tram has decided to use OnToology as a tool that comes into action from the evaluation/documentation part within the ontology development process.

Challenge	OnToology should be able to manage a large corpora of reference ontologies, and will adapt it in order to automate the generation/extension/revision of the ontologies to be developed during the ontology development process.
Goal	EU-Tram would like to benefit from a more automated collaborative approach for the generation/extension/revision of its ontologies and manage a repository of ontologies where EU-Tram can reuse data models.
Involved IF Components	Artefact Registry

Table 5 US-9 Collaborative Ontology Manager

Asset Manager as NAPs aggregator	
Type	Adaptation Phase
Actors	New-Move: An European <u>TSP</u> , offering cross-border services and mobility in several EU countries.
Story	New-Move is a transport operator mainly focused on cross-border services. To plan the improvement of their network, they need to study the possible connections with other TSPs in Europe. Being aware of the existence of the National Access Points, they need to access the NAP of each EU member State where they want to operate. Being this a repetitive task, they want to automate it and aggregate all the relevant information in one single place. Since they are one of the IF early adopters, they instruct their Asset Manager to access assets metadata from their target NAPs.
Challenge	The Asset Manager should access remote NAP catalogues, fetch the metadata of the published assets, convert them according to an Asset Manager metadata ontology, and show them both in the Publisher and in the Store applications.
Goal	New-Move wants to plan and organize its services from a unified application. Moreover, while developing new Converters, they want to reuse data which was officially published on National Access Points
Involved IF Components	Asset Manager, Converter

Table 6 US-12 Asset Manager as metadata aggregator for multiple NAPs

Asset Manager as NAP contributor	
Type	Adaptation Phase
Actors	New-Move: An European <u>TSP</u> , offering cross-border services and mobility in several EU countries.
Story	New-Move is a transport operator mainly focused on cross-border services. After having planned their new service offer, according to the EU regulations they need to send the NeTEx data representing their offer. Being this a repetitive task, they want to automate it. Being an early adopter of the IF, they configure their Asset Manager so that whenever they upload a new Journey Planning information related to an EU member State, the information is automatically converted into NeTEx and sent to the related NAP.
Challenge	The Asset Manager should be able to convert the provided data to NeTEx using a Converter, and convert the metadata to the format accepted by the specific NAP. Then, the Asset Manager should be compliant to the specific way of submitting metadata and dataset to the NAP, since each NAP implementation has its different process for contributing new assets.
Goal	New-Move wants to publish Journey planning information just once in an Asset Manager, which would then take care of automatically pushing the NeTEx representation of the dataset onto the right NAP.
Involved IF Components	Asset Manager, Converter

Table 7 US-13 Asset Manager as NAP contributor

4. FINAL DESIGN OF THE INTEROPERABILITY FRAMEWORK ARCHITECTURE

This section first reviews the main requirements and challenges that shaped the architecture of IF. It then presents the updated design of the architecture of the S2R IF with respect to the architecture described in deliverable D3.2. We overview the main components (already presented in D3.2) and introduce new/enhanced components.

4.1 REVIEW OF REQUIREMENT ANALYSIS

In previous SPRINT deliverable, D2.2, we have reported an extensive analysis of the IF requirements. We came up with around thirty functional and non-functional requirements overall, summarized in Table 8. In this section we briefly overview the IF requirement reported in WP2. For more details of the requirements analysis we recommend the interested reader to refer to Deliverable D2.1 to D2.4.

Data Management Requirement	
DR1	Data standardization and portability
DR2	Data accessibility and openness
DR3	Dataset lifecycle management
DR4	Depth of data
DR5	Efficiency
DR6	Machine readable data
DR7	Preservation of information
DR8	Quality of data
DR9	Security and Privacy
Service Management Requirement	
SeR1	Dataset publication and subscription services
SeR2	Discoverability
SeR3	Inclusion and accessibility for all types of users
SeR4	Multilingualism
SeR5	Quality of Service
SeR6	Service Efficiency
SeR7	Service Standardization

SeR8	User activity monitoring
SeR9	User-centricity of service design and implementation
System Management Requirement	
SyR1	Administrative simplification
SyR2	Authorization and Authentication mechanisms
SyR3	Fault tolerance and backup / Recovery mechanisms
SyR4	Guidelines
SyR5	Integration of complementary services
SyR6	Interoperable and flexible laws and regulations
SyR7	Profit vs Cost ratio
SyR8	Reusability
SyR9	Scalability
SyR10	Subsidiarity and proportionality
SyR11	System Efficiency
SyR12	System monitoring and assessment
SyR13	Technological neutrality and system/infrastructure harmonization
SyR14	Transparency

Table 8 Index of Data, Service and System Management Requirements

The identified requirements have been categorized into three main groups; Shorty recapped as follows.

- **Data Management Requirement (DR):** that represents the main interoperability challenges and requirements with respect to various aspects concerning data, including the management, sharing, access and distribution of any types of data within and across the transportation ecosystem.
- **Service Management Requirement (SeR):** that represents the main interoperability challenges and requirements concerning the design, implementation and cooperation of different types of IT services in the transportation domain, as well as the consumer expectations and needs for interacting with such services.

- **System Management Requirement (SyR):** that represents the main interoperability challenges and requirements concerning the design and development of the IF itself as an interoperability framework.

Furthermore, a set of requirements have been categorized as interoperability bottlenecks since they had a greater impact to tackle interoperability, or they were prerequisites to other requirements, and so any deficiency to address them may block combating other challenges. Such requirements also have been classified into three groups as follows.

- **Conceptual barriers:** They are concerned with the syntactic and semantic differences of information to be exchanged. These problems concern the modelling at the high level of abstraction (such as for example the enterprise models of a company) as well as the level of the programming (for example XML models).
- **Technological barriers:** These barriers refer to the incompatibility of information technologies (architecture and platforms, infrastructure, etc.). These problems concern the standards to present, store, exchange, process and communicate data through the use of computers.
- **Organizational barriers:** They relate to the definition of responsibility (who is responsible for what?) and authority (who is authorized to do what?) as well as the incompatibility of organization structures (matrix vs. hierarchical ones, for example).

In addition to them, we have identified a set of requirements as the essential and primary requirements since they have been repeatedly identified and reported by other frameworks, initiatives and stakeholders to address interoperability. We recall them here as following.

DR1.Data Standardization and Portability: In general, it aims at the harmonization of data specifications and representation formats, the unification of data communication protocols/interfaces and the convergence of database models and systems. This, in turn, makes data coming from various systems portable and compatible with other systems, and leads to an interoperable ecosystem.

DR2.Data Accessibility and Openness: It highlights the importance of encouraging and practicing free access to data. In general, Data Accessibility and Openness include two concepts, namely Legally and Technically open data. The former refers to increasing the accessibility of data by placing them in the public domain with minimal restriction, while the latter means that data should be openly discoverable, assessable, processable, and re-usable.

DR9.Security and Privacy: In general, it refers to the requirement of keeping data safe and secure, and to make each piece of information only available for authorized entities.

SeR1.Service Efficiency: How well a service utilizes available resources.

SeR7.Service Standardization: It refers to the need to develop and publish services that adhere to some standard to ease their invocation.

SyR4.Guidelines: This requirement covers two different categories of audiences: firstly, end-users, through the provision of comprehensive instructions for them to engage with the system; secondly, business partners, potential followers and any interested party who might enhance the

system in future, through the provision of generic rules and recommendations to facilitate and direct them.

SyR13.Technological neutrality and system/infrastructure harmonization: As the name suggests, this requirement highlights the lack of standardization in the lower layer of the technology stack and the need to decoupling the services/functions provided by a system from the underlying enabling technologies.

4.2 IF ARCHITECTURE

Figure 3 depicts the high level schematic view of IF which is centered around the concept of Asset (Section 4.2.2), and it is composed of a central block holding the Asset Manager (AM) component along with a group of its sub-components (Section 0), the logical Data Abstraction and the Service Abstraction layer (Section 4.2.1).

The identification of the main challenges and requirements greatly helped us in the design of the architecture from the early stages. Accordingly, the three blocks of IF were anticipated to deal with the three categories of requirements mentioned in the previous section. In this direction, AM is mainly in focusing on the **System Management Requirements**, while Data Abstraction and Service Abstraction are collecting the IF components that are mainly dealing with the **Data** and **Service Management Requirements**. Please note that Data and Service blocks are a logical representation of groups of components that are collectively working on a particular purpose. Hence the distinction is not exclusive, but a component might be considered as part of both service and data abstraction. Moreover, to foster the reusability principle, in many cases IF components are using the functionality of other sub-components. In other words, we could perceive IF as its central component - AM- that has some service abstractions and some data abstractions aspects. In this regard, all of IF components are partially contributing to combating the **Organizational, Technological** and **Conceptual** Barriers.

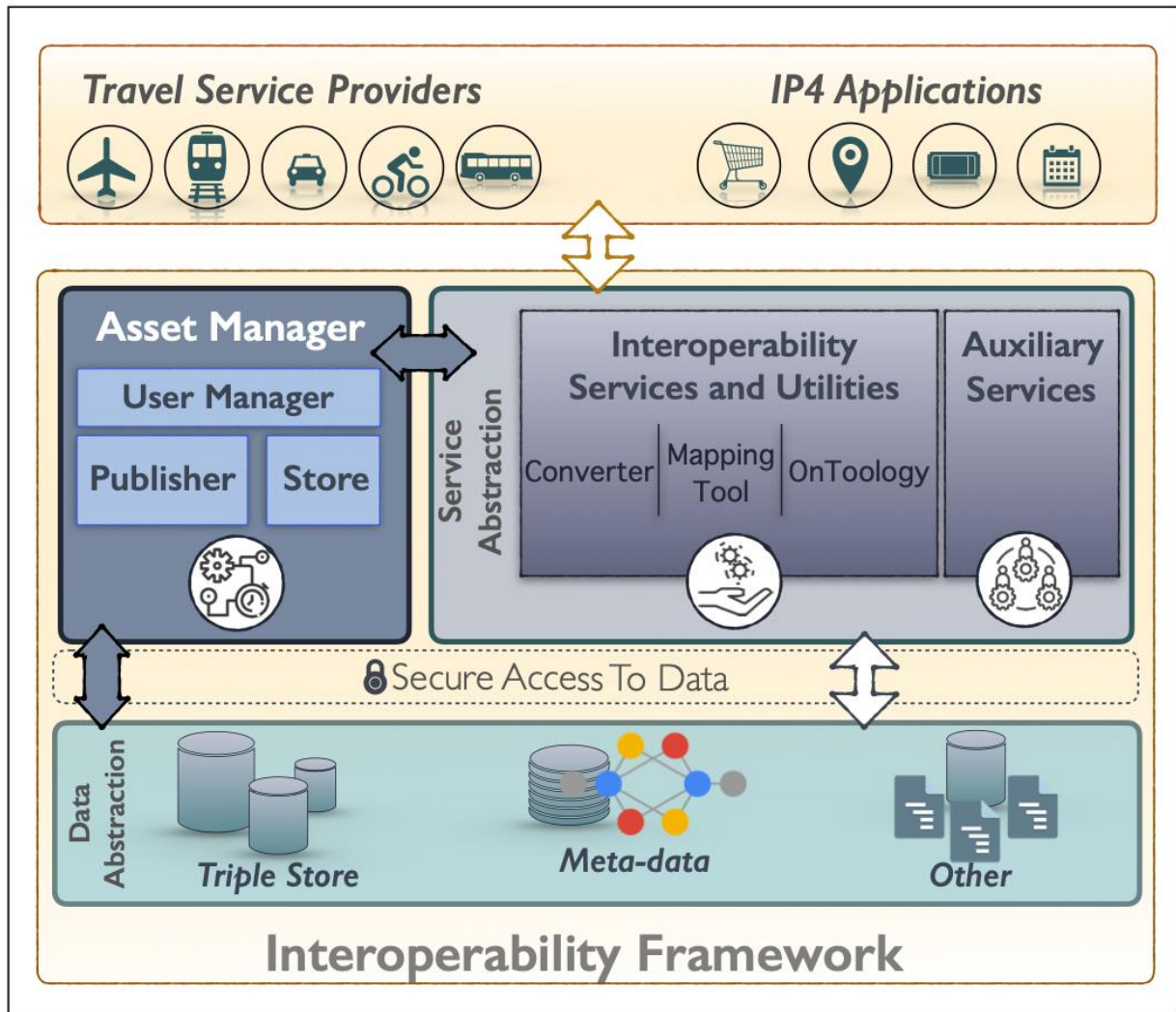


Figure 3 Schematic IF Architecture

4.2.1 Data and Service Abstraction

More specifically, the data abstraction layer is referred to back-end databases and triple data stores, and front-end interfaces and mechanisms (managed by Asset Manager) responsible for the handling of operations such as linked-data and meta-data creation and management to address **DR2.Data Accessibility and Openness**. As well as the collection, storage, and retrieval of data, ontologies, vocabularies and meta-data provided different parties to address and/or created by internal components of IF to address R8, **DR1.Data Standardization and Portability**, **DR2.Data Accessibility and Openness** and **DR6.Machin Readability**.

Service Abstraction then represents those IF components/aspects in charge of dealing with service management requirements, and in specific with a focus on **SyR6-Service Efficiency** (and automation), **SyR7. Service Standardization**, **SyR2. Discoverability**, and **SeR9. User-centricity of service design and implementation**. In addition, the Service Abstraction emphasizes on micro-service-oriented feature of IF architecture. As comprehensively discussed in previous deliverables (SPRINT Deliverable D.3.1 and D.3.3) where we have analyzed various architectural alternatives and enabling technologies for IF, we have concluded that the best approach to design and develop

IF is through modular software architecture. Our design of IF structure hence follows a service-oriented architecture that consists of a set of self-contained, reusable, composable and extendible interoperability services.

In this direction, **Interoperability Services** in IF are the special-purpose operations and processes to shield the heterogeneous and distributed nature of the transportation domain and facilitate the co-operation among them. These services address the generic and primary interoperability requirements (mainly WRT Service Management requirements and the Technological Barriers) that are shared among various actors of the transport ecosystem and might be utilized by them in different manners and to accomplish different goals and applications.

In addition to them, with the aim of fostering a collaborative transportation ecosystem, IF architecture has been designed to be extendible by any interested party. Any registered contributor (See User Management Section), can contribute to developing the pool of interoperability services. Such services are called **Auxiliary Services** in the sense that they are not an essential part of IF, but people can voluntarily add their services and let others use them. This particular aspect of IF is in line with system management requirements such as **SyR5- Integration of complementary services** and **SyR8.Reusability**.

4.2.2 Asset

An asset is an artifact/resource that has a lifecycle, some descriptions and a type that determines the states of its lifecycle and the specification of its description. Such artefact is associated with some permissions, it is discoverable and could be accessed, read, shared and utilized by any interested users and other assets. In the scope of IF we have defined various types of assets, where users can register assets of that type by providing the necessary substances (for more detail please refer to SPRINT deliverable D3.3). Furthermore, to achieve utter completeness and in direction to address requirements such as user-centricity, IF offers the possibility of extending the types and creating a new asset type.

As represented in Figure 4 which is an updated version of the asset domain model introduced in SPRINT deliverable D3.3, an asset has supplemented with the concept of permission (See section 0) and a new asset type named parametric search query (See Distributed SPARQL Query and Exploration API). To briefly recap, the other three type of assets are Data, Utilities and Component. The former, besides the asset descriptions and meta-data, is part of the materialization of Data Abstraction and it includes any kind of data in the transportation domain (E.g., fares data, logistics data, code lists, ontologies, ticketing and payment data, historical mobility data, etc.). On the other hand, the Utilities and Components are the realizations of the Service Abstraction. In other words, Utility Assets (e.g., Ontology Editor,

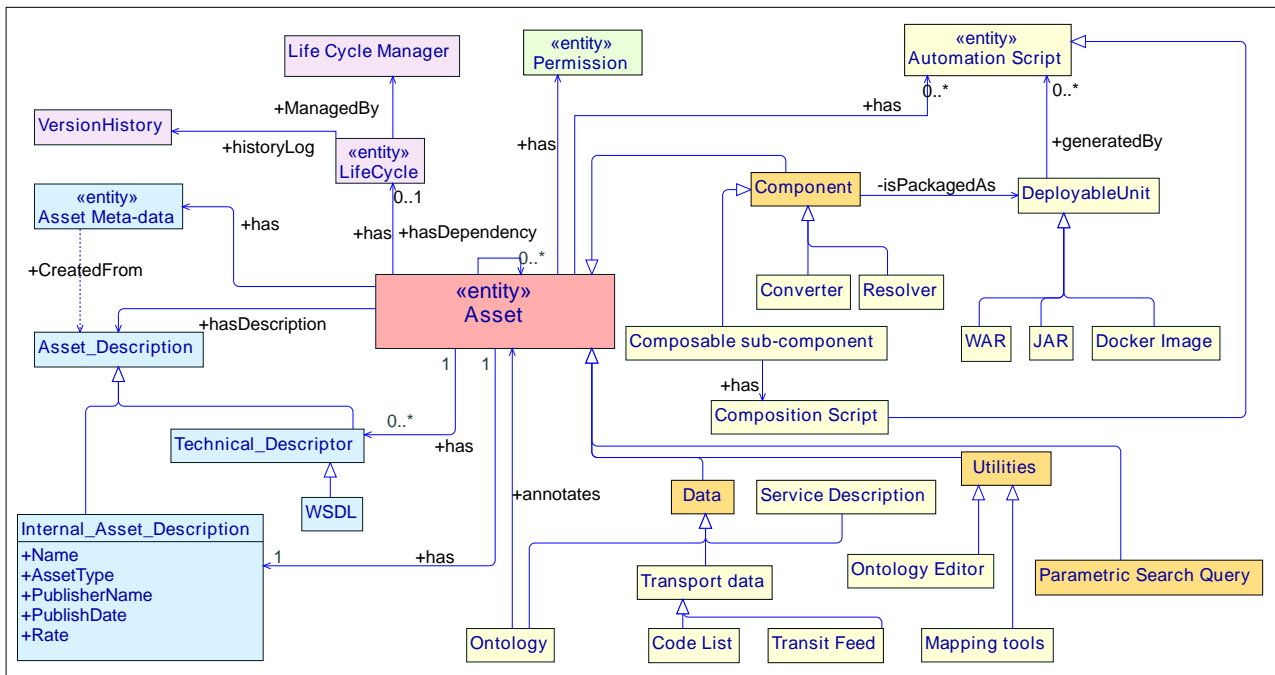


Figure 4 Asset domain model

Mapping IDE explained further down) and Components Assets (e.g., Converter explained further down) are tools and services to enhance the interoperability, that might be provided and utilized by external actors as well as IF itself.

Finally, IF enables the automation of the whole lifecycle of a Component from the composition to the deployment stage. It is a great contribution to IF address **SyR13. Technological neutrality and system/infrastructure harmonization** and to foster the **automation** which plays a key role in the fulfillment of **SeR6.Service Efficiency**, **SyR11.System Efficiency**.

4.2.3 Asset Manager

AM plays the role of the official Catalogue of IF artifacts and it is mainly in charge basic functionality of IF that includes publishing, sharing, discovering, maintaining and managing various artefacts that might be published/utilized by external and internal components of IF. AM hence constitutes the initial point of interaction with the IF from the external client point of view, and it is the key element that interconnects different components and layers of the IF from an internal perspective. As shown in Figure 5, the Asset Manager is composed of sub-component including **Store Frontend** and **Publisher**. AM then offers two distinct web user interfaces, the publisher for the contributors/providers and the store frontend interfaces for common consumer users.

Publisher

In specific, the main operations of the **Publisher** component are grouped into three main interfaces called **Dashboard**, **Process** and **Tasks**.

As represented in the figure, Dashboard interface provides the user (i.e., the contributor) with basic management of their assets such as registering new asset/service, creating new asset type, viewing all of their own assets, the status of the assets (e.g., published, waiting for confirmation) with viewing and editing asset meta-data, etc. Similarly, Task and Process interfaces are offering other contributor-related operations such as the lifecycle management of assets and managing the access requests initiated by consumers (to accept or reject). Publisher component itself has several sub-components, some of which are used directly by users and others by other sub-components of IF. For example, the life-cycle manager component is the one in charge of managing most of the operations offered by the process interface, while the Distributed SPARQL query component is mainly employed indirectly and thorough other components.

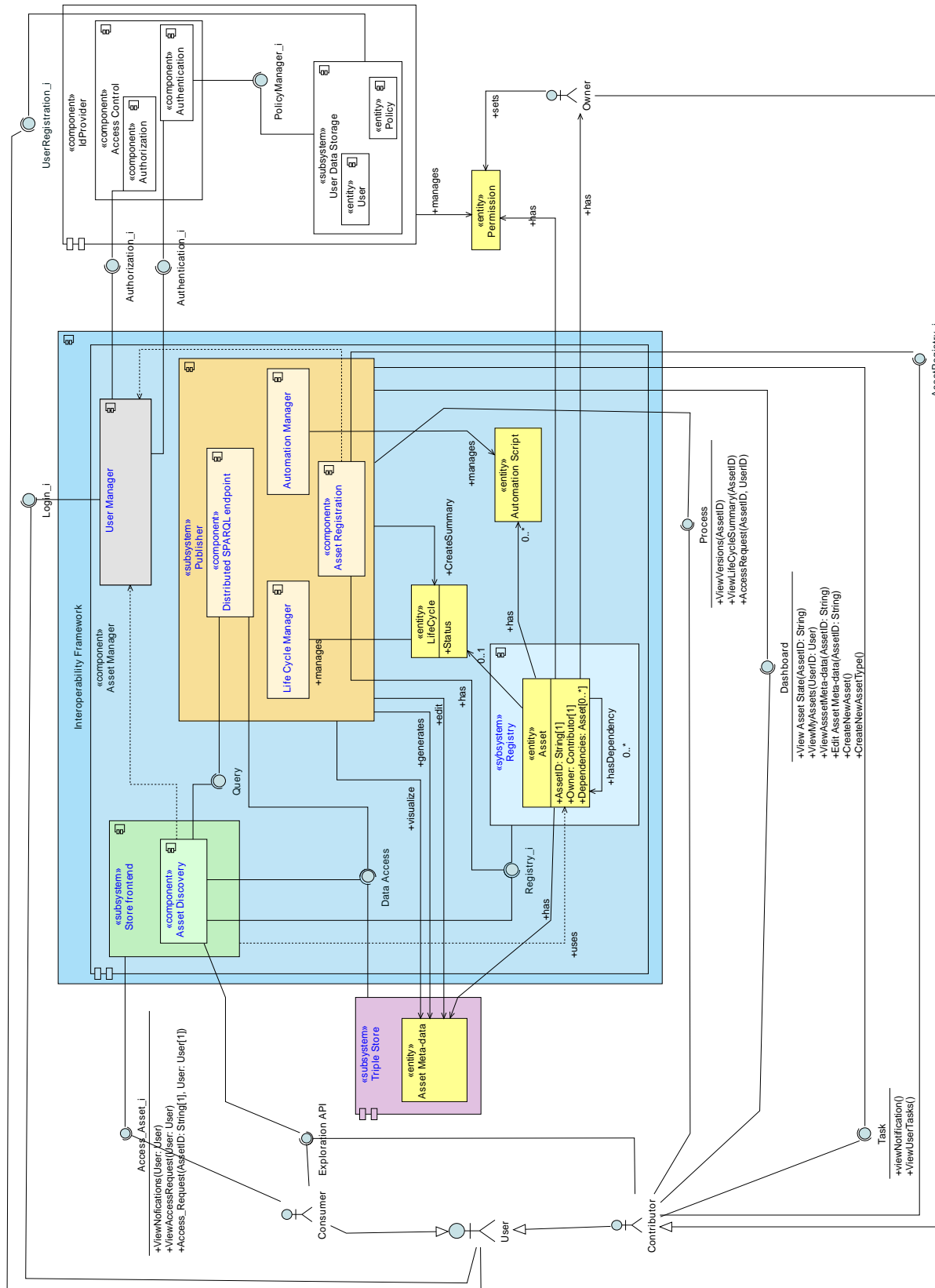


Figure 5 IF Architecture

Store

Through Store interface, users can explore the assets which are already registered and published by providers. It provides users with a powerful searching capability to discover assets based on various criteria. Asset Manager automatically transforms the descriptions of an asset - supplied by the owner of the asset in the registration time- to RDF (represented as “Asset Meta-data” in Figure 4) and stores them in the available RDF Repositories which are later used by Distributed SPARQL engine to enable a federated and semantic-aware assets discovery. Upon Asset discovery, user can view the generic asset description and can fully access it if the asset is public and open; Otherwise, a user required to take the permission from the asset owner following the process explained in User Manager section.

Life-cycle Management

is a sub-component of publisher, to let users define and manage various stages for each asset type in IF and in general to define some sort of executable process. The component employs the BPMN formalism that offers a graphical means to define desired life cycle and processes per a specific asset type. In other words, thanks to this component, each asset in IF could be associated to specific set of processes and tasks (for human and services) that would be triggered in some defined orders. Definition and assignment of such processes are totally configurable through a user-friendly way. As a sample procedure, we have defined a Converter asset, in which users should provide a script to define the source and target standards and a couple of other inputs, as the first stage in its life cycle. Upon submission of the scrip, another component of IF, the **Automation Manager**, would be triggered to actually synthesis the described converter and ultimately publish it.

Distributed SPARQL Query and Exploration API

The asset discovery capability can be supported by the Distributed SPARQL Endpoint and the Asset Manager by means of Exploration API. An Exploration API is a parametric SPARQL query over a SPARQL endpoint, which is automatically exposed by the Asset Manager as an API transforming the parameters of the query onto the parameters of an HTTP GET API call. The underlying SPARQL endpoint can be the RDF metadata repository which is directly managed by the Asset Manager, or an external endpoint provided, for instance, by a Distributed SPARQL endpoint component. An Exploration API is added to the Asset Manager as a “Parametric Search Query” asset (as shown in Figure 4), and then the Asset Manager exposes it as an API and documents its parameters using OpenAPI/Swagger.

The Distributed SPARQL Endpoint is the component in charge of evaluating queries over a set of triple stores providing a unified access of data or a knowledge graph. The user specifies SPARQL queries over the knowledge graph and when a SPARQL query is received, the Distributed SPARQL Endpoint first identifies which triple stores will be used to answer the SPARQL query through the mappings between the knowledge graph and its triple stores. To transform a SPARQL query, several sub-queries are generated to be evaluated over each triple store and also a query plan is created with the order in which will be executed these sub-queries. Afterward, each sub-query is finally executed by each selected triple store, and the results obtained for each subquery are integrated and returned.

4.2.4 User Management

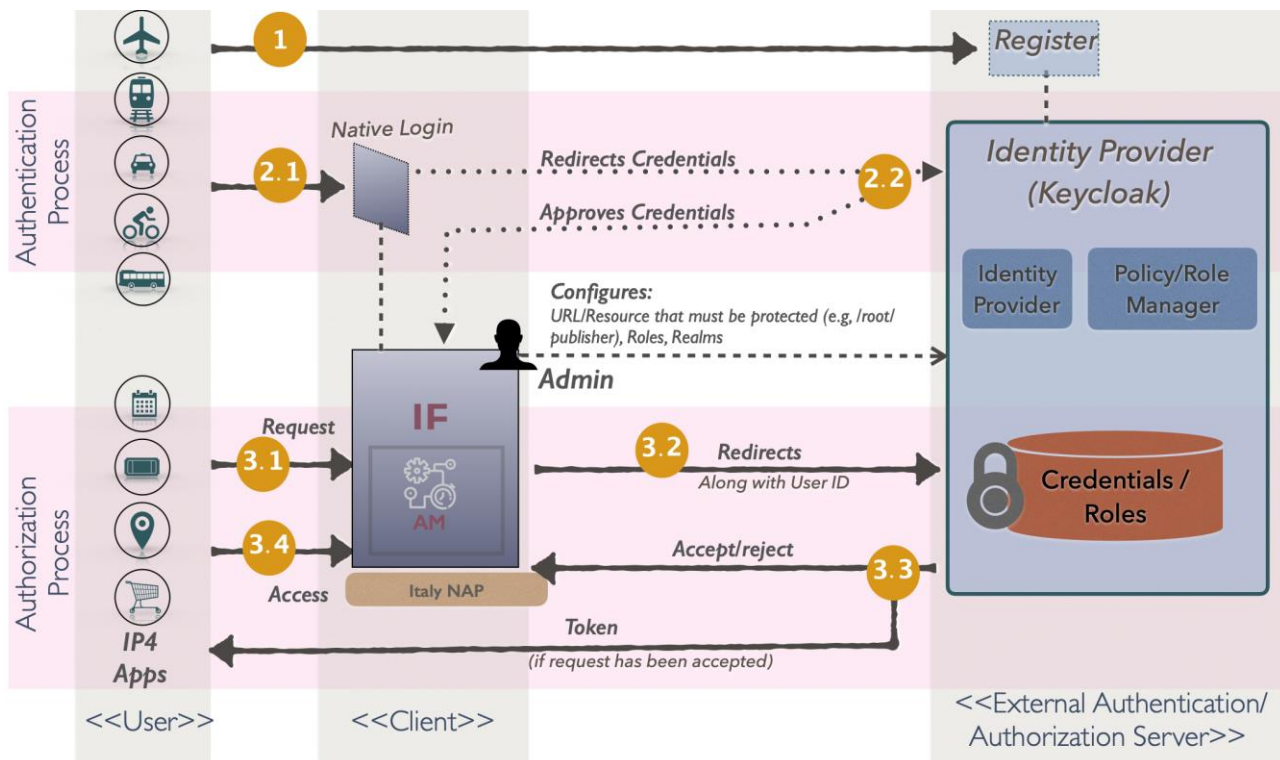


Figure 6 IF and External Identity Provider

The user manager is in charge of handling User entity's related aspects including the registration, login, assignment of roles and access rights to interact with an Asset. As explained previously, the offered functions of AM are tailored based on the nature/intention of the user to interact with IF. In specific, Publisher is the endpoint for Contributors while Consumers are only concerned with Store. To this end, IF required to assign and keep track of the Roles of users who are communicating with IF. Furthermore, in reference to **DR9. Security and Privacy** and **SyR2.Authorization and Authentication mechanisms** which stem from the fact that transportation actors are concerned with the protection of their data/assets IF has anticipated an intuitive role-based access control. The owner of an asset has the possibility to configure an asset as a fully open or a protected one. For the latter, such asset would be discoverable through Exploration API but user can only view the generic asset description. To obtain full access an interested user should explicitly send an access request to owner and then it is up to owner to accept or reject such request.

In this direction, as depicted in Figure 6, IF employs the concept of Open ID protocol where an external Identity provider has the responsibility of storing the users' credentials, their roles and their permissions related to actions to be performed onto specific asset types. The Identity Provider performs both authentication and authorization. This design facilitates joining to IF-ecosystem (in direction with addressing the **SyR1. Administrative simplification** requirement) and such component thus becomes central in the deployment of an IF-based ecosystem, since it is accessed by any component which needs to understand whether a user is allowed to access a functionality.

4.2.5 Interoperability Services and Utilities

Converter

Converters act as adapter services between two distinct formats and are able to map the information expressed in one format to the other (as described in SPRINT deliverable D5.2). They are the actual executors of the semantic interoperability principles, performing a conversion which first “lifts” the incoming data from its format to a graph according to an ontology, and then “lowers” the data, extracting the right contents from the graph to build the destination message. The concept of a “conversion pipeline” allowed us to break down the overall process in a set of processing blocks which can be arranged, connected and configured independently (or omitted in case they’re not necessary). Using a declarative approach, we can therefore design complex conversion processes and package them as either mediation services or as batch converters.

Mapping Tool

Mapping tool is other utility of IF that could be discovered and deployed as a stand-alone software. Its application domain is the conversion of any piece of data from one standard to another standard. Given that one of the inputs to the Converter in IF are the one-to-one mapping between concepts/terms of the target standard and ontology, the Mapping tool has been designed to drive such mapping in an automated manner. To this end, it exploits the syntax and structure of the standards as well as the linguistic semantic of the terms of such standards. By applying some machine-learning technique the tool learns the similarity among concepts based on their semantic/meaning to drive with the pair of terms as the equivalent concepts in two given standards. It then enhances such suggestion using the information that could be extracted from structure of the standard based on syntactical role/position of the term in that standard (E.g., class in ontology and Complex Element in XSD). For more details about the architecture and implementation of Mapping tool please refer to SPRINT deliverable D4.2 and D5.3.

OnToology

OnToology is a Collaborative Ontology Management utility of IF. Particularly, OnToology is a specialized tool that implements a continuous integration pipeline for ontology development. A continuous integration pipeline for ontology development allows ontology developers to integrate ontologies into a shared repository several times a day. Each check-in is verified by an automated build process, allowing developers to detect any problems early. The pipeline process is triggered when the ontology files are committed to a repository, e.g. a GitHub repository. Then OnToology compiles the files and runs evaluation, documentation, etc. Once tests are evaluated successfully, the ontology can be published. In addition, a continuous integration pipeline can be configured by means of a file called Jenkins file, which is part of the infrastructure provided in the Shift2Rail Asset Manager. For more details about the architecture and implementation of Collaborative Ontology Management tool please refer to SPRINT deliverable D4.2.

XSD2OWL

XSD2OWL is a utility of IF that automatically generates ontologies from non-ontological sources. Considering the fact that many of the ontologies to be developed for the Shift2Rail ontology network are strongly based on existing XML Schemas that have been used for the exchange of data across multiple systems, IF ecosystem can benefit from semi-automating some of the steps involved in the ontology development process.

To perform transformations from XML Schemas (XSD) to RDF/OWL automatically, XSD2OWL can be used to create RDF/OWL representation of XML Schemas, and XML instances that comply with such XML Schemas. Once the XSD2OWL system is run on this XML Schema file, the set of classes and properties presented are generated and it produces OWL ontologies that make explicit the semantics of the corresponding XML Schemas. For more details about the architecture and implementation of Automatic Ontology Generation tool please refer to SPRINT deliverable D4.2.

4.3 DEPLOYMENT STRATEGIES

Another prominent feature of IF is its compatibility to deal with various conditions and circumstances and a diverse range of application domain and use-cases. Figure 5 depicts different packaging and deployment possibilities to engage with any type of assets and utilities in IF. The provision of multiple deployment strategies is an IF contribution for addressing **SyR13.Technological neutrality and system/infrastructure harmonization** requirement. Since the customization of the deployment enables seamless integration with IF, regardless of the consumer system and infrastructure.

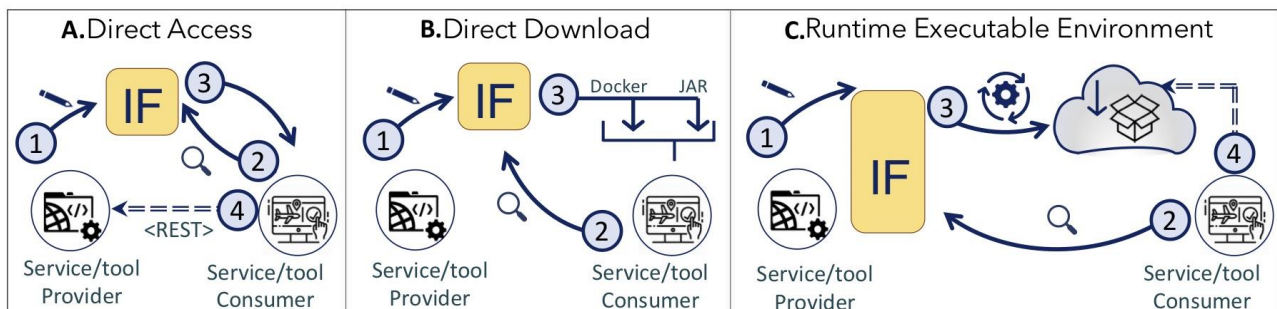


Figure 7 IF Deployment Possibilities

4.3.1 Ready to use services, tools and utilities

In a nutshell IF offers three procedures to deploy registered assets in IF, namely Direct Access, Direct Download and Runtime executable environment. In following we briefly overview each approach that have been described in detail in Deliverable D3.2.

The simplest and most loosely-coupled approach for engaging with IF, is **Direct Access** that follows the methodology service-oriented architecture. As represented in Figure 5, the procedure starts when a publisher registers a service of its own to IF along with necessary descriptions and the endpoint to reach service/API). AM then automatically creates an RDF meta-data out of such description and stores it in its repository which in turn makes the service discoverable by Distributed SPARQL endpoint. In consumer end, after the discovery of the desired service, they can simply follow the link to the service/API.

The next approach is **Direct Download**, through that users can download a deployable package of a tool. As represented in Figure 3, an asset in IF could be encapsulated to different deployment artefact such as container image, JAR and WAR file. Former is suitable for clients that prefer a micro-service-based solution while the latter is the best option for conventional service-oriented architecture. It is up to the publisher to wrap up a service/tool, in a particular downloadable and runnable package and upload such a package to IF. Then, through a similar approach to the direct

access, users can search for the desired tool via the Exploration API as well as simple search capability of the Store interface to find and download an asset.

The last approach is called **Runtime executable environment** that offers the client not only the service but an executable environment to directly run and deploy it. Following a plat-form-as-a-service approach IF is establish itself as more than a mere service registry. The service publisher, or, any provider should supply IF to utilize an external cloud environment based in its need and with its own authority. So, after the discovery process, the users can request IF to deploy the service and return to them the endpoint to directly consume the service. The running of the service on such an environment is temporary and would be terminated according to the user and IF agreement. Furthermore, IF supports autoscaling of cloud deployment exploiting Kubernetes technology which is a cloud orchestrator for managing the containerized workloads. AM provides features for the user to configure the Kubernetes POD by specifying metrics such as CPU utilization for a specific containerized asset.

4.3.2 Continues Development and Deployment

In addition to the registration and publication of ready-to-use components, services and tools, the Asset Manager offers the possibility of creating a new component, on the fly, by assembling already available assets. The Automation Manager component in Figure 4, integrates a continuous integration/delivery tool (Jenkins) to materialize the concept of Build automation -that is referred as the automatic creation of a software build and its verification against predetermined tests- in IF. This feature of IF is designed to leverage the generic automation requirement which is part of the main goals of SPRINT projects and greatly contributes to addressing requirements such as **SyR5. Integration of complementary services**, **SyR8. Reusability** and **SyR13. Technological neutrality and system/infrastructure harmonization** and more technical details of this feature are reported in SPRINT deliverables D4.1, D4.2 and D5.2.

In short, through this approach users can submit a description of automated creation of an asset. It contains the URL to all the required assets to build this new asset as well as a Jenkins job script. AM then fetches all the stipulated elements, synthesize them to create a stand-alone asset and wrap it up to a deployable and runnable package and expose it to the Store interface as any other assets. It is then a new asset which is discoverable and accessible by any interested user.

5. TESTING INFRASTRUCTURE

The design of IF testing infrastructure aims to measure the performance and scalability for each IF component and the whole IF system to analyze both components and the system in load situations. For the whole section, we exemplify the proposal of the testing infrastructure with the Distributed SPARQL query component that is part of the IF. The section describes: 1) how to define an overview of the test cases for all the components of the IF; 2) a checklist of recommendations for testing the component of the IF and; 3) provides the whole workflow of the testing infrastructure for performance and scalability, together with a set of representative examples.

5.1 DEFINING TEST CASES FOR THE IF

In the process of the definition of the test cases for the components of the IF, it is important to have an overview of what of the components are being tested in each evaluation. For this reason, and before starting with definition of each component test cases set, an overview of the test cases has to be provided. To help in this process, and the first step element of this testing infrastructure is a set of tables that correlates the feature to be tested with a component, together with relevant information such as the related user story of the requirements to be tested. These tables will be useful to have a first overview (like a snapshot) of what has been tested at each time and what were the requirements and accepted criteria of each component, which will help to compare how these features change over the time but without giving any technical detail.

<<Feature>> (e.g. Performance or Scalability) Testing for <<Component>> (e.g. Converter)	
Process	Brief description of the activity performed
Related User Story	Which User Story is it linked to?
Performance/Scalability Requirements	Write requirements identifier here
Performance/Scalability Criteria	Performance: <i>expected workload tested</i> Scalability: <i>what is “scaled”</i>

5.2 TEST CHECKLIST

The checklist reflects a set of recommendations we select to define an objective and representative testing of any component of the IF. This list is a recommendation and should be adapted to specific cases:

1. **Select a repeatable process for conducting your tests during the application's lifecycle:** identify what is the frequently executed process on which the performance/scalability will be measured.
2. **Define performance/scalability criteria for your tests:** Performance measures how fast and efficiently a software system can complete certain computing tasks, while scalability measures the trend of performance with increasing load
3. **Prepare a set of tools to run your tests:** Define which are all the tools necessary to execute the tests.
4. **Define the testing environment and configure any hardware you need to perform tests:** Define a setup of software and hardware for the testing teams to execute test cases.
5. **Plan your test scenarios:** Create a plan with the functionalities to be tested.
6. **Create and verify testing scripts and load test scenarios** (e.g. by using JMeter): By using a test tool like JMeter, we can build a test plan that will be checked locally before being deployed on the server.
7. **Execute your load tests.** (e.g. with Jenkins)
8. **Analyze your results and generate reports.**

In order to help in the definition of the checklist, we provide a table where the testing users can provide the information of the component they want to test, for example in Table 9 we show the same example as before (Distributed SPARQL query).

Performance and Scalability Testing for Distributed SPARQL query		
Process		SPARQL query over multiple endpoints
Performance/Scalability Criteria		Performance: increase the number of endpoints Scalability: number of requests per second
Preparation		S.O. Ubuntu, Apache JMeter, Ontario, Docker, docker-compose
Testing (setup)	Environment	Ubuntu 18.04, Jenkins v2.222.4, JMeter v5.3, Ontario v0.3, Docker version 19.03.6, docker-compose version 1.23.1

Test Scenarios	US-2,UCS-4 (Distributed SPARQL endpoint)
Testing Scripts	JMeter generates a Test Plan.jmx from our Use Case ¹ .
Execution	The generated plan is deployed and executed with the Jenkins server
Analyzing results	The average, minimum, maximum, standard deviation and error percentage of the execution time for each query were obtained. Based on these statistics, an analysis can be conducted.

Table 9 An example of a Test Checklist

5.3 PERFORMANCE AND SCALABILITY TESTING INFRASTRUCTURE

To evaluate the performance and scalability of the IF components, we have designed the following workflow that involves the requirement elicitation and automatization of test cases using tools that we subsequently recommend. In Figure 8, we can observe the workflow to evaluate the IF components independently of the technology used, thus separating our design from the technology considering that the technology always evolves and changes. The approach of this workflow is based on the generalization of the testing process which is very complex and will depend on each tool and component to be tested. After that, we will detail the steps and the tools we will recommend and we illustrate by means of some examples specific technologies that we will use. However, after explaining this workflow, we will mention and recommend several tools, such as Docker, JMeter, and Jenkins, that can be used to perform tests.

¹ For example: <https://github.com/jatoledo/Ontario/blob/master/Ontario.jmx>

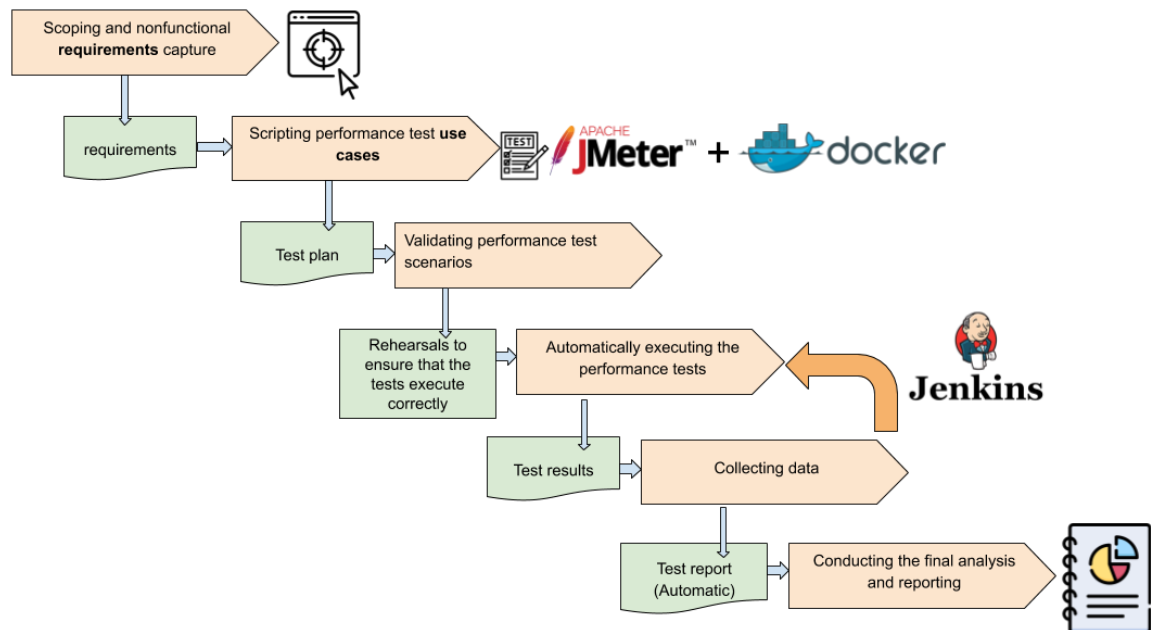


Figure 8 Design of Testing Infrastructure

We describe more in detail the steps we propose to run a performance and scalability test for the IF components.

5.3.1 Scoping and non-functional requirements capture

For each component, all nonfunctional requirements about performance and scalability are specified in this phase. In this part, we can include the performance/scalability requirements specific to each of the developed components. At minimum level, this phase must:

- Design the test environment trying to approximate it to a real environment. This test environment must not be affected by other user/server activities.
- Identify, document, and script key uses cases
- Identify parts of each use case that should be monitored separately
- Determine input, target, and data requirements for the key use cases
- Identify performance test scenarios based on the number, type, use-case content, and virtual user deployment.
- Identify and document KPIs
- Develop a high-level plan that includes resources, timelines, and milestones based on performance/scalability requirements

- Develop a detailed performance test plan that includes detailed scenarios and use cases, load models, and environment information

5.3.2 Scripting performance test use cases

This step includes preparation and setup of the test environment, and test scripting.

Preparation of the test environment consists of identifying the hardware, software, and network requirements for the test environment. We need to create a test environment as a close approximation of the production environment. At a minimum, your setup must reflect the IF component deployment of the production environment and the target database or dataset must be realistically populated in terms of both data content and sizing.

Setup of the test environment involves selecting the software to be used in order to indicate the version of the tools that will be used for IF component. In this part, we must choose the tools that best meet the requirements of the tool to be tested. For example, we have chosen Python during the preparation of the test environment, but we need to indicate which version of Python we will use, Python v2 or Python v3. In this way, we can better determine the possible errors during the testing process or incompatibility of versions in some cases. Also, we must install and check the test tools with the IF component to be tested.

Finally, **Test Scripting** implies the test plans are determined and then scripts are created to automate the planned tests from the defined scenarios related to KPIs. One tool that can be used to automate or create these scripts is JMeter. In addition, test execution must be of validating your component performance and scalability targets.

5.3.3 Validating performance test scenarios

Here, we must perform rehearsals to ensure that the tests execute correctly. To validate test correctness, we must define a baseline or a basis of comparison to test how the performance improves over time. This baseline will be executed right before the current test so that baseline and current test are equally affected by any factors on the server. It is important to decide what is the baseline for each IF component in order to assess both its performance and scalability. We must also ensure that the component is correctly deployed into the test environment.

5.3.4 Automatically executing the performance tests

Test automation will ensure each test or subsequent re-test is run in the same manner. To automate our tests, we can use JMeter and deploy it on Jenkins, although other tools can also be used to test the IF components in the same way (e.g., Selenium²), we recommend the use of these two tools when it is possible.

² <https://www.selenium.dev/>

5.3.5 Collecting data

In all our tests, we ensure to monopolize the hardware the test is running on to avoid unexpected results. This is achieved by using docker images to deploy the IF components and then removing the images for each test so as not to interfere with the performance and scalability results. For creating the docker images of each component, we recommend to use the following template for a dockerfile where the folder of resourceN defines the location of the different resources the engine has to use to test its performance and scalability and a folder for the possible outputs of the engine:

```
FROM ubuntu:18.04

RUN apt-get update && apt-get install -y vim
RUN mkdir /app
COPY . /app

RUN mkdir /resource1
RUN mkdir /resource2
RUN mkdir /resource3
...
RUN mkdir /resourceN
RUN mkdir /output

CMD ["tail", "-f", "/dev/null"]
```

We assume that the component is packaged and ready to run (e.g., in a JAR file for Java). In the case that our component has multiple services (e.g., Distributed SPARQL query engine needs to access several SPARQL endpoints), they have to be deployed independently. For that case, we can use the Docker Composer tool, a CLI for automating deployments that use a YAML configuration file. The 'services' section contains the containers that will be created to deploy our application. In this case, two will be created: one that we will call "service1" and another that we will call "service2". These names can be defined by the testing user and will serve to identify the services. After that, we need a way to share the resources to be used for testing the component between the docker image (service) and our local host. To do this we define the shared volumes in the 'volumes' section where we define the relation between the path of the resource in the local host (e.g., './service1-resource') and the path inside the docker image (that have to correspond to the path created in the Dockerfile for that resource). The folder output is optional, can be defined in the case of the tested component outputs information that could be useful (e.g., the number of results of a SPARQL query).

We recommend the following template for a docker-compose.yml:

```
version: "2.3"
services:
  service1:
    container_name: service1
    image: service1
    restart: always
    volumes:
      - ./service1-resource1:/resource1
      - ./service1-resource2:/resource2
      - ./service1-resource3:/resource3
      - ./service1-output:/output

  service2:
    container_name: service2
    image: service2
    restart: always
    volumes:
      - ./service2-resource1:/resource1
      - ./service2-resource2:/resource2
      - ./service2-resource3:/resource3
      - ./service2-output:/output
```

5.3.6 Conducting the final analysis and reporting

This step includes analyzing data from all test runs, creating reports and possibly retesting. You must determine success or failure by comparing your test results to KPI targets set, and subsequently the test results are analyzed and discussed. Finally, you can document the results using your preferred reporting template, but it must include sections for each of the performance targets defined during non-functional requirements capture.

5.4 APPLYING TESTING INFRASTRUCTURE WORKFLOW

The use case that we discuss in this section is related to an expansion in terms of endpoints and a need to explore the capacity limits of the Distributed SPARQL query engine component. The aim is to establish a baseline model for increasing the number of endpoints at the Distributed SPARQL query engine component. In this section, we exemplify each step of our workflow using the Distributed SPARQL endpoint component of the IF, which has Ontario as a tool in the backend.

Scoping and non-functional requirements capture: The only performance test identified is a progressive ramp-up to the number of endpoints. KPIs focused on generic performance metrics.

Thus, we define our KPI for Distributed service/asset discovery as follows:

Requirement	
Referenced User Story	US-2 (Distributed service/asset discovery)
KPI	Asset/Service Discovery Response Time
Definition	Time required to perform a Distributed SPARQL query on two RDF endpoints that publish metadata.
Requirement	The activity of looking for assets in the Asset Manager in a distributed environment. A discovery task implies multiple retries of a simple distributed search operation, each time adding filters to the previous try.
Target Value	The execution time for a Distributed SPARQL query on two endpoints managed by the Asset Manager will be low and it should therefore not exceed 10 seconds to avoid frustrating users during the discovery task.

Scripting performance test use cases: We've divided this step into three sub-steps: Preparation, Setup and Test Scripting. First, in Preparation, we select all software needed to perform tests on our IF component, the Distributed SPARQL query engine. In our test case, we have chosen the following software: Ubuntu as operating system, Apache JMeter as test tool, Jenkins as Test automation tool, Docker and docker-compose for containerizing applications, Ontario as the Distributed SPARQL query engine, DCAT-AP Metadata3 as datasets and Virtuoso as a database engine. Then, once the software needed to measure performance/scalability has been selected, we determine the software versions corresponding for our testing and install them according to the recommendations from the software manufacturers. For our scenario related to the Distributed SPARQL query engine, we have selected Ubuntu 18.04, JMeter v5.3, Ontario v0.3, docker-compose version 1.25.5, Jenkins v2.222.4 and Docker images from Virtuoso 7. Once the setup of the testing environment has been completed, Figure 9 depicts an architecture of the Distributed SPARQL query engine to be tested in our workflow application. We can observe that Ontario is the Distributed SPARQL query engine that receives a SPARQL query to be distributed and executed on multiple metadata catalogues stored by Virtuoso triplestores, and produces results integrated from several catalogues.

³ <https://github.com/cef-oasis/DCAT-AP/tree/master/TransportDCAT-AP>

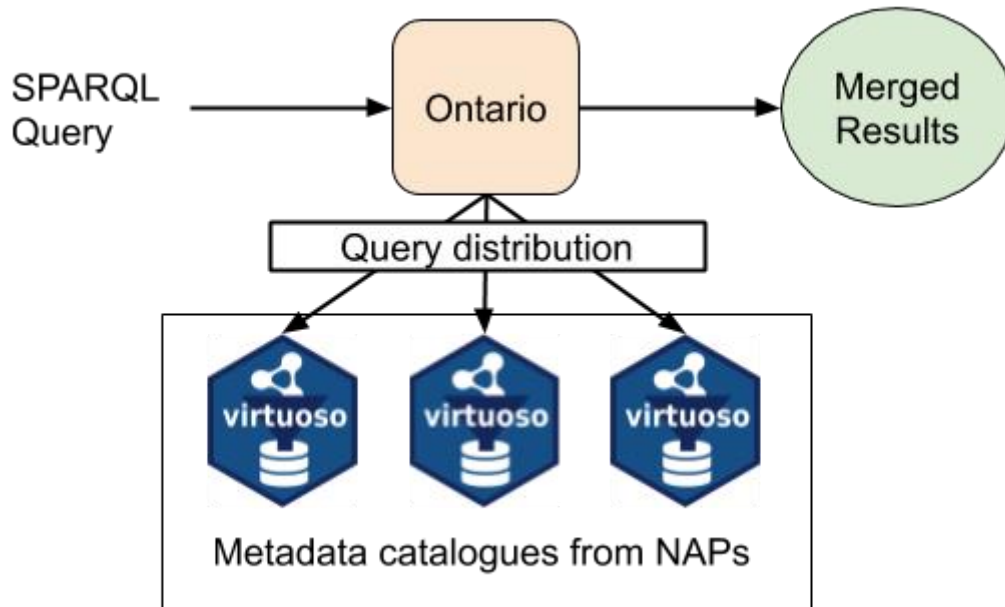


Figure 9 Distributed SPARQL query engine Architecture

This architecture has been built by means of docker images for Virtuoso and Ontario as shown Figure 10. The aim is for Ontario to execute a Distributed SPARQL query on metadata from Belgium and Spain catalogues stored by two Virtuoso triplestores.

```

jtoledo@ubuntu:~/Desktop/apache-jmeter-5.3/bin$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
813232c9d3b0   kemele/ontario:0.3                 "/ontario/start_spar..." 9 days ago    Up 4 hours   0.0.0.0:5001->5000/tcp
cc391be37489   kemele/virtuoso:7-stable            "/bin/bash /virtuoso..." 9 days ago    Up 4 hours   0.0.0.0:1117->1111/tcp, 0.0.0.0:11386->8890/tcp
641b0cc74ec1   kemele/virtuoso:7-stable            "/bin/bash /virtuoso..." 9 days ago    Up 4 hours   0.0.0.0:1116->1111/tcp, 0.0.0.0:11385->8890/tcp
  
```

Figure 10 Docker images for Virtuoso and Ontario

Before performing the **Test Scripting** sub-step, we need to run JMeter as follows:

1. Download Jmeter from <http://apache.uvigo.es/jmeter/binaries/apache-jmeter-5.3.zip>
2. unzip apache-jmeter-5.3.zip
3. cd apache-jmeter-5.3/bin/
4. sh jmeter.sh

After you run `jmeter.sh`, JMeter client should have started as depicted in Figure 11.

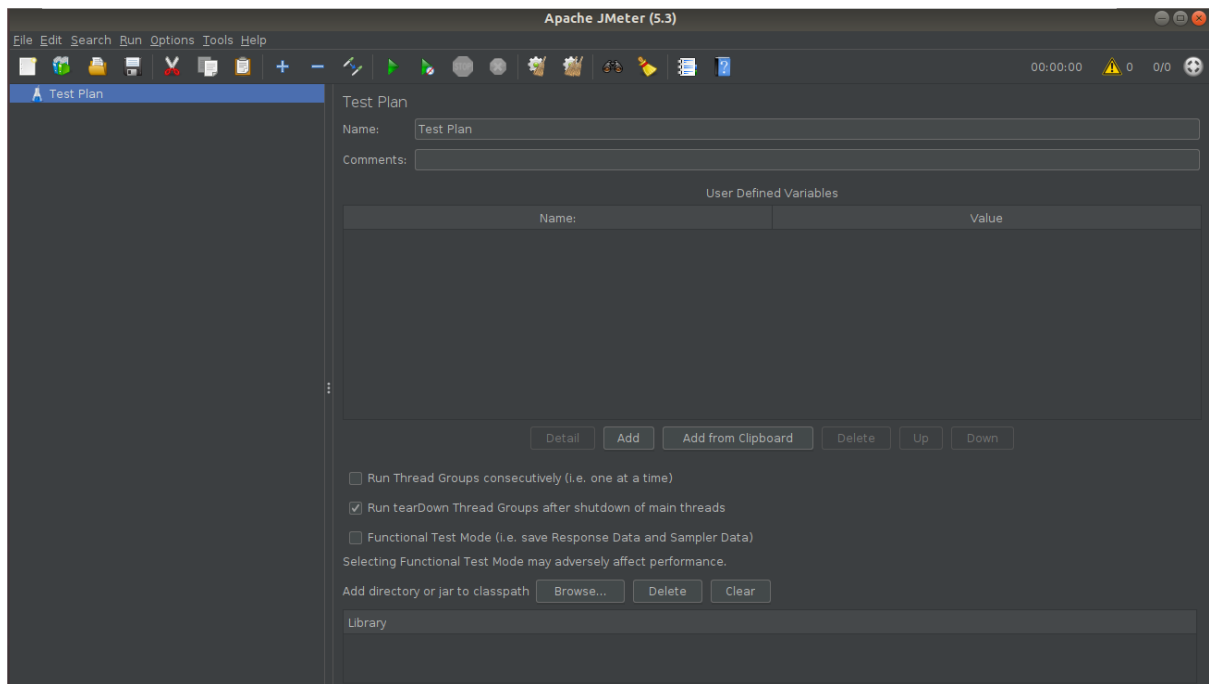


Figure 11 JMeter Client

Finally, we just need to perform the **Test Scripting** sub-step. Once JMeter is started, let's create the script (.jmx) to perform the test:

1. Right-click on Test Plan and create a thread group.

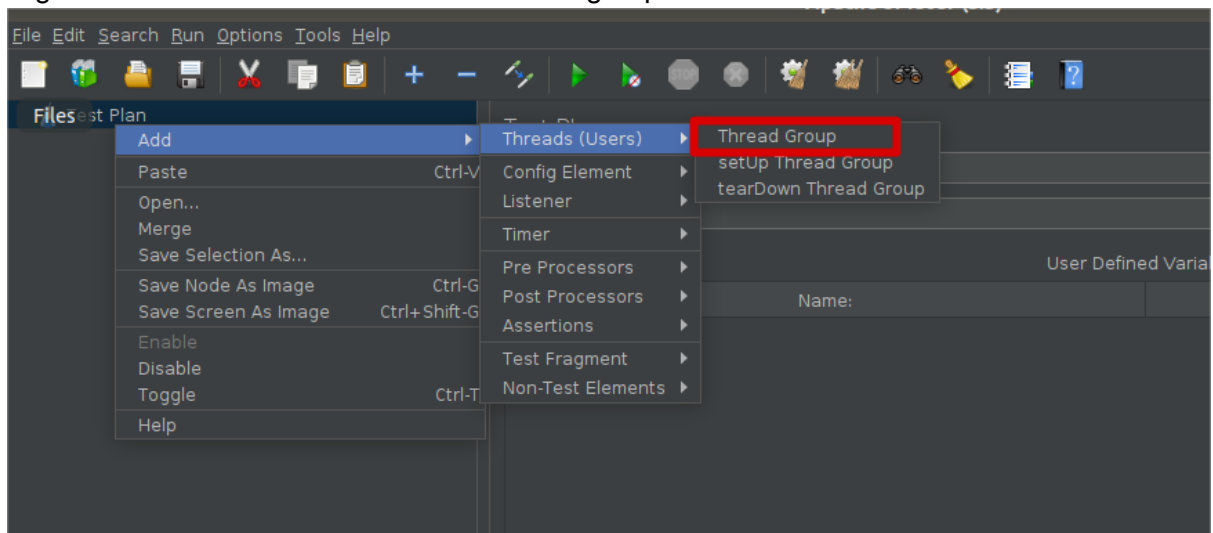


Figure 12 Group creation

2. Select the number threads per second. For example, 30 threads (users) per 5 seconds.

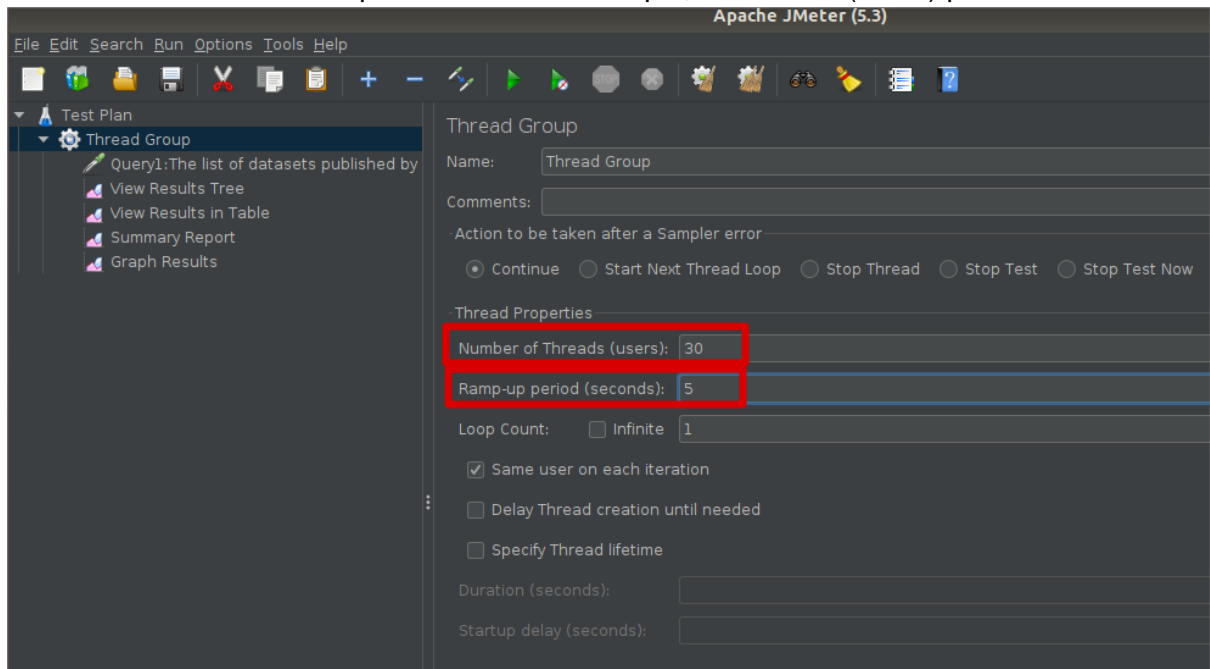


Figure 13 Selecting the number threads per second

3. Right-click on Thread Group to select HTTP Request. We need a HTTP request because we work on the Ontario API.

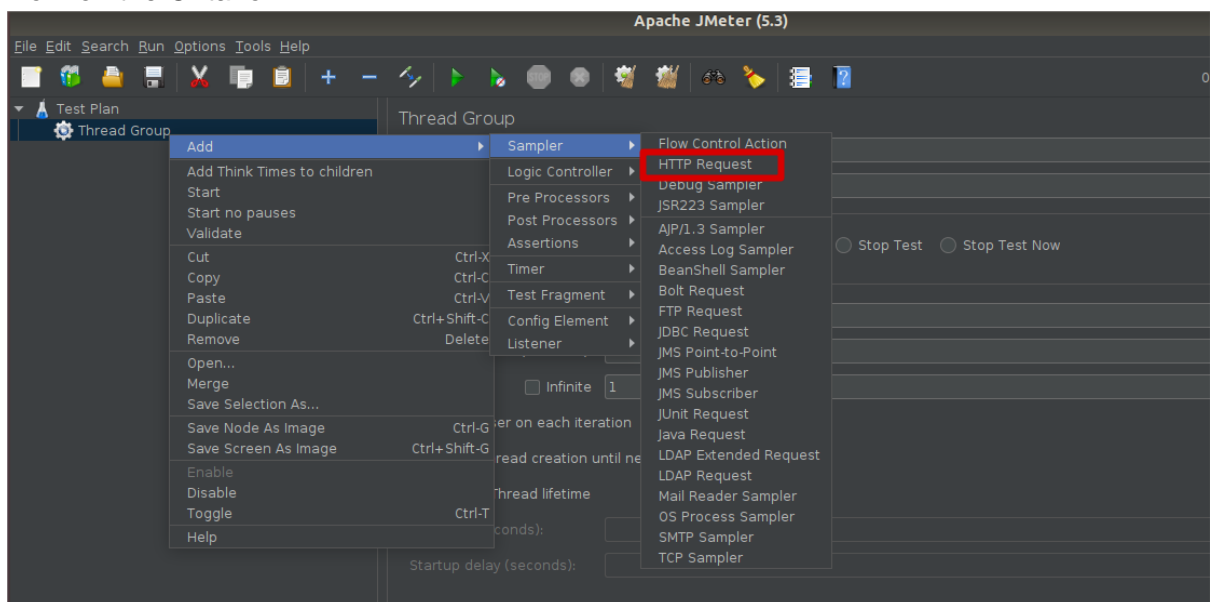


Figure 14 Selecting HTTP Request

4. Fulfill the fields. For example, we configure the fields needed to connect to Ontario API: <http://localhost:5001/sparql>

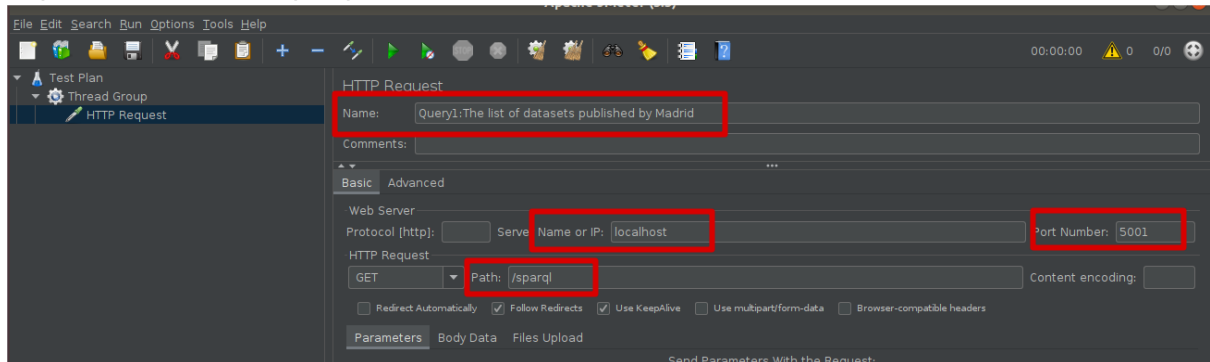


Figure 15 Showing configuration for an Ontario query

5. Then, we must define the queries to be tested in the Value field. One of the queries related to Scenario S4 is query1 which was added and will be executed by the previously configured Ontario endpoint: <http://localhost:5001/sparql>.

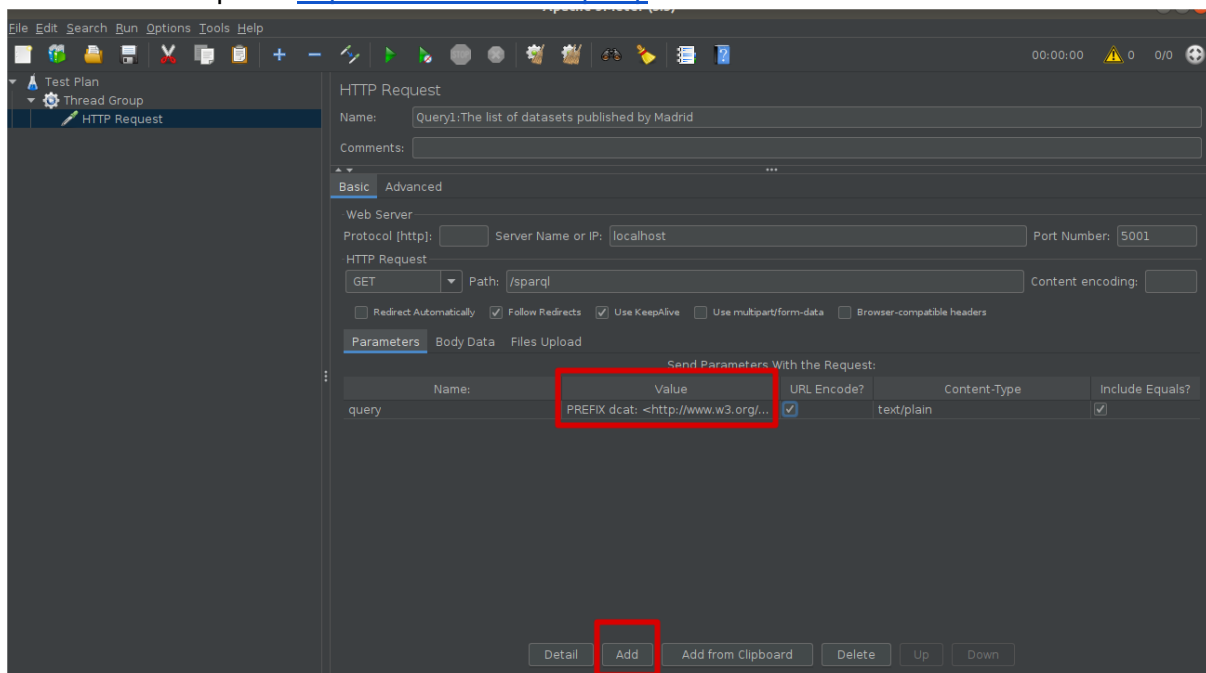


Figure 16 Adding a query

The Query1 is the following:

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX org: http://www.w3.org/ns/org#

SELECT ?dataset WHERE {

  ?DatasetURI a dcat:Catalog .

  ?DatasetURI dcat:dataset ?dataset .

  ?DatasetURI dct:publisher ?crtm .

  ?crtm a org:Organization.

  ?crtm foaf:name "Consortio Regional de Transporte de Madrid"

}
```

- Finally, we have added the Listener in the Thread Group to observe the results. We have clicked on View Results Tree, Summary Report, Graph Results and View Results in Table as shown in Figure 17.

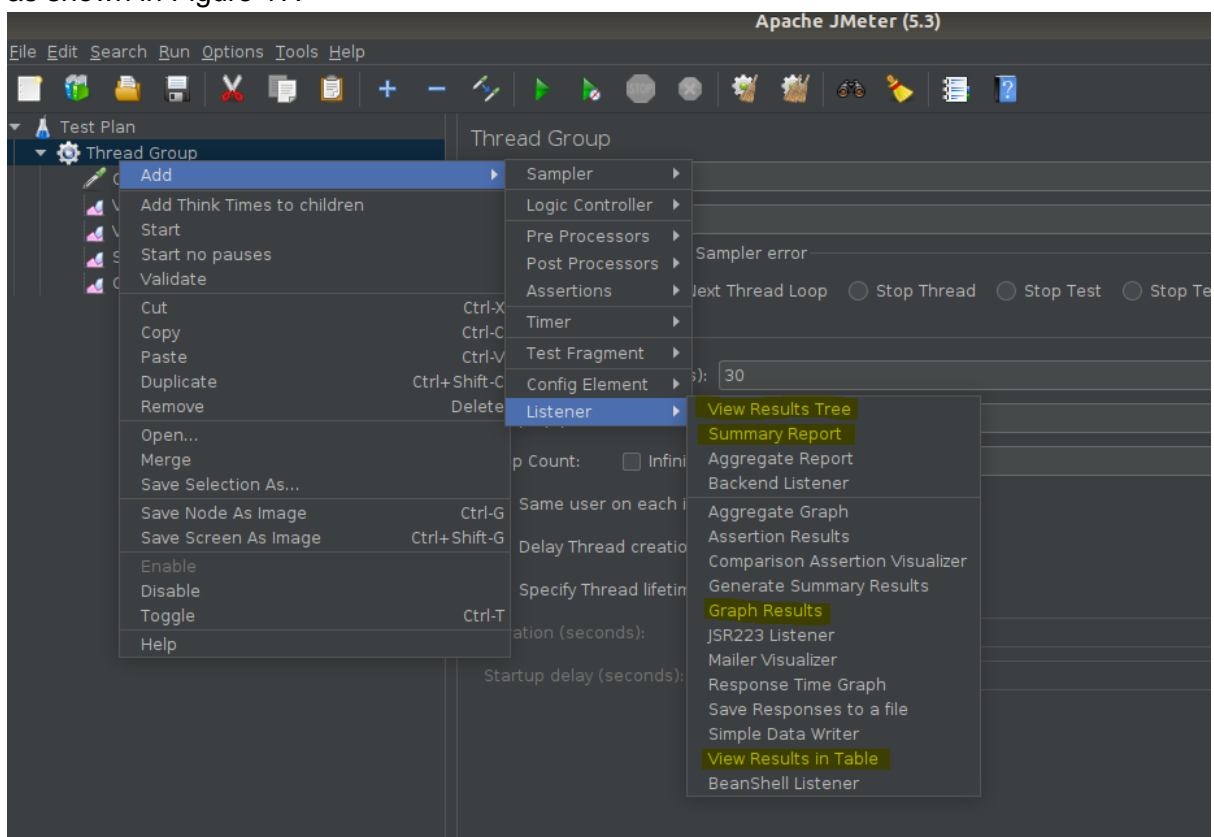


Figure 17 Adding listeners

The added listeners will be found below the Thread group as you can observe in the Figure 18.

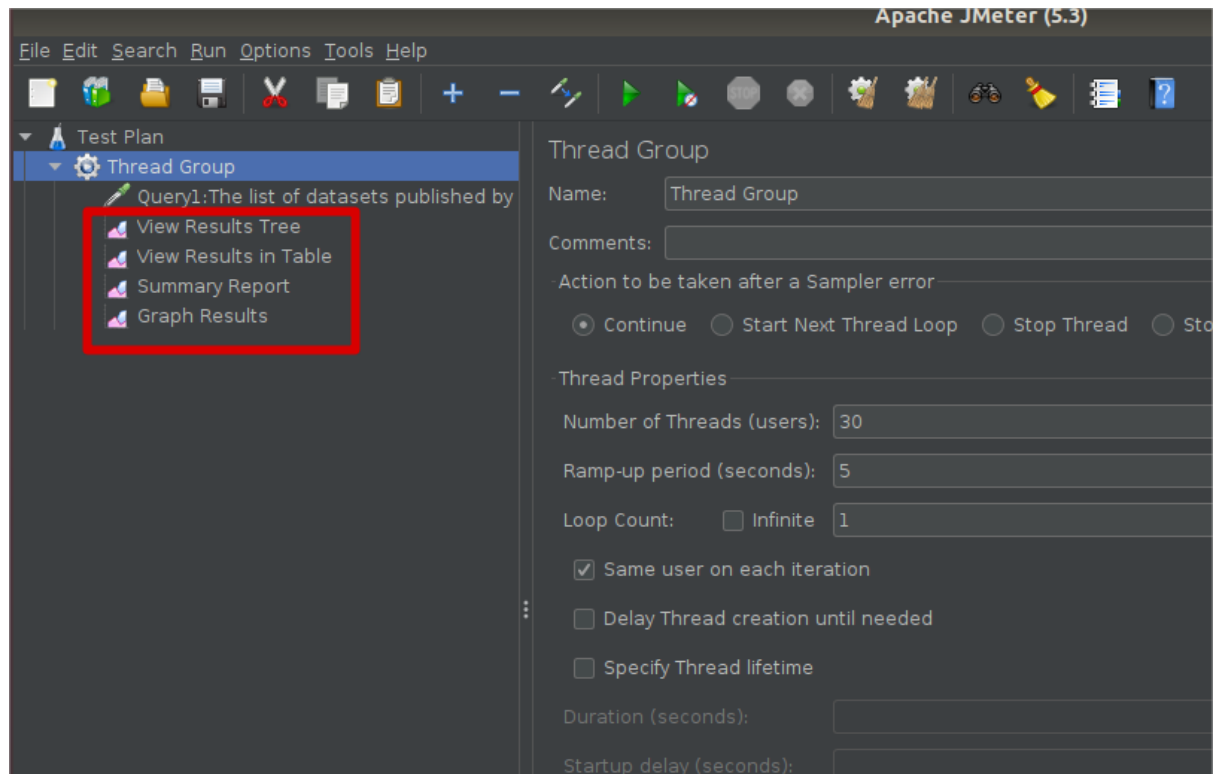


Figure 18 Added listeners

We have repeated the previous steps for the second query of our scenario and the final screen is shown in Figure 19. We can observe we have defined two queries related to Scenario S4 (Query1 and Query2) that will be executed by <http://localhost:5001/sparql>, through an API request with a given query (Query1 or Query2). For each query, these parameters must be specified. Then, our tests are already prepared to be executed by JMeter.

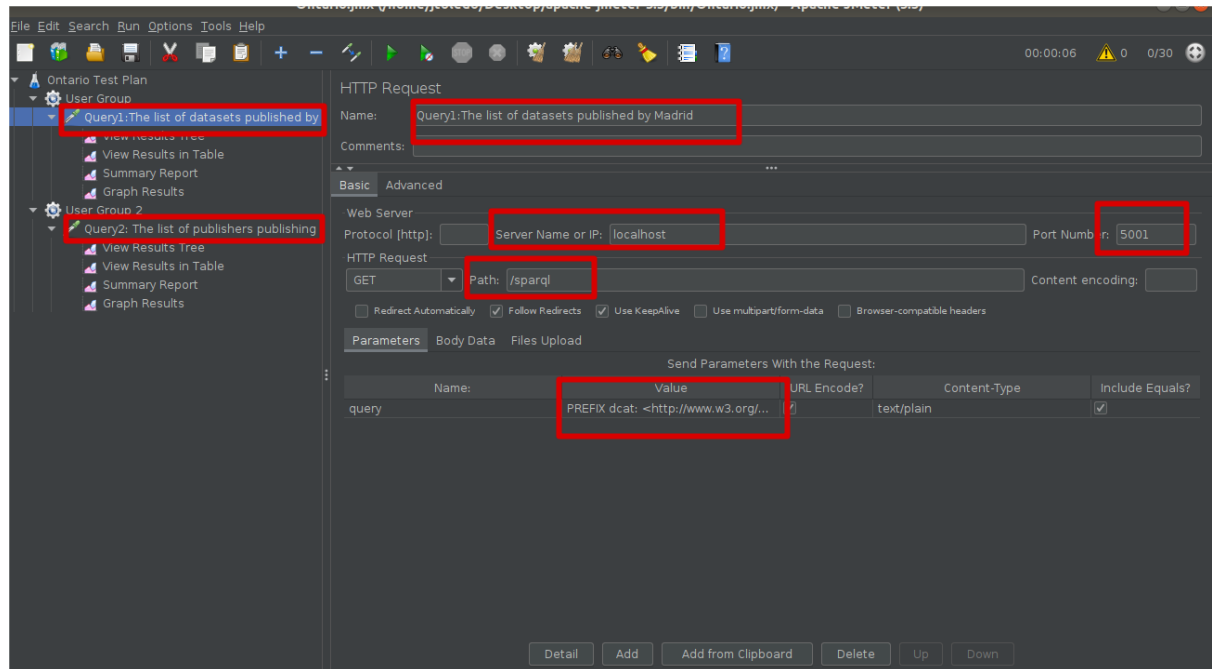


Figure 19 Created Test Plan

Validating performance test scenarios: Performance test execution principally involved multiple executions of ramp-up for the number of endpoints. In our case, the baseline to test the performance of the Distributed SPARQL endpoint will be the query execution on two endpoints (the minimum number of endpoints for a distributed engine).

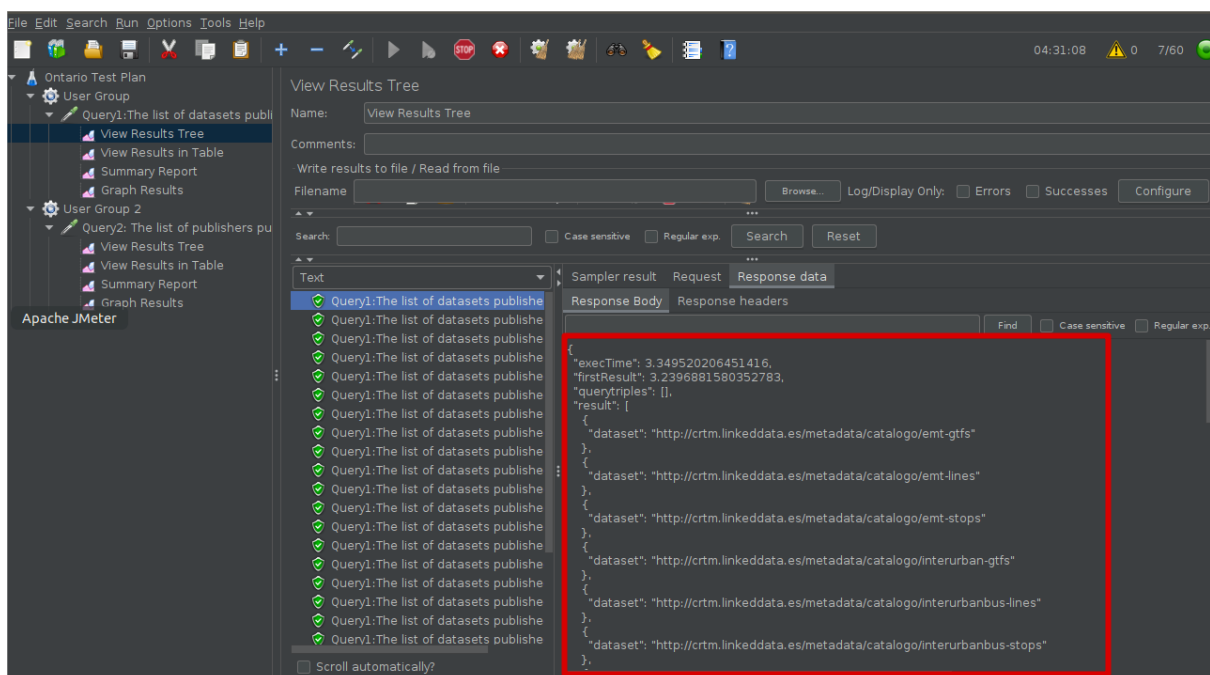
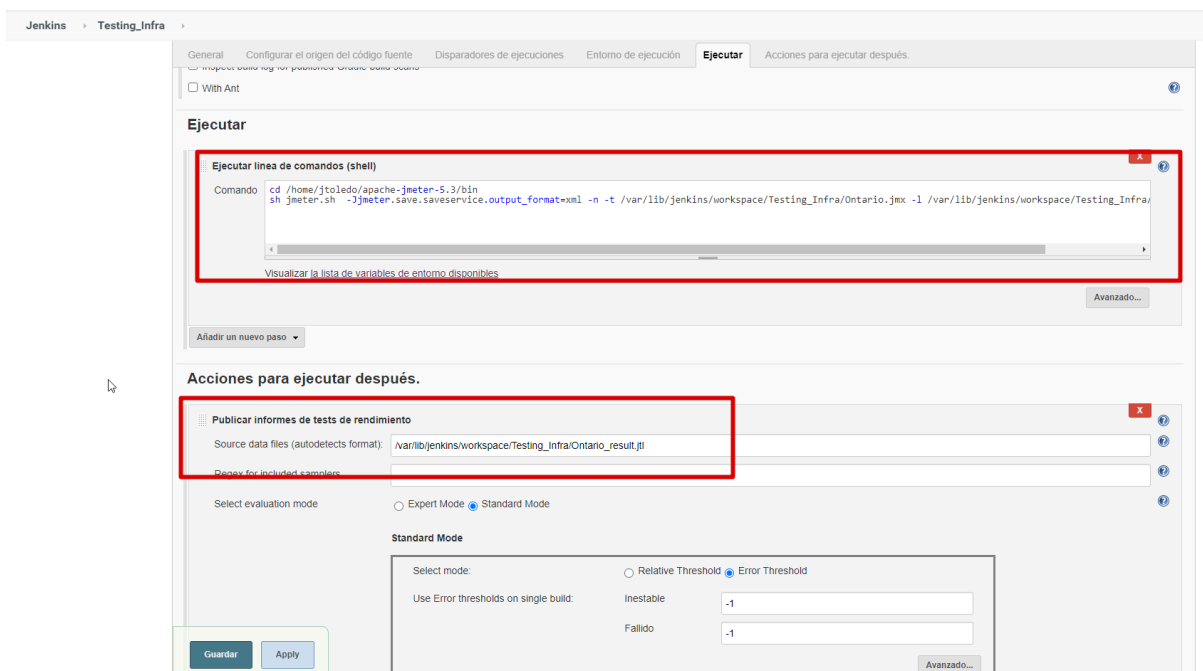


Figure 20 Validating results in JMeter

Automatically executing the performance tests: First, we must execute the following command in Jenkins:

```
sh jmeter.sh -Jjmeter.save.saveservice.output_format=xml -n -t  
/var/lib/jenkins/workspace/Testing_Infra/Ontario.jmx -l  
/var/lib/jenkins/workspace/Testing_Infra/Ontario_result.jtl
```

The script `jmeter.sh` receives the output format in xml, the plan generated by Jmeter (Ontario.jmx) and the output file (Ontario_result.jtl). This command must be specified in Jenkins in the first box of Figure 21. The output file is specified in the second box of Figure 21.



Jenkins - Testing_Infra

General Configurar el origen del código fuente Disparadores de ejecuciones Entorno de ejecución **Ejecutar** Acciones para ejecutar después

☐ With Ant

Ejecutar

Ejecutar línea de comandos (shell)

Comando

```
cd /home/jtoledo/apache-jmeter-5.3/bin  
sh jmeter.sh -Jjmeter.save.saveservice.output_format=xml -n -t /var/lib/jenkins/workspace/Testing_Infra/Ontario.jmx -l /var/lib/jenkins/workspace/Testing_Infra/Ontario_result.jtl
```

Visualizar la lista de variables de entorno disponibles

Avanzado...

Añadir un nuevo paso

Acciones para ejecutar después.

Publicar informes de tests de rendimiento

Source data files (autodetects format): /var/lib/jenkins/workspace/Testing_Infra/Ontario_result.jtl

Deny for included samples

Select evaluation mode

☐ Expert Mode ☒ Standard Mode

Standard Mode

Select mode:

☐ Relative Threshold ☒ Error Threshold

Use Error thresholds on single build:

Inestable -1

Fallido -1

Guardar Apply

Avanzado...

Figure 21 Configuring a test in Jenkins

Finally, once the test plan generated by JMeter is executed by Jenkins, the results are shown by Jenkins as shown in Figure 22.

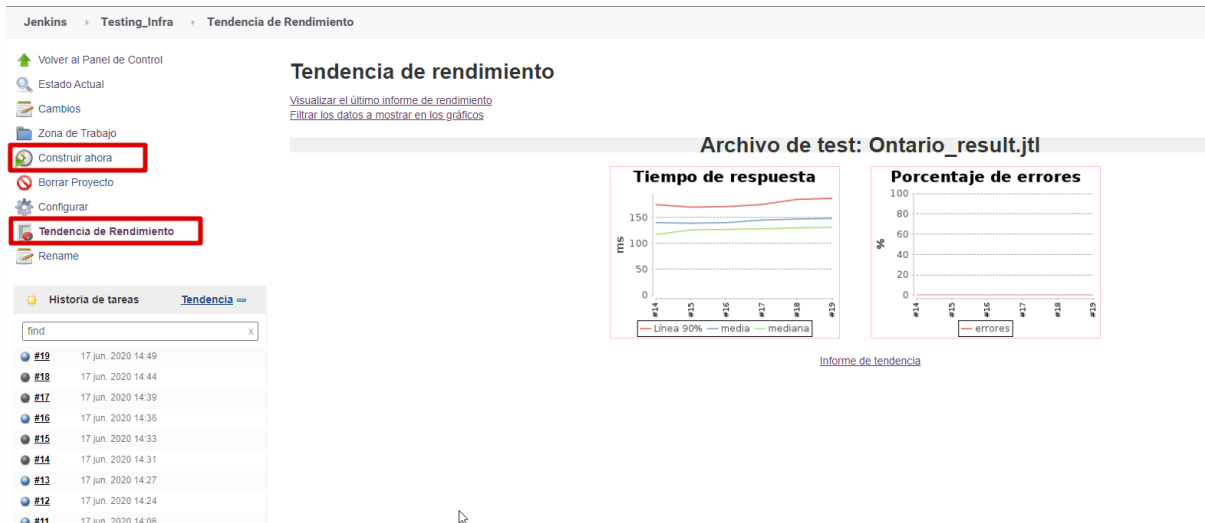


Figure 22 Executing a test plan in Jenkins

Collecting data: Both JMeter and Jenkins allow the data to be collected. The results obtained by the execution of the two queries by Ontario are shown by Jenkins and JMeter in Figure 23 and Figure 24.

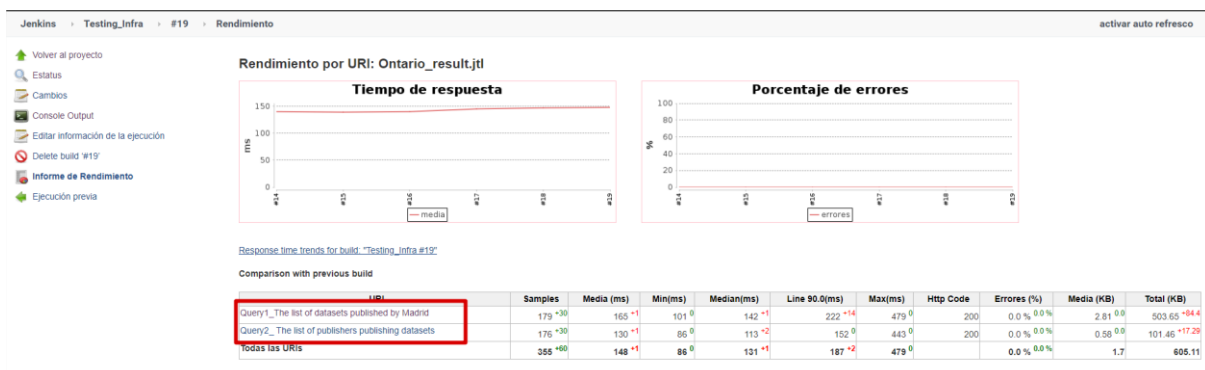


Figure 23 Collecting data results in Jenkins

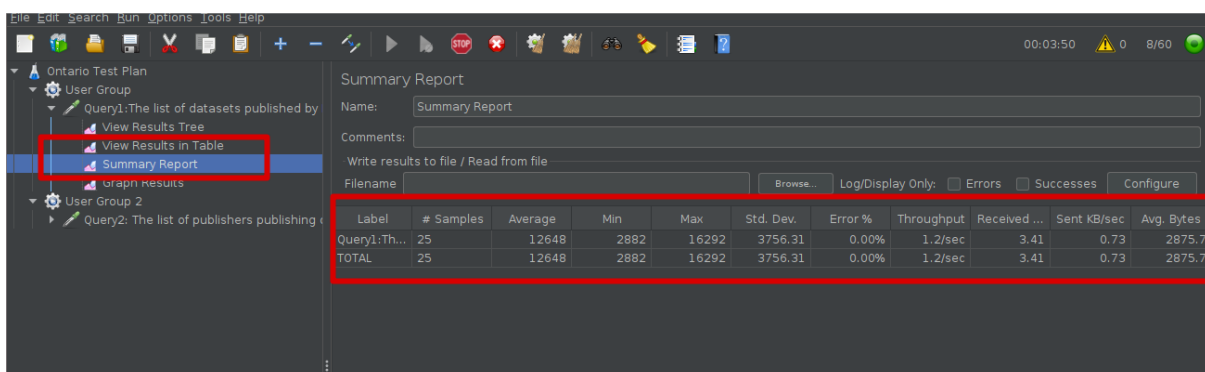


Figure 24 Collecting data results in JMeter

Conducting the final analysis and reporting: The reported results will be analyzed. Jenkins provides more detailed results. In our case, we can observe the average execution time obtained by each query and the total execution time obtained. Also, median, min, max, etc. are reported by Jenkins.

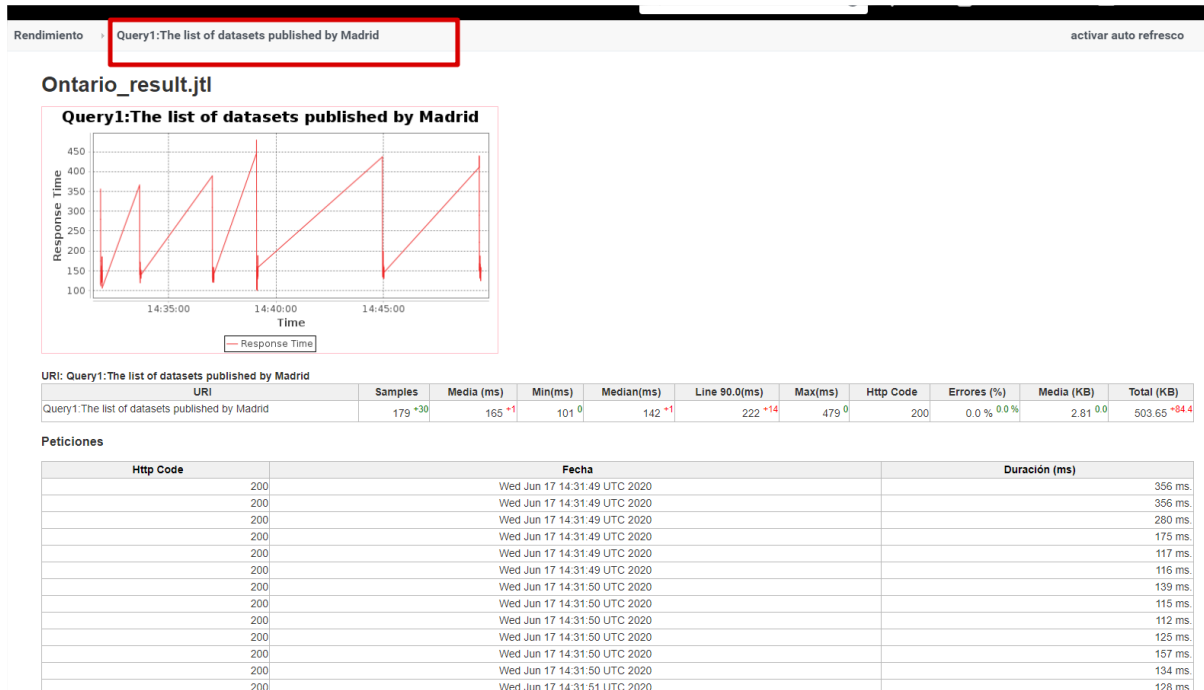


Figure 25 Final analysis and reporting

6. PERFORMANCE AND SCALABILITY REQUIREMENTS

This section complements performance and scalability requirements defined in D3.2, it updates requirements description based on the results of the validation of the pilot implementation (C-REL) and provides requirements for additional components developed within the project.

SPRINT aim is to provide a consistent set of tools and methodologies to support the establishment of the IP4 ecosystem. As SPRINT didn't develop any specific Resolver, we decided not to add requirements for such components. Resolvers also cover very specific functional requirements which are related to how the high level features of the IP4 ecosystem are implemented.

Since tracking user stories, requirements and test cases in all the previous deliverables might be complex, Table 10 contains a traceability matrix to help the readers understand where such topics were described. To avoid repeating old requirements which were unchanged, in this section we report only past requirements which were updated, and new requirements which were added to cover all the components developed in the context of this project.

Component	Requirement ID	Description	Ref. the full description of the Requirement and KPIs
Asset Manager	PR-1	Performance requirements for Ad-hoc Asset creation: (exemplified scenario for Converter asset)	Requirement in Deliverable D3.2: Table 13 – Req 2
	PR-2	Performance requirements for Exploration API: DCAT-AP metadata retrieval	Requirement in Deliverable D3.4: Table 13
	SR-1	Scalability requirements for Direct Download of an Asset (exemplified scenario for Converter asset)	Requirement in Deliverable D3.2: Table 16
	SR-2	Scalability requirements for Exploration API: DCAT-AP metadata retrieval	Requirement in Deliverable D3.4: Table 14
Distributed SPARQL query	PR-3	Asset/Service Discovery Response Time	Requirement in Deliverable D3.2: Table 13 – Req 1 Modified in Deliverable D3.4 Table 11

	SR-3	Scalability of Query/search time	Requirement in Deliverable D3.2: Table 14 Modified in Deliverable D3.4: Table 12
Converter	PR-4	Response Time to convert the whole data set	Requirement in Deliverable D3.2: Table 15 – Req 1
	PR-5	Response Time to convert one message	Requirement in Deliverable D3.2: Table 15 – Req 2
	SR-4	Scalability requirements for Runtime environment deployment of an Asset (exemplified scenario for Converter asset)	Requirement in Deliverable D3.2: Table 17
	SR-5	Response Time to convert big datasets	Requirement in Deliverable D3.4: Table 18
	SR-6	Response Time to convert multiple concurrent messages	Requirement in Deliverable D3.4: Table 19
Mapping Tool	PR-6	Execution Time	Requirement in Deliverable D3.2: Table 15 – Req 3
	PR-7	Usability	Requirement in Deliverable D3.2: Table 15 – Req 4
Collaborative Ontology Management Tool	PR-8	Ontology Documentation/Validation Time	Requirement in Deliverable D3.4: Table 13
	SR-7	Scalable Ontology Documentation/Validation Time	Requirement in Deliverable D3.4: Table 14

Automatic Generation Of Ontologies From Non-Ontological Sources (From Xsd To Ontology)	PR-9	Non-ontological conversion time	Requirement in Deliverable D3.4: Table 15
	SR-8	scalable Non-ontological conversion time	Requirement in Deliverable D3.4: Table 16

Table 10 – Requirements Traceability Table

6.1 MODIFIED REQUIREMENTS

6.1.1 Distributed SPARQL Endpoint

Requirement PR-3	
Referenced User Story	US-2 (Distributed SPARQL Endpoint)
KPI	Asset/Service Discovery Response Time
Definition	Time required to perform a Distributed SPARQL query on two RDF endpoints that publish metadata.
Requirement	The activity of looking for assets in the Asset Manager in a distributed environment. A discovery task implies multiple retries of a simple distributed search operation, each time adding filters to the previous try.
Target Value	The execution time for a distributed SPARQL query on two endpoints managed by the Asset Manager will be low and it should therefore not exceed 10 seconds to avoid frustrating users during the discovery task.

Table 11 Performance requirement for Distributed SPARQL Endpoint

Requirement SR- 3	
Referenced User Story	US-2 (Distributed SPARQL Endpoint)
KPI	Distributed Query/search time
Definition	Time required to perform a SPARQL query on multiple RDF endpoints that publish metadata.
Requirement	As the discovery/search capabilities can be implemented by means of distributed SPARQL queries, the execution time for such queries must stay constant or at worst grow linearly as the number of endpoints grows.
Target Value	<p>The response time on a distributed SPARQL query over multiple endpoints containing metadata, comprises many factors that can affect it. Mainly, the response time will be affected by the number of endpoints, and the number of results returned per query over a specific number of endpoints.</p> <p>For a distributed SPARQL query, the response time will be longer than a centralized one since it must first select which endpoints are of interest for the query and then execute a query for each of the selected endpoints in order to subsequently integrate the results obtained by each query. Since the integration of distributed query is a more complex process, it is expected that the response time will worsen by at least an order of magnitude when the number of endpoints is large. For example, if the average time for a query is less than 10 seconds, then in the worst case, the average response time on a distributed query could be less than 20 seconds.</p>

Table 12 Scalability requirement for Distributed SPARQL Endpoint

6.2 NEW REQUIREMENTS

6.2.1 Asset Manager

Requirement PR-2	
Referenced User Story	US-1 (Join and search)
KPI	Response time for the “Explore” page
Definition	Search feature in the Asset Manager is exposed via the “Explore” page. Opening such page triggers an Exploration API to retrieve the basic DCAT-AP metadata of all the assets. This in turn triggers a SPARQL query on the RDF repository. Measuring response time related to this feature therefore allows checking multiple components in the backend of the Asset Manager.
Requirement	Loading time of the Explore page must not disrupt the user experience
Target Value	Loading time for the Explore page in the Asset Manager Publisher should be kept under 5 seconds

Table 13 Performance requirements for Exploration API: DCAT-AP metadata retrieval

Requirement SR- 2	
Referenced User Story	US-1 (Join and search)
KPI	Response time for the “Explore” page
Definition	<p>Search feature in the Asset Manager is exposed via the “Explore” page. Opening such page triggers an Exploration API to retrieve the basic DCAT-AP metadata of all the assets. This in turn triggers a SPARQL query on the RDF repository. Measuring response time related to this feature therefore allows checking multiple components in the backend of the Asset Manager.</p> <p>Scalability in this case is affected by both the number of concurrent users and by the number of the assets published in the RDF repository.</p>

Requirement	Loading time for the Explore page in the Asset Manager should scale with respect to the number of concurrent users and the number of assets.
Target Value	Loading time for the Explore page in the Asset Manager Publisher should be kept under 5 seconds with up to 1000 assets and 10 concurrent users.

Table 14 Scalability requirements for Exploration API: DCAT-AP metadata retrieval

6.2.2 Collaborative Ontology Management Tool

Requirement PR- 8	
Referenced User Story	US-9 (Collaborative Ontology Manager)
KPI	Ontology Documentation/Validation Time
Definition	The time required to produce diagrams, complete documentation and validation based on common pitfalls for one ontology.
Requirement	The documentation, generation and validation imply the documentation with several proposals for diagram representation, versioning and evaluation of the ontology in the user's repository. The activity of documentation, generation and validation for one ontology varying number of classes, number of relationships, number of data properties, and number of object properties.
Target Value	The execution time of the whole process of both the evaluation and the generation of documentation have an average time of 5 minutes.

Table 15 Performance requirement for Collaborative Ontology Manager

Requirement SR- 5	
Referenced User Story	US-9 (Collaborative Ontology Manager)
KPI	Scalable Ontology Documentation/Validation Time
Definition	Time required to produce diagrams, a complete documentation and validation based on common pitfalls for multiple ontologies.
Requirement	As the ontology documentation and validation capabilities can be supported by several tools (Oops, Widoco, Ar2Tool, etc), the execution time for such tasks must stay constant or at worst grow linearly as the number of ontologies, classes and properties grows.
Target Value	The execution time of the entire process, both evaluation and documentation generation, has an average time of 5 minutes per ontology. therefore, as the number of ontologies increases, linear growth is expected over time.

Table 16 Scalability requirement for Collaborative Ontology Manager

6.2.3 Automatic Generation of Ontologies from Non-ontological Sources (from XSD to Ontology)

Requirement PR- 9	
Referenced User Story	US-8 (Generation of ontologies from non-ontological sources)
KPI	Non-ontological conversion time
Definition	Time required to transform an XML schema into an ontology.
Requirement	It is required a well-formed and valid XML document to transform an XML schema to an ontology. The target execution time must be such as to avoid degrading the overall transformation time. Therefore, its order of magnitude must be in terms of a few seconds.

Target Value	The execution time for transforming a non-ontological source will be low and it should therefore not exceed 5 seconds to avoid frustrating users during the transformation task.
---------------------	--

Table 17 Performance requirement for Generation of ontologies from non-ontological sources

Requirement SR-6	
Referenced User Story	US-8 (Generation of ontologies from non-ontological sources)
KPI	Scalable non-ontological conversion time
Definition	Time required to transform an XML schema into an ontology for multiple number of elements and types.
Requirement	The transformation time of an XML schema will be longer as the number of elements and types increases and a lineal growth is expected at the worst case.
Target Value	The execution time for transforming multiple non-ontological sources must stay constant, or at worst grow linearly as the of elements and types and it should therefore not exceed 1 min to avoid frustrating users during the transformation task.

Table 18 Scalability requirement for Generation of ontologies from non-ontological sources

6.2.4 Converter

Requirement SR-5	
Referenced User Story	US-3 (Batch Data Conversion)
KPI	Response Time
Definition	Time required to convert big datasets.
Requirement	The same batch conversion procedure defined by a Converter between two formats should scale considering growing size input datasets.
Target Value	The execution time for converting a dataset must grow linearly as the size of the input dataset grows. Batch conversion do not have strict time requirements and can take up to a few days to be considered acceptable.

Table 19 – Scalability requirement SR-4 for the Converter

Requirement SR-6	
Referenced User Story	US-4 (Runtime Data/Message Conversion)
KPI	Response Time
Definition	Time required to convert multiple concurrent messages.
Requirement	The same runtime data/message conversion procedure defined by a Converter between two formats should scale considering a growing size of concurrent requests.
Target Value	The execution time for converting a message must stay constant and at worst grow linearly as the number of concurrent requests grows. Message conversion has strict time requirements and can not take more than 10 seconds.

Table 20 – Scalability requirement SR-5 for the Converter

7. PERFORMANCE AND SCALABILITY TEST CASES

Based on the design of IF Testing Infrastructure, the first step according Section 5 is to summarize the test cases at a high level. In this section we will therefore relate the requirements/criteria to be tested along with their relevant information such as user story and the process that is most frequently executed for the component to be tested. This will provide a solid base for the technical development of the test cases which will be implemented in the context of WP5, and validated in D5.6.

7.1 COLLABORATIVE ONTOLOGY MANAGEMENT

For one ontology, we will generate its documentation and evaluate its quality to measure how affect the ontology structure on the semi-automatic documentation generation//evaluation process.

TC1 - Performance Testing for Collaborative ontology manager	
Process	Documentation, generation and evaluation of an ontology during the ontology development process
Related User Story	US-9 Collaborative Ontology Manager
Performance Requirements	PR-8. Ontology Documentation/Validation Time
Performance Criteria	Average execution time of the whole process of both the evaluation and the generation of documentation for one ontology

We will generate documentation and evaluate quality for multiples ontologies to measure how the input of multiple ontologies affects the documentation generation//evaluation process.

TC2 - Scalability Testing for Collaborative ontology manager	
Process	Documentation, generation and evaluation of several ontologies during the ontology development process
Related User Story	US-9 Collaborative Ontology Manager
Scalability Requirements	SR-7.Scalable Ontology Documentation/Validation Time
Scalability Criteria	Average execution time of the whole process of both the evaluation and the generation of documentation for several ontologies

7.2 AUTOMATIC GENERATION OF ONTOLOGIES FROM NON-ONTOLOGICAL SOURCES

We will transform from one XML schema with different structures (number elements and types) to one ontology to measure how affect the XML schema structure on the automatic ontology generation process.

TC3 - Performance Testing for Automatic generation of ontologies from non-ontological sources	
Process	Transformation of one XML schema to an ontology
Related User Story	US-8 Generation of Ontologies from Non-ontological Sources
Performance Requirements	PR-9.Non-ontological conversion time
Performance Criteria	Average execution time to transform an XML schema to an ontology

We will transform to one ontology from one XML schema scaling its number elements and types to measure how affect growth in the number of elements/types of an XML schema on the automatic ontology generation process.

TC4 - Scalability Testing for Automatic generation of ontologies from non-ontological sources	
Process	Transformation of an XML schema into an ontology for multiple number of elements and types
Related User Story	US-8 Generation of Ontologies from Non-ontological Sources
Scalability Requirements	SR-8.Scalable Non-ontological conversion time
Scalability Criteria	Average execution time to transform an XML schema to an ontology when multiple number of elements and types are considered

7.3 MAPPING TOOL

TC5 - Performance Testing Mapping Tool	
Process	To run the Mapping tool for various pairs of the Source and Target Standard in multiple formats and lengths (number of terms).
Related User Story	US-7 Mapping process for the conversion user story
Performance/Scalability Requirements	PR-6. Execution Time
Performance/Scalability Criteria	Average execution time to perform the mapping process

TC6 - Performance Testing Mapping Tool	
Process	To run Mapping Tool and evaluate the number of required steps to fulfill whole process.
Related User Story	US-7 Mapping process for the conversion user story
Performance/Scalability Requirements	PR-7. Usability
Performance/Scalability Criteria	Number of steps

7.4 ASSET MANAGER

TC7 - Performance Testing for Asset manager

Process	Loading of the Explore page in the Asset Manager Publisher
Related User Story	US-1 (Join and search)
Performance Requirements	PR-1A. Performance requirements for Exploration API: DCAT-AP metadata retrieval
Performance Criteria	Average page loading time.

TC8 - Scalability Testing for Asset manager

Process	Loading of the Explore page in the Asset Manager Publisher
Related User Story	US-1 (Join and search)
Scalability Requirements	SR-1A. Scalability requirements for Exploration API: DCAT-AP metadata retrieval
Scalability Criteria	Average page loading time with an increasingly high number of concurrent requests and increasingly high number of assets.

7.5 CONVERTER

To test the Converter, we define test cases considering the two main scenarios: *batch data conversion* and *runtime data/message conversion*. Performances highly depends on the different format of input data (CSV, JSON, XML, ...), the complexity of the defined mappings (e.g., number of joins between different data sources) and the size/density of the intermediate knowledge graph. For this reason, to compare test results between different approaches and software artifacts implementing the conversion, it is important to consider the same pipeline defined between the same formats, with the same mappings and generating the same intermediate knowledge graph.

TC9 - Performance Testing for Batch Data Conversion	
Process	A pipeline for batch conversion from standard A to standard B is executed providing an input dataset encoded using standard A. The response time and resource usage (mem, CPU) are measured during the conversion.
Related User Story	US-3 Batch Data Conversion
Performance Requirements	PR-4 Response Time to convert the whole data set
Performance Criteria	Average execution time to convert a data set.

TC10 - Performance Testing for Runtime Data/Message Conversion	
Process	A pipeline for message conversion from standard A to standard B is executed to reply a request with payload encoded using standard A. The response time and resource usage (mem, CPU) are measured during the conversion.
Related User Story	US-4 Runtime Data/Message Conversion
Performance Requirements	PR-5 Response Time to convert one message
Performance Criteria	Average response time to perform the conversion and reply to a request.

In order to assess the scalability of the Converter it is important to compare results obtained considering two different scalability criteria: the size of the dataset for *batch data* conversion, and the number of concurrent requests for *runtime data/message* conversion.

TC11 - Scalability Testing for Batch Data Conversion	
Process	A pipeline for batch conversion from standard A to standard B is executed providing input datasets with growing size encoded using standard A. The response time and resource usage (mem, CPU) are measured during the conversion.
Related User Story	US-3 Batch Data Conversion
Scalability Requirements	SR-5 Response Time to convert big datasets
Scalability Criteria	Average execution time to convert a data set considering growing sizes of the input.

TC12 - Scalability Testing for Runtime Data/Message Conversion	
Process	A pipeline for message conversion from standard A to standard B is executed to reply multiple concurrent requests with payload encoded using standard A. The response time for each request and the resource usage (mem, CPU) are measured during the conversion.
Related User Story	US-4 Runtime Data/Message Conversion
Performance Requirements	SR-6 Response Time to convert multiple concurrent messages
Scalability Criteria	Average response time to perform the conversion and reply to a request considering a growing number of concurrent requests.

The TC12 test case should be performed considering one replica of the Converter. However, multiple replicas of the Converter can improve scalability using load-balancing to distribute the requests. Therefore, to test the scalability of the deployment solution we define an additional test case for the Converter.

TC13 - Scalability Testing for Runtime Environment Deployment	
Process	A Converter is deployed on a runtime environment measuring the time to complete a deployment and the time required to scale the deployment.
Related User Story	US-5 Fast Adaptation to Peaks
Performance Requirements	SR-4 Scalability requirements for Runtime environment deployment of an Asset (exemplified scenario for Converter asset)
Scalability Criteria	Average time to complete a deployment considering a different number of replicas to be deployed. Average time to add X replicas to the deployment considering different values for X.

7.6 DISTRIBUTED SPARQL ENDPOINT

We will execute a set of SPARQL queries without preference criteria which will be compared against queries including preference criteria to measure how much it costs to evaluate preference criteria on SPARQL because preference evaluation can be costly. In this context, we will analyze the performance the queries on two endpoints.

TC14 - Performance Testing for Distributed SPARQL endpoint	
Process	Execution of federated queries with and without preferences over two endpoints
Related User Story	US-2 for Distributed service/asset discovery
Performance Requirements	PR-3. Asset/Service Discovery Response Time
Scalability Criteria	Total execution query time: Average execution time of queries

We will execute a set of SPARQL queries with/without preference criteria in a scenario with a higher number of endpoints. The aim is to analyze how growth in the number of endpoints impacts performance of the Distributed SPARQL endpoint.

TC15 - Scalability Testing for Distributed SPARQL endpoint	
Process	Execution of federated queries with and without preferences over multiple endpoints
Related User Story	US-2 for Distributed service/asset discovery
Performance Requirements	SR-. Query/search time
Scalability Criteria	Total execution query time: Average execution time of queries

8. CONCLUSIONS

This deliverable described an architecture which can be used by Shift2Rail IP4 project to lay the foundations of a digital ecosystem for passenger-centric services. The test cases related to performance and scalability will be evaluated and documented in D5.6, which will contain also a functional validation of the F-Rel version of the SPRINT Interoperability Framework.

9. REFERENCES

- [1] [C. Richardson, *Microservices Patterns*, Manning Publications, 2018.](#)
- [2] [A. Dimou, M. V. Sande, P. Colpaert, E. Mannens and R. V. d. Walle, "Extending R2RML to a Source-independent Mapping Language for RDF," in *International Semantic Web Conference \(Posters & Demos\)*, 2013.](#)
- [3] [SPRINT, "D3.2 - PERFORMANCE AND SCALABILITY REQUIREMENTS FOR THE IF," 2019. \[Online\]. Available: <http://sprint-transport.eu/Page.aspx?CAT=DELIVERABLES&IdPage=1e2645be-e780-4d99-8117-bae57b67b453>.](#)
- [4] [S. Goedertier, *DCAT application profile for data portals in Europe*, 2013.](#)
- [5] [M. Dekkers, "Asset description metadata schema \(adms\)," *W3C Working Group*, 2013.](#)
- [6] ["Docker," \[Online\]. Available: <https://www.docker.com/>.](#)
- [7] ["Kubernetes," \[Online\]. Available: <https://kubernetes.io>.](#)
- [8] [A. Meroño-Peñuela and R. Hoekstra, "grlc makes GitHub taste like linked data APIs," in *European Semantic Web Conference*, 2016.](#)
- [9] [E. Daga, L. Panziera and C. Pedrinaci, "Basil: A cloud platform for sharing and reusing SPARQL queries as Web APIs," in *CEUR Workshop Proceedings*, 2015.](#)
- [10] [M. Saleem, G. Szárnyas, F. Conrads, S. A. C. Bukhari, Q. Mehmood and A. C. Ngonga Ngomo, "How representative is a sparql benchmark? an analysis of rdf triplestore benchmarks," in *The World Wide Web Conference*, San Francisco, USA, 2019.](#)
- [11] [D. Chaves-Fraga, F. Priyatna, A. Cimmino, J. Toledo, E. Ruckhaus and O. Corcho, "GTFS-Madrid-Bench: A Benchmark for Virtual Knowledge Graph Access in the Transport Domain".](#)
- [12] [C. Bizer and A. Schultz, "The Berlin SPARQL benchmark," *Int. J. Semantic Web Inf. Syst*, vol. 5, pp. 1-24, 2009.](#)
- [13] [D. Lanti, M. Rezk, M. Slusnys, G. Xiao and D. Calvanese, "The NPD benchmark for OBDA systems," in *CEUR Workshop Proceedings*, 2014.](#)
- [14] [M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte and T. Tran, "FedBench: A Benchmark Suite for Federated Semantic Data Query Processing," in *International Semantic Web Conference*, 2011.](#)
- [15] [G. Montoya, M. Vidal, O. Corcho, E. Ruckhaus and C. Buil-Aranda, "Benchmarking federated SPARQL query engines: Are existing testbeds enough?," in *International Semantic Web Conference*, 2012.](#)

- [16] [D. Lanti, G. Xiao and D. Calvanese, "VIG: Data scaling for OBDA benchmarks," *Semantic Web*, vol. 10, no. 2, pp. 413-433, 2019.](#)
- [17] [S. Das, S. Sundara and R. Cygania, "R2RML: RDB to RDF Mapping Language. W3C Recommendation," September 2012. \[Online\]. Available: <https://www.w3.org/TR/r2rml/>.](#)
- [18] [F. Michel, L. Djimenou, C. Faron-Zucker and J. Montagnat, "xR2RML: Relational and Non-Relational Databases to RDF Mapping Language," 2014.](#)
- [19] [D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro and G. Xiao, "Ontop: Answering SPARQL queries over relational databases," *Semantic Web*, vol. 8, no. 3, pp. 471-487, 2017.](#)
- [20] [M. Schmidt, T. Hornung, M. Meier, C. Pinkel and G. Lausen, "SP2Bench: A SPARQL Performance Benchmark," in *International Conference on Data Engineering*, 2009.](#)
- [21] [Y. Guo, Z. Pan and J. Heflin, "LUBM: A benchmark for OWL knowledge base systems," *J. Web Semant.*, vol. 3, pp. 158-182, 2005.](#)
- [22] [M. Morsey, J. Lehmann, S. Auer and A. Ngomo, "DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data," in *International Semantic Web Conference*, 2011.](#)
- [23] ["GTFS Static Overview," \[Online\]. Available: <https://developers.google.com/transit/gtfs/>.](#)
- [24] [A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini and R. Rosati, "Linking data to ontologies," *Journal on data semantics X*, pp. 133-173, 2008.](#)
- [25] ["General Transit Feed Specification," \[Online\]. Available: \[https://www.transitwiki.org/TransitWiki/index.php?title=General Transit Feed Specification\]\(https://www.transitwiki.org/TransitWiki/index.php?title=General+Transit+Feed+Specification\).](#)
- [26] [S. Harris and A. Seaborne, "SPARQL 1.1 Query Language W3C Recommendation," 21 March 2013. \[Online\]. Available: <https://www.w3.org/TR/2013/REC-sparql11-query-20130321/>.](#)