

SEMANTICS FOR PERFORMANT AND SCALABLE INTEROPERABILITY OF MULTIMODAL TRANSPORT

D5.6 – Final report on the results of the validation of pilot implementation

Due date of deliverable: 31/10/2020

Actual submission date: 21/01/2021

Leader/Responsible of this Deliverable: CEFRIEL

Reviewed: Y

Document status		
Revision	Date	Description
0.1	01/08/2020	Draft, working version
0.2	11/12/2020	First complete draft of the Functional evaluation section
0.3	21/12/2020	First complete draft of the Performance and Scalability section
0.4	22/12/2020	Added Introduction and Conclusions
1.0	18/01/2021	Final version for TMC approval
2.0	22/01/2020	Final version after TMC approval and quality check

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: 27 months

EXECUTIVE SUMMARY

The aim of the deliverable D5.6 is to validate the functional and nonfunctional capabilities of the implemented solution within the final release (F-REL).

Section 2 reports the functional validation of the use case scenarios defined in D5.4, describing how each scenario has been implemented using the tools developed in SPRINT.

One of the aims of the SPRINT project is to study performances and scalability of semantic-based interoperability solutions. Such topics are described in Section 3, which reports the results and the evaluation of the performance and scalability tests defined in D3.4.

Section 3.7.6 then provides a business and market validation, evaluating the improvements of the F-REL prototype using the same indicators for the evaluation based on the recommendations for the IF development and deployment produced in GOF4R D5.2 “Toolkit of recommendations and KPI Scoreboard”.

Finally, Annex A reports the whole set of requirements which have been considered while implementing the final version of the SPRINT tools by providing a traceability matrix, and Annex B reports additional data collected in the performance and scalability evaluation discussed in Section 3.

ABBREVIATIONS AND ACRONYMS

Abbreviation	Description
C-REL	Core Release
F-REL	Final Release
EU	European Union
GA	Grant Agreement
H2020	Horizon 2020 framework programme

TABLE OF CONTENTS

Executive Summary	2
Abbreviations and Acronyms	3
Table of Contents.....	4
List of Figures	7
List of Tables	11
1. Introduction	14
2. Functional validation	15
2.1 Scenario S1: Joining the IF Use case (User Registration)	16
2.1.1 Tools configuration	16
2.1.2 Validation.....	16
2.2 Scenario S2: Joining the IF Use case (Provider Registration)	19
2.2.1 Tools configuration	19
2.2.2 Validation.....	20
2.3 Scenario S4: Distributed service/asset discovery	22
2.3.1 Tools configuration	23
2.3.2 Validation.....	23
2.4 Scenario S7: Automated Mapping Process for the Conversion use case	25
2.4.1 Tools configuration	26
2.4.2 Validation.....	28
2.5 Scenario S8: Automatic converter building Use case	36
2.5.1 Tools configuration	36
2.5.2 Validation.....	38
2.6 Scenario S9: Fast Adaptation to Peaks Use case	41
2.6.1 Tools configuration	41
2.6.2 Validation.....	42
2.7 Scenario S10: Special Purpose Asset Discovery Package: Resolver	45
2.7.1 Tools configuration	46
2.7.2 Validation.....	51
2.8 Scenario 11: (Collaborative) Ontology Manager	53
2.8.1 Tools configuration	53
2.8.2 Validation.....	55
2.9 Scenario 12: Ontology Creation using non-ontological sources.....	57
2.9.1 Tools configuration	57
2.9.2 Validation.....	58
2.10 Scenario S13: Asset Manager as National Access Points aggregator	60

2.10.1	Tools configuration	60
2.10.2	Validation.....	60
2.11	Scenario S14: Asset Manager as a tool for contributing to a National Access Point	63
3.	Performance and scalability evaluation	64
3.1	Collaborative ontology management	64
3.1.1	TC1 - Performance Testing for Collaborative ontology manager	64
3.1.2	TC2 - Scalability Testing for Collaborative ontology manager	68
3.2	Automatic generation of ontologies from non-ontological sources	72
3.2.1	TC3 - Performance Testing for Automatic generation of ontologies from non-ontological sources	72
3.2.2	TC4 - Scalability Testing for Automatic generation of ontologies from non-ontological sources	80
3.3	Mapping Tool	83
3.3.1	TC5 - Performance Testing Mapping Tool	83
3.3.2	TC6 - Usability Testing Mapping Tool	84
3.3.3	TC5 and TC6 Testing Activities.....	84
3.4	Asset Manager	88
3.4.1	TC7 - Performance Testing for Asset manager	88
3.4.2	TC8 - Scalability Testing for Asset manager	93
3.5	Converter	97
3.5.1	TC9 - Performance Testing for Batch Data Conversion	98
3.5.2	TC10 - Performance Testing for Runtime Data/Message Conversion	106
3.5.3	TC11 - Scalability Testing for Batch Data Conversion.....	113
3.5.4	TC12 - Scalability Testing for Runtime Data/Message Conversion	120
3.5.5	TC13 - Scalability Testing for Runtime Environment Deployment	123
3.6	Distributed SPARQL Endpoint.....	127
3.6.1	TC14 - Performance Testing for Distributed SPARQL endpoint.....	127
3.6.2	TC15 - Scalability Testing for Distributed SPARQL endpoint	131
3.7	Overall Performance and Scalability Evaluation	135
3.7.1	Collaborative Ontology Manager.....	135
3.7.2	Automatic generation of ontologies from non-ontological sources.....	135
3.7.3	Mapping Tool.....	136
3.7.4	Asset Manager	137
3.7.5	Converter.....	138
3.7.6	Distributed SPARQL endpoint.....	138
4.	Business and market validation.....	140
5.	Conclusions	143

6. Annex A: Functional Requirements traceability matrix.....	144
6.1 Automatic Generation of Ontologies from Non-ontological Sources	144
6.2 Collaborative Ontology Manager	145
6.3 Mapping suggerter.....	146
6.4 Distributed SPARQL endpoint.....	147
6.5 Asset Manager	148
6.6 User Manager	151
6.7 Converter	152
7. Annex B: Additional Data Performance and Scalability Evaluation	154
7.1 TC8 – Scalability Testing for Asset Manager	154
7.2 TC10 - Performance Testing for Runtime Data/Message Conversion	158
7.2.1 Annotation approach.....	158
7.2.2 Declarative approach.....	168
7.3 TC11 - Scalability Testing For Batch Data Conversion.....	172
7.4 TC12 - Scalability Testing For Runtime Data/Message Data Conversion	174
7.4.1 Annotation approach.....	174
7.4.2 Declarative approach.....	177
8. Bibliography	183

LIST OF FIGURES

Figure 1 Asset Manager login screen with Keycloak external authentication enabled	17
Figure 2 Shift2Rail IP4 ecosystem Single Sign-On login page	18
Figure 3 Escalation to Provider – BPMN process.....	20
Figure 4 Escalation to Provider – Triggering the request.....	20
Figure 5 Request to become a Provider – Notification to the Administrator	21
Figure 6 Request to become a Provider – Administrator’s tasks list	21
Figure 7 Request to become a Provider – Decision task for the Administrator	21
Figure 8 SPARQL interfaces with metadada in RDF from Belgium and Spain	23
Figure 9 Query and results obtained over two endpoints.....	24
Figure 10 Results obtained over twelve endpoints	24
Figure 11 Home page of the mapping and annotation application.....	27
Figure 12 Starting Wizard	27
Figure 13 Start Mapping Generation	27
Figure 14 Mapping Suggestions.....	28
Figure 15 Output generation	28
Figure 16 FSM to Transmodel mappings provided by MappingTool.....	30
Figure 17 FSM to Transmodel Mapping Statitics	31
Figure 18 FSM to IT2Rail mappings provided by MappingTool	32
Figure 19 FSM to IT2Rail Mapping Statistics	33
Figure 20 Partial snapshot of an annotated Java file. The added annotations are marked by red rectangles.	34
Figure 21 Dependencies between a Converter and other assets	37
Figure 22 Default lifecycle management process with dependency tracking	38
Figure 23 Kubernetes deployment descriptor for a Chimera converter created by the Asset Manager	39
Figure 24 Dependency update notification email sent from the Asset Manager	40
Figure 25 The Converter Resolver enables a “universal converter” to access needed resources from the Asset Manager at runtime.	47
Figure 26 Converter dependencies resources API exposed by the Asset Manager via the Exploration API	48
Figure 27 SPARQL query to retrieve all the resources required to run a specific conversion	49
Figure 28 JSON-LD Frame applied to the dependency tracking SPARQL query	50
Figure 29 Converter dependencies resources API result example	50
Figure 30 Example message obtained from the HaCon VBB endpoint.....	51
Figure 31 Example message in TRIAS obtained through the defined conversion	52
Figure 32 Logs of the Converter Resolver and the “universal” Converter for a VBB to TRIAS request	52

Figure 33 Repository structure	54
Figure 34 Jenkinsfile environment setup	54
Figure 35 Repository configuration in Jenkins.....	55
Figure 36 Documentation generated by widoco	55
Figure 37 Evaluation generated by Oops!	56
Figure 38 Documentation generated by vocablite	56
Figure 39 Usage XSD2OWL	58
Figure 40 Result netex_facility_support.xsd to OWL file.....	59
Figure 41 Differences between an automatically created and an ontology created manually	59
Figure 42 Asset Manager Publisher showing results from multiple NAPs.....	61
Figure 43 NAP asset details.....	62
Figure 44 Constants defined in the Jenkinsfile	65
Figure 45 Pipelines execution	66
Figure 46 Results of the execution of the Collaborative Ontology Tool.....	67
Figure 47 Ontologies generated from NeTEx XML schema files	69
Figure 48 Pipelines execution over several ontologies	70
Figure 49 View of the netex_facility_support.owl file	71
Figure 50 Results from scalability test.....	72
Figure 51 NeTEx XSD to OWL.....	74
Figure 52 Create new Item Jenkins.....	74
Figure 53 Setting up GitHub repository on jenkins	75
Figure 54 Build configuration on Jenkins.....	75
Figure 55 Example build execution	77
Figure 56 Build success Log	77
Figure 57 Test execution log	78
Figure 58 Elapsed time for each XSD file.....	79
Figure 59 Results with all measured parameters	79
Figure 60 Netex_facility_version.xsd.....	81
Figure 61 Netex_accounting_version.xsd	81
Figure 62 Scalability results	82
Figure 63 Mapping Tool Execution Time Evaluation	86
Figure 64 TC7 JMeter test plan properties	90
Figure 65 TC7 Asset Manager response times from JMeter	91
Figure 66 Asset Manager TC8 CPU usage during scalability test (yellow=Django, green=Blazegraph).....	96
Figure 67 Asset Manager TC8 RAM usage during scalability test (yellow=Django, green=Blazegraph).....	96

Figure 68 Parametric docker-compose file for the Chimera Converter.	99
Figure 69 Converter and GraphDB memory consumption for TC-11 with 50-csv dataset.	119
Figure 70 Item in Jenkins that executes the test script	129
Figure 71 Performance test results	130
Figure 72 Example of a query with and without preferences	132
Figure 73 Running a query test script for 12 SPARQL endpoint.....	133
Figure 74 Scalability test results.....	134
Figure 75 Asset Manager scalability test TC8: results with 10 parallel requests.....	154
Figure 76 Asset Manager scalability test TC8: results with 50 parallel requests.....	155
Figure 77 Asset Manager scalability test TC8: results with 100 parallel requests.....	155
Figure 78 Asset Manager scalability test TC8: results with 150 parallel requests.....	156
Figure 79 Asset Manager scalability test TC8: results with 200 parallel requests.....	156
Figure 80 Asset Manager scalability test TC8: results with 500 parallel requests.....	157
Figure 81 TC10 JConsole monitoring for Type 1 request	158
Figure 82 TC10 JConsole monitoring for Type 1 response	159
Figure 83 TC10 JConsole monitoring for Type 2 request	160
Figure 84 TC10 JConsole monitoring for Type 2 response	161
Figure 85 TC10 JConsole monitoring for Type 3 request	162
Figure 86 TC10 JConsole monitoring for Type 3 response	163
Figure 87 TC10 JConsole monitoring for Type 4 request	164
Figure 88 TC10 JConsole monitoring for Type 4 response	165
Figure 89 TC10 JConsole monitoring for Type 5 request	166
Figure 90 TC10 JConsole monitoring for Type 5 response	167
Figure 91 TC10 JConsole monitoring for VBB request with <i>crel-m</i> configuration.....	168
Figure 92 TC10 JConsole monitoring for VBB request with <i>frel-m-0</i> configuration	169
Figure 93 TC10 JConsole monitoring for VBB request with <i>frel-m-1</i> configuration	170
Figure 94 TC10 JConsole monitoring for VBB request with <i>frel-m-2</i> configuration	171
Figure 95 TC12 Response time 10 requests for annotation approach.....	174
Figure 96 TC12 Response time 50 requests for annotation approach.....	175
Figure 97 TC12 Response time 100 requests for annotation approach.....	175
Figure 98 TC12 Response time 150 requests for annotation approach.....	176
Figure 99 TC12 Response time 200 requests for annotation approach.....	176
Figure 100 TC12 Response time 500 requests for annotation approach.....	177
Figure 101 TC12 Response time 10 messages for declarative approach.....	178
Figure 102 TC12 Response time 50 messages for declarative approach.....	178
Figure 103 TC12 Response time 100 messages for declarative approach.....	179

Figure 104 TC12 Response time 150 messages for declarative approach.....	179
Figure 105 TC12 Response time 200 messages for declarative approach.....	180
Figure 106 TC12 Response time 500 messages for declarative approach.....	180
Figure 107 TC12 Response time 1 000 messages for declarative approach.....	181
Figure 108 TC12 Response time 2 500 messages for declarative approach.....	181
Figure 109 TC12 Response time 5 000 messages for declarative approach.....	182

LIST OF TABLES

Table 1 List of involved Component/Tools of the IF scenarios defined in D5.4.....	15
Table 2 Scenario S1: Joining the IF Use case (User Registration)	16
Table 3 Scenario S2: Joining the IF Use case (Provider Registration)	19
Table 4 Scenario S4: Distributed service/asset discovery	22
Table 5 Scenario S7: Automated Mapping Process for the Conversion use case	25
Table 6 Summary of terms in Source (FSM) and Target (TransModel) Ontology	29
Table 7 Description of Mapping Labels of the column "Decision" in Figure 17 and Figure 19.....	29
Table 8 Summary of terms in Source (FSM) and Target (IT2Rail).....	31
Table 9 Java Annotation Correctness Evaluation Results for the Test Case 1	34
Table 10 Java Annotation Correctness Evaluation Results for the Test Case 2	35
Table 11 Java Annotation Correctness Evaluation Results for all Test Cases.....	35
Table 12 Scenario S8: Automatic converter building Use case	36
Table 13 Scenario S9: Fast Adaptation to Peaks Use case	41
Table 14 Scenario S10: Special Purpose Asset Discovery Package: Resolver	45
Table 15 Scenario S11: (Collaborative) Ontology Manager.....	53
Table 16 Scenario S12: Ontology Creation using non-ontological sources	57
Table 17 Scenario S13 Asset Manager as National Access Points aggregator	60
Table 18 Scenario S14: Asset Manager as a tool for contributing to a National Access Point	63
Table 19 TC1 Performance Testing for Collaborative ontology manager	64
Table 20 TC2 Scalability Testing for Collaborative ontology manager.....	68
Table 21 TC3 Performance Testing for Automatic generation of ontologies from non-ontological sources	73
Table 22 TC4 Scalability Testing for Automatic generation of ontologies from non-ontological sources	80
Table 23 TC5 Performance Testing Mapping Tool	83
Table 24 TC6 Usability Testing Mapping Tool.....	84
Table 25 Mapping Tool Test Case Description.....	85
Table 26 Summary of Analysis of the Results of Mapping Tool Evaluations.....	87
Table 27 TC7 Performance Testing for Asset manager.....	88
Table 28 TC7 Asset Manager average response time	92
Table 29 TC8 Scalability Testing for Asset manager	93
Table 30 Asset Manager TC8: average response time under increasing load.....	95
Table 31 TC9 Performance Testing for Batch Data Conversion	98
Table 32 Tests for TC9 with the 1-csv dataset	102
Table 33 Tests for TC9 with the 1-csv dataset and optimized lifting mappings.....	103
Table 34 Tests for TC9 with 1-csv, 1-json and 1-xml dataset	104

Table 35 Tests for TC9 with 1-csv, 1-json and 1-xml dataset with optimized lifting mappings	105
Table 36 TC10 Performance Testing for Runtime Data/Message Conversion.....	106
Table 37 TC10 Results of tests for the annotation approach.....	110
Table 38 TC10 Cold start results for the annotation approach	111
Table 39 TC10 Comparison ST4RT and SPRINT	111
Table 40 Tests for TC10 for the Declarative approach	112
Table 41 TC11 Scalability Testing for Batch Data Conversion	113
Table 42 Tests for TC11 with the 10-csv dataset using an in-memory repository	116
Table 43 Tests for TC11 with the 10-csv dataset using an external repository	117
Table 44 Tests for TC11 with the GTFS-Madrid-Bench datasets. Average conversion times in seconds (s) are reported for each dataset.	118
Table 45 Tests for TC11 with 50-csv and external repository.....	119
Table 46 TC12 Scalability Testing for Runtime Data/Message Conversion.....	120
Table 47 TC12 Results of tests for the annotation approach.....	123
Table 48 TC12 Results of tests for the declarative approach	123
Table 49 TC13 Scalability Testing for Runtime Environment Deployment.....	124
Table 50 Average <i>Available</i> time per replica in TC13.....	126
Table 51 TC14 Performance Testing for Distributed SPARQL endpoint.....	127
Table 52 TC15 Scalability Testing for Distributed SPARQL endpoint.....	131
Table 53 Collaborative Ontology Manager tool performance and scalability requirements	135
Table 54 XSD2OWL tool performance and scalability requirements	136
Table 55 Mapping suggestion tool performance and scalability requirements	136
Table 56 Asset Manager performance and scalability requirements	137
Table 57 Converter performance and scalability requirements.....	138
Table 58 Distributed SPARQL endpoint performance and scalability requirements.....	139
Table 59 Functional requirements for SPRINT xsd2owl tool	144
Table 60 Functional requirements for SPRINT Collaborative Ontology Manager tool.....	145
Table 61 Functional requirements for the SPRINT Mapping Suggester	146
Table 62 Functional requirements for the SPRINT Distributed SPARQL Endpoint.....	147
Table 63 Functional requirements for the SPRINT Asset Manager	148
Table 64 Functional requirements for the SPRINT User Manager.....	151
Table 65 Functional requirements for the SPRINT Converter framework.....	152
Table 66 TC8 Summary of results obtained for Asset Manager	154
Table 67 TC10 Complete results for Type 1 request	158
Table 68 TC10 Complete results for Type 1 response	159
Table 69 TC10 Complete results for Type 2 request	160
Table 70 TC10 Complete results for Type 2 response	161

Table 71 TC10 Complete results for Type 3 request	162
Table 72 TC10 Complete results for Type 3 response	163
Table 73 TC10 Complete results for Type 4 request	164
Table 74 TC10 Complete results for Type 4 response	165
Table 75 TC10 Complete results for Type 5 request	166
Table 76 TC10 Complete results for Type 5 response	167
Table 77 TC10 Complete results for VBB request with <i>crel-m</i> configuration.....	168
Table 78 TC10 Complete results for VBB request with <i>frel-m-0</i> configuration	169
Table 79 TC10 Complete results for VBB request with <i>frel-m-1</i> configuration	170
Table 80 TC10 Complete results for VBB request with <i>frel-m-2</i> configuration	171
Table 81 Tests for TC11 with 5-csv, 5-json and 5-xml dataset	172
Table 82 Tests for TC11 with 10-csv, 10-json and 10-xml dataset	172
Table 83 Tests for TC11 with 50-csv, 50-json and 50-xml dataset	173
Table 84 TC12 Average response times for annotation approach.....	174
Table 85 TC12 Average response times for declarative approach	177

1. INTRODUCTION

The SPRINT tools cover different aspects related to the Interoperability Framework:

- Ontology engineering
- Mapping engineering
- Integration of heterogeneous systems
- Governance and publishing automation

Such were the aspects which we deemed as worthy to be investigated in a research project. We tested how our tools support the collaborative development of an ontology, which is a key phase in a large multilateral effort as the IP4 IF. Since modelling parts of a domain are carried out also when developing data schemas, we also evaluated the possibility of automatically generating ontologies out of XML Schemas, therefore providing a first version of the ontology ready for refinement. As the most critical issue in building a Converter is the development of mappings, requiring deep knowledge about both ends of a communication channel, we also created a Mapping suggerter which can guide the mapping developer in his creation task.

Integrating heterogeneous systems usually poses lots of technological requirements, and developers use different styles and techniques to try to cope with them. With Chimera, SPRINT extended a well known open source integration framework to offer semantic-based data conversion capabilities while exploiting the integration options supported by the underlying framework. This means that by using Chimera a developer can exploit a set of pre-defined blocks simply configuring the steps required in the semantic conversion process. Chimera offers different technological solutions for lifting incoming data into RDF (ST4RT Annotations in Java code and RML declarative mappings) and lowering RDF to a target data format (ST4RT Annotations in Java code, Template-based declarative approach).

Moreover, we investigated the possibility to perform distributed queries, therefore avoiding the needs to copy data between systems.

Regarding governance and automation, we implemented a tool (Asset Manager) which can be used to publish and consume descriptions of digital assets according to specific governance processes. As a part of such governance processes, we also offer the possibility to automate technical tasks by integrating the features of Continuous Integration and Deployment (CI/CD).

The evaluation of the final release of the SPRINT tooling collection, as also in D5.3, is organised into three main sections. First of all, we evaluated the functional aspects of our tools (in Section 2) and verified that they were flexible enough to cover a wide set of use cases. Then, we proceeded into evaluating the performance and scalability aspects to assess whether the tools were mature enough to be used in the context of the building of the Interoperability Framework ecosystem (Section 3).

2. FUNCTIONAL VALIDATION

The scenarios of this section cover the use cases which have been described in D5.4, involving the following components/tools of the IF:

Table 1 List of involved Component/Tools of the IF scenarios defined in D5.4

IF Component/Tools	Use Case Scenario
Asset Manager	<p>Scenario S3: Service/Asset Discovery (Simple Discovery)</p> <p>Scenario S4: Service/Asset Discovery (Distributed SPARQL endpoints)</p> <p>Scenario S5: Direct Access use case for Batch Data Conversion</p> <p>Scenario S6: 1.1 Direct download for runtime data/message conversion</p> <p>Scenario S8: Automatic converter building</p> <p>Scenario S9: Fast Adaptation to Peaks</p> <p>Scenario S10: Special Purpose Asset Discovery Package: Resolver</p> <p>Scenario S13: Asset Manager as National Access Points aggregator</p> <p>Scenario S14: Asset Manager as a tool for contributing to a National Access Point</p>
Converter	<p>Scenario S10: Special Purpose Asset Discovery Package: Resolver</p> <p>Scenario S13: Asset Manager as National Access Points aggregator</p> <p>Scenario S14: Asset Manager as a tool for contributing to a National Access Point</p>
User Manager	<p>Scenario S1: Joining the IF Use case (User Registration)</p> <p>Scenario S2: Joining the IF Use case (Provider Registration)</p>
SPARQL endpoint	Scenario S4: Service/Asset Discovery (Distributed SPARQL endpoints)
Asset Discovery	Scenario S3: Service/Asset Discovery (Simple Discovery)
Ontology tools	<p>Scenario S11: (Collaborative) Ontology Manager</p> <p>Scenario S12: Ontology Creation using non-ontological sources</p>
Mapping tools	Scenario S7: Automated Mapping Process for the Conversion use case

S3, S5 and S6 were already validated in C-Rel, and F-Rel implementation was unchanged. Therefore, we omitted to repeat the same contents which were already contained in D5.3.

For all the other scenarios, we will explain how the various tools have been used to validate the fulfilment of the goals.

2.1 SCENARIO S1: JOINING THE IF USE CASE (USER REGISTRATION)

Table 2 Scenario S1: Joining the IF Use case (User Registration)

Actor	HT-train (TSP): a train service provider
Target Component/Sub-system/Entity	User Management
Description	This scenario illustrates the registration process for a user that intends to join IF as a “Service Provider”
Story	<p>Bill is an employee in HT-train which is responsible to register this operator to IF.</p> <p>He goes to IF website of Italy [IF].it and selects to register to IF as Service Provider Role. He is redirected to the IF Identity Provider, which manages identity across all IF components.</p> <p>To create an account, he inserts all the required information related to himself as well as his company, including username, password, type of the company, etc.</p> <p>After successful registration and confirmation of the identity of the registered user, he is then redirected to the Back-Office view of Asset Manager (AM). Back-Office is conceived as the provider's panel in IF and presents required interfaces to the functionalities available for a service provider such as Asset Registration.</p>

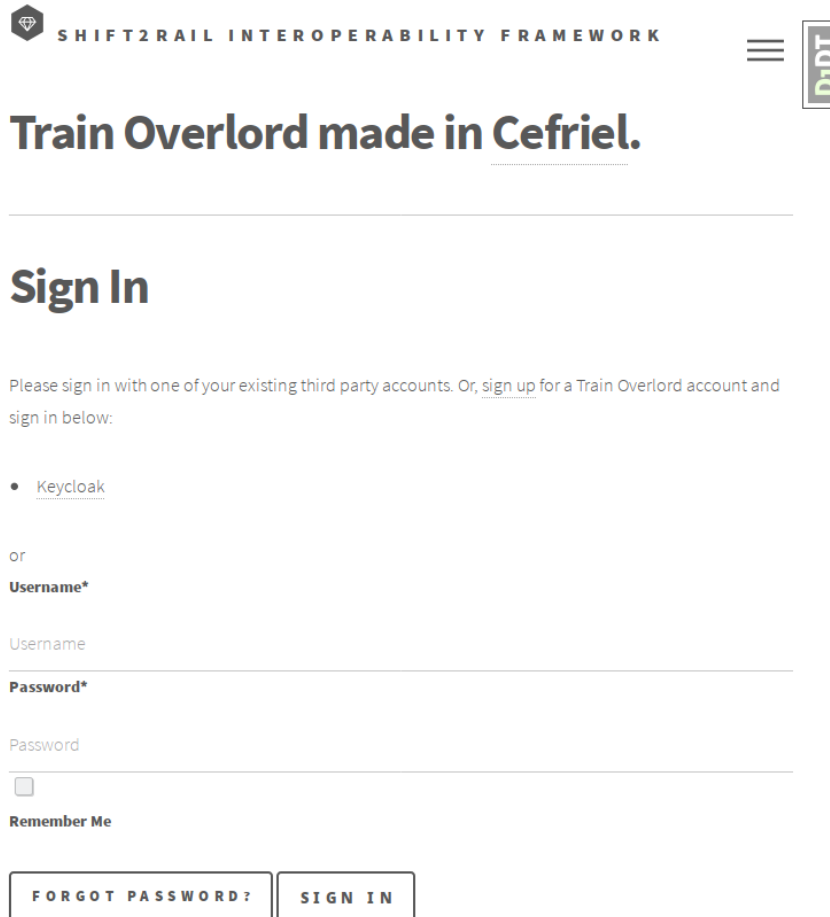
2.1.1 Tools configuration

The Shift2Rail IP4 ecosystem developed by Connective features a OAuth2-based authentication system, which is available at <https://keycloak.shift2railcloud.com/auth/> . We configured the Asset Manager to access an external Identity Provider leveraging on the Django application “allauth”, which is able to delegate authentication to a wide array of external services.

2.1.2 Validation

Whenever the user is trying to either access the Publisher or the protected pages of the Store application, he is redirected to the main login screen. There he's able to either login providing local

credentials, or to login via an external provider. Since we configured a Keycloak external provider, to use Shift2Rail IP4 Single Sign-on the user must press “Keycloak”, as illustrated in Figure 1.



SHIFT2RAIL INTEROPERABILITY FRAMEWORK

Train Overlord made in Cefriel.

Sign In

Please sign in with one of your existing third party accounts. Or, [sign up](#) for a Train Overlord account and sign in below:

- [Keycloak](#)

or

Username*

Username

Password*

Password

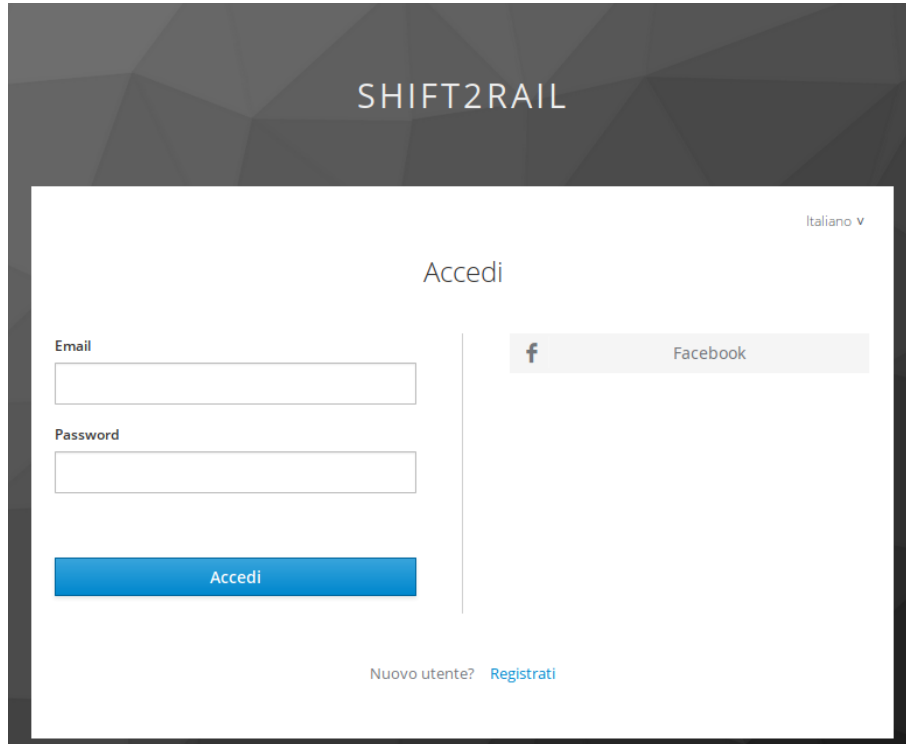
☐

Remember Me

[FORGOT PASSWORD?](#) [SIGN IN](#)

Figure 1 Asset Manager login screen with Keycloak external authentication enabled

The user is then redirected onto the main login screen of the Shift2Rail IP4 Ecosystem, as shown in Figure 2. There he can either login with existing credentials, or register as a new user.



The image shows the login page of the Shift2Rail IP4 ecosystem. The page has a dark grey header with the 'SHIFT2RAIL' logo in white. Below the header is a white login form. The form is titled 'Accedi' (Login) in the center. On the left side of the form, there are two input fields: 'Email' and 'Password'. Below these fields is a blue button labeled 'Accedi'. On the right side of the form, there is a Facebook login option with a Facebook 'f' icon and the text 'Facebook'. At the bottom of the form, there is a link that says 'Nuovo utente? Registrati' (New user? Register).

Figure 2 Shift2Rail IP4 ecosystem Single Sign-On login page

After successful registration or login, the user is redirected to the desired page (either in the Store or in the Publisher).

By default, any new user will be allowed to enter the system as a Consumer with a default set of permissions which will enable him to perform all the operations in the Store application.

2.2 SCENARIO S2: JOINING THE IF USE CASE (PROVIDER REGISTRATION)

Table 3 Scenario S2: Joining the IF Use case (Provider Registration)

Actor	SafeTravel (TrSP): Travel Applications for smartphone
Target Component/Sub-system/Entity	User Management
Description	This scenario illustrates the registration process for a user that intends to join IF as a “Service Consumer”
Story	<p>Alice is an employee in SafeTravel company that develops a smartphone application for ticket search and booking.</p> <p>Alice is responsible to register this transport application to IF. She goes to IF website and she is redirected to the IF Identity Provider, which manages identity across all IF components.</p> <p>To create an account, she inserts all the required information related to herself as well as the company, including username, password, type of the company, etc. She selects to register to IF as Service Consumer Role.</p> <p>After successful registration and confirmation of the identity of the registered user, she is then redirected to the Front-end view of Asset Manager, which is conceived as the consumer's panel in IF and presents required interfaces to the functionalities available for a service provider such as Asset Discovery.</p>

2.2.1 Tools configuration

The configuration of the Asset Manager for S2 is the same as in S1. Since a Provider is supposed to have higher privileges with respect to a Consumer, a privilege escalation workflow has been implemented.

When the user logs in the first time, he is considered as a Consumer with a default set of privileges (as described in S1). Assigning him Provider privileges requires approval from an administrator, and we decided to implement this request via a BPMN process which is shown in Figure 3.

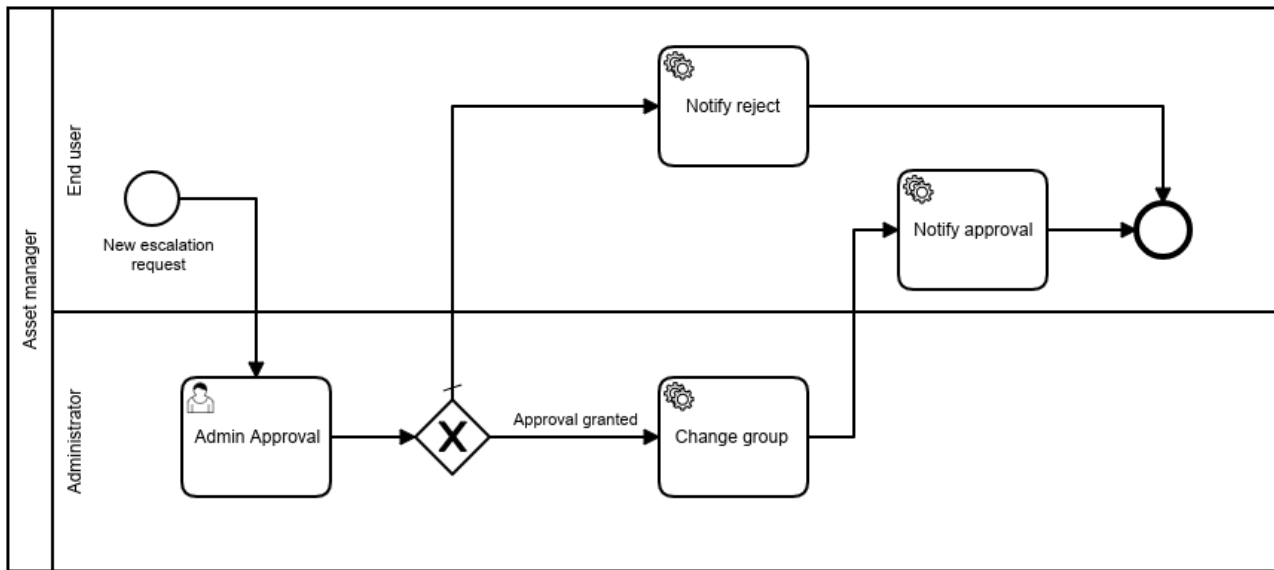


Figure 3 Escalation to Provider – BPMN process

2.2.2 Validation

We are now assuming that the steps described for Scenario 1 have already been accomplished, and the user is a newly registered user with a confirmed email. After registering as a Consumer, the user is able to access the Asset Manager Store. If he tries to access the Publisher application, he is redirected to the page shown in Figure 4.

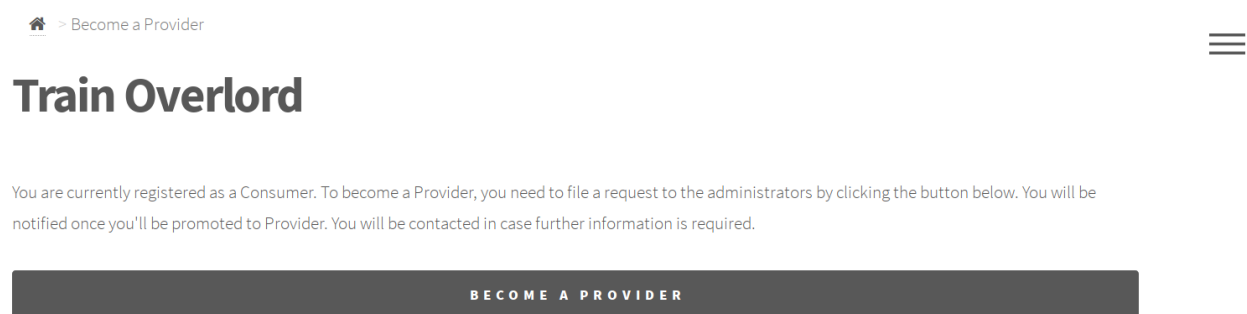


Figure 4 Escalation to Provider – Triggering the request

As soon as the user presses the “Become a Provider” button, the BPMN process described before is triggered, and the request is sent to the Administrator, who is also notified about the request via email (as depicted in Figure 5). The Administrator then accesses the “Tasks” section in the Publisher application (Figure 6), and he is able to grant or reject the user’s request, as shown in Figure 7. The user is then notified via email about the Administrator’s decision. In case the request is approved, the BPMN engine calls the appropriate Asset Manager API to assign the user to the Provider group.

The user then acquires all the privileges linked to such group (which can be defined in the administration console).

New notification from Shift2Rail IF Ecosystem

Dear Administrator,
Shift2Rail IF Ecosystem has a new notification for you:

The user alessio.carenini@cefriel.com is asking to be promoted into the Providers group. Please check the request inside the Publisher application.

Best regards

Shift2Rail IF Ecosystem

 Reply

 Forward

Figure 5 Request to become a Provider – Notification to the Administrator



 > Tasks

User tasks

Privilege escalation approval

 OPEN TASK

 DELETE

Figure 6 Request to become a Provider – Administrator's tasks list



 > Tasks and notifications

User task

Do you want to grant user alessio.carenini@cefriel.com the permission to access the Publisher application?

Yes

SUBMIT

Figure 7 Request to become a Provider – Decision task for the Administrator

2.3 SCENARIO S4: DISTRIBUTED SERVICE/ASSET DISCOVERY

Table 4 Scenario S4: Distributed service/asset discovery

Actor	<p>B-Com: A Belgian-based TSP 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: 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>
Target Component/Sub-system/Entity	Distributed SPARQL endpoint
Description	This scenario illustrates querying the SPARQL engine on several endpoints
Story	<p>MyMobility is interested in expanding its routes services across Europe, it wants to discover which service providers are publishing public transport data that may be interesting for them. Once that IF receives a request from MyMobility about catalogues describing data of public transport providers, a distributed SPARQL query process is started. The company requests the list of publishers (transport service providers) of the datasets containing information of the different public means of transport.</p> <p>Since MyMobility is focused on Belgium, Spain and 10 other countries to expand its routes services, we treat S-Com, B-Com, V-com₁, V-com₂, ..., V-com₁₀ as twelve different data sources, whose resources can be combined to find metadata catalogues of the NAP datasets. For this, IF needs to make use of the services provided by B-Com, S-Com, V-com₁, V-com₂, ..., and V-com₁₀ independently of where they are coming from. As such, IF will issue the query to the asset manager distributed SPARQL query engine, which will perform the usual steps of generation of subqueries for each selected source, generating a query plan, rewriting the subqueries considering potential inferences, translating those subqueries and executing them so that the results can be integrated and delivered to the asset manager.</p>

	<p>To expand its route services in Madrid, MyMobility refines its search after an initial request for catalogues to discover a more detailed information about the Spain datasets comprising descriptions of the stations, bus stops and other infrastructures of S-Com.</p> <p>Also, MyMobility wants to further filter its results by using preferences. For example, MyMobility can specify preferences on update frequency and creation date of the datasets belonging to the catalogues in order to find those datasets that were initially updated frequently but are now stable because they are no longer updated, i.e., the most frequently updated datasets (qual_freq) with the oldest last modified time (metadata_date).</p>
--	---

2.3.1 Tools configuration

The distributed SPARQL endpoint has to access twelve SPARQL interfaces (see Figure 8) which host RDF datasets and whose resources need to be combined for answering the input query. The engine used for this aim is Ontario. It will execute a federated SPARQL query over several SPARQL interfaces and combine resources from all of them in order to answer the input query.

The installation and deployment is done using Docker containers and the documentation is detailed in the GitHub¹ repository.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d5e17aaa4aea	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 52 seconds	0.0.0.0:1129->1111/tcp, 0.0.0.0:11396->8890/tcp	com10
b8f39ae7e99b	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 53 seconds	0.0.0.0:1121->1111/tcp, 0.0.0.0:11388->8890/tcp	com3
67b583949a9e	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 52 seconds	0.0.0.0:1124->1111/tcp, 0.0.0.0:11391->8890/tcp	belgium1
e430a7fcd0a1	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 51 seconds	0.0.0.0:1119->1111/tcp, 0.0.0.0:11384->8890/tcp	com1
fbf4c55ee97a	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 50 seconds	0.0.0.0:1120->1111/tcp, 0.0.0.0:11387->8890/tcp	com2
85c38ba7324f	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 48 seconds	0.0.0.0:1125->1111/tcp, 0.0.0.0:11392->8890/tcp	com6
1358228f4f46	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 54 seconds	0.0.0.0:1126->1111/tcp, 0.0.0.0:11393->8890/tcp	com7
aec3ba6579e7	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 56 seconds	0.0.0.0:1128->1111/tcp, 0.0.0.0:11395->8890/tcp	com9
a9a63b682f3c	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 51 seconds	0.0.0.0:1118->1111/tcp, 0.0.0.0:11383->8890/tcp	spain1
789240202a78	kemele/ontario:0.5	"/Ontario/start_spar..."	About a minute ago	Up 59 seconds	0.0.0.0:6001->5000/tcp	ontario2
478a68f375d4	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 58 seconds	0.0.0.0:1123->1111/tcp, 0.0.0.0:11390->8890/tcp	com5
5d8973dcb2b5	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 57 seconds	0.0.0.0:1127->1111/tcp, 0.0.0.0:11394->8890/tcp	com8
e24f8ba69fa8	kemele/virtuoso:7-stable	"/bin/bash /virtuoso..."	About a minute ago	Up 55 seconds	0.0.0.0:1122->1111/tcp, 0.0.0.0:11389->8890/tcp	com4

Figure 8 SPARQL interfaces with metadada in RDF from Belgium and Spain

2.3.2 Validation

For the demonstration, as a proof of concept, the distributed SPARQL endpoint accesses twelve different SPARQL interfaces which host different RDF datasets and whose resources need to be combined. Providing to Ontario the information about the SPARQL interfaces using the RDF-MT metadata, the engine is able to generate subqueries that are answers by each SPARQL interface, depending on its resources. Lastly, Ontario combines the results from the 12 endpoints in a single answer or SPARQL result set. As an example, Figure 9 shows a query executed on two SPARQL endpoints (top) together with the number of results obtained (bottom), while Figure 10 shows the result when running the same query over the twelve SPARQL endpoints.

¹ <https://github.com/oeg-upm/sprint/tree/main/Ontario>

Query

```
# Query for the frequently updated datasets (qual_freq) with the modified time (metadata_date)
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT * WHERE {
  ?d a dcat:Dataset .
  ?d dct:qualFreq ?qf .
  ?d dct:metadataDate ?md .
}
```

Result with 2 endpoint

```
...
...
"md": {
  "datatype": "http://www.w3.org/2001/XMLSchema#date",
  "type": "typed-literal",
  "value": "2019-09-20"
},
"qf": {
  "datatype": "http://www.w3.org/2001/XMLSchema#int",
  "type": "typed-literal",
  "value": "9"
}
}
],
"totalRows": 31,
"vars": [
  "md",
  "qf",
  "d"
]
}
```

Figure 9 Query and results obtained over two endpoints

Result with 12 endpoints

```
...
...
{
  "d": {
    "type": "uri",
    "value": "https://gtfs.irail.be/metadata#tec"
  },
  "md": {
    "datatype": "http://www.w3.org/2001/XMLSchema#date",
    "type": "typed-literal",
    "value": "2019-11-16"
  },
  "qf": {
    "datatype": "http://www.w3.org/2001/XMLSchema#int",
    "type": "typed-literal",
    "value": "844"
  }
}
],
"totalRows": 159,
"vars": [
  "md",
  "d",
  "qf"
]
}
```

Figure 10 Results obtained over twelve endpoints

2.4 SCENARIO S7: AUTOMATED MAPPING PROCESS FOR THE CONVERSION USE CASE

Table 5 Scenario S7: Automated Mapping Process for the Conversion use case

Actor	Best_Travel (ISA): A service/application provider
Target Component/Sub-system/Entity	Converter: Mapping IDE
Description	This scenario describes utilisation of Mapping IDE.
Story	<p>According to Best_Travel analysis, the Standard-K is becoming more and more popular and widely used that can substitute the other famous but legacy Standard-P. So, they decide to develop a K-P converter and publish it in the market for potential users. To this end, they need to generate the “Mapping” between the concepts and terms in both standards, as well as, the Java annotations.</p> <p>John is an ontology engineer and specialist in the transport domain standardization in Best_Travel who is part of the team for developing the converter.</p> <p>Mary is a member of the team in Best_Travel organization, who is a software engineer and knows the Mapping Tool of the IF. She has already run the tool utilizing its docker image.</p> <p>By initializing the tool, she goes through a wizard to provide the program with the required resources to start the process. To this end, she selects the format of source and target file (XML, OWL, ttl, etc) and afterward, she can select the files.</p> <p>Default outputs are the Java annotated files (which are then the required inputs for Converter components) but a user can configure the tool to receive the actual mappings (one-to-one translation of terms from source to target standard) as well.</p> <p>After the successful uploading of the standards, the tool starts the mapping process and notifies user upon the termination of the job. At this point, Mary views a list of the concepts in source format and the suggested equivalent concepts to the target standards which are ranked based on the probability of being similar.</p> <p>So, Mary, with the help of John, can go through suggestions, where the top-ranked suggestion is considered as the confirmed mapping by default. Nevertheless, they then can select another suggestion as the</p>

	<p>confirmed one, or, manually add a term if the desired term is not among the suggestions.</p> <p>When they are done with reviewing all the mappings, the tool proceeds with the generation of the final outputs which are the mappings and annotated java files. The final outputs, then, would be stored in the desired directory specified by the user.</p>
--	---

2.4.1 Tools configuration

Extract the contents of the provided package into a folder (it is then the root folder of the application)

- Navigate to the frontend/angularUI folder in order to change the IP address of the server that will be hosting the frontend application
 - in src/environments folder open the environment.prod.ts file and change the API_Endpoint value to the host machine's actual IP address, leaving the api/v1 suffix as is. The final value should look like this:

`http(s)://your.hostname.or.domain/api/v1`

- Navigate back to the root directory and start the docker-compose process using the docker-compose.yml. This will start the build process of the application for both frontend and backend parts.
- (This step is optional if the package already contains the mentioned files. If those files have been downloaded separately this step is mandatory) Once the building process is finished put the jaxb_impl-0.1.3.jar and the model.bin files into the input folder.
- Now start the docker containers using docker-compose once more.

If it is required to change the internal API service port, both the docker-compose file in the root folder and the NginX proxy configuration file need to be modified according to the desired configuration: change the port number in the command line of the docker-compose file that starts the uvicorn server for the API. If the NginX service is used as well the aforementioned change shall be reflected in the upstream block of the proxy.conf file. Both files shall contain the same port number for the application to work unless another custom service is to be used instead of NginX otherwise the API won't work. To summarize the needed steps:

- In the root folder, open the docker-compose file and change the last parameter in line command: [...] that indicates the port on which the API will be running
- Navigate to the frontend/angularUI/service_conf folder and open the proxy.conf file with a standard text editor.
- The setting inside upstream block represents the internal process port the backend API will be running on and to which the NginX will forward http requests. This is not meant to be exposed to an external user necessarily and it is for internal use only. Change the parameters to be identical to the choice made in the previous step.

In the default configuration, the application uses ports 80 and 8081 on the host machine, which could be changed at any time according to the user preferences.

After the successful running of the application, it is accessible from browser, and the home page looks like Figure 11.

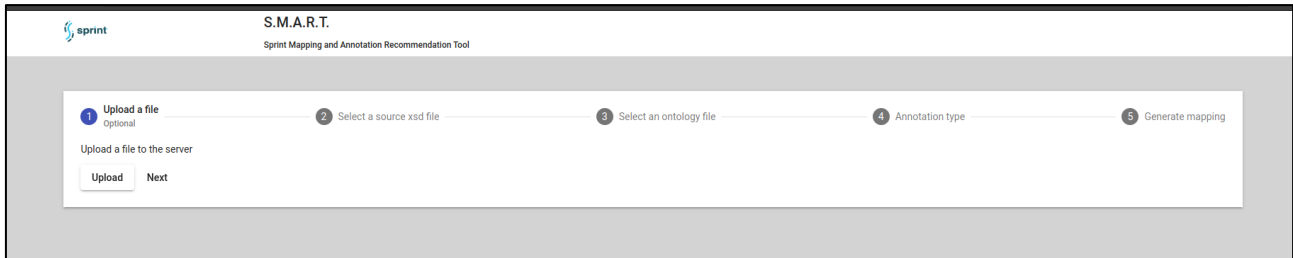


Figure 11 Home page of the mapping and annotation application

Following the starting wizard, user can upload the files of the source and target standards, configure the desired annotation type (Java-based or RML-based annotation), as represented in Figure 12 , and the initiate the mapping generation as shown in Figure 13.

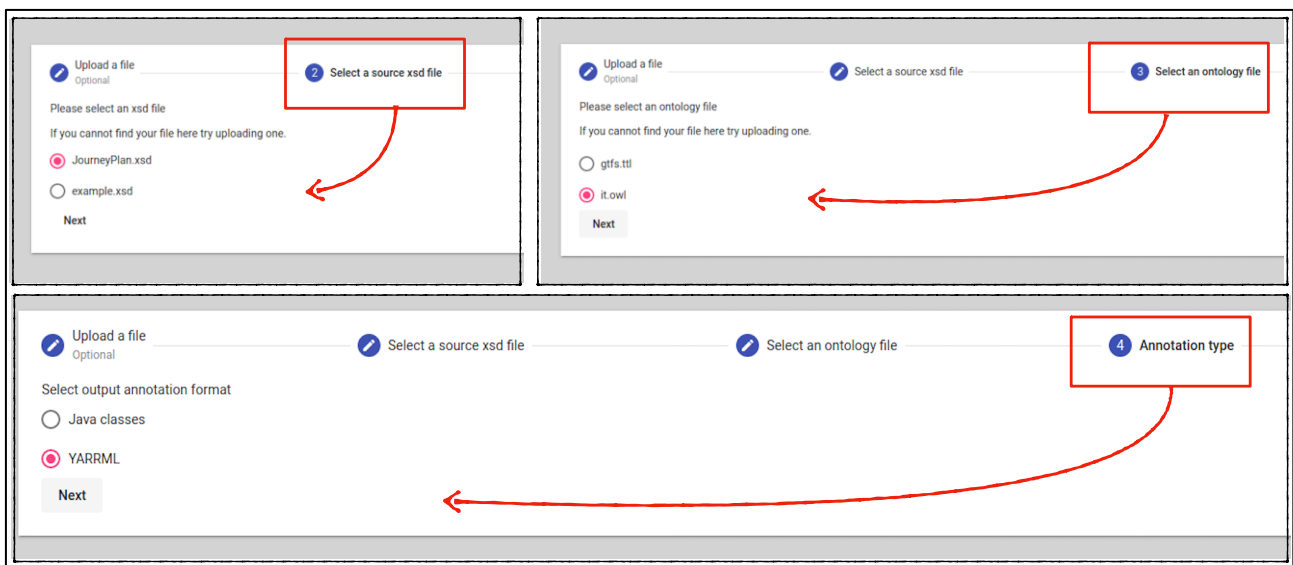


Figure 12 Starting Wizard

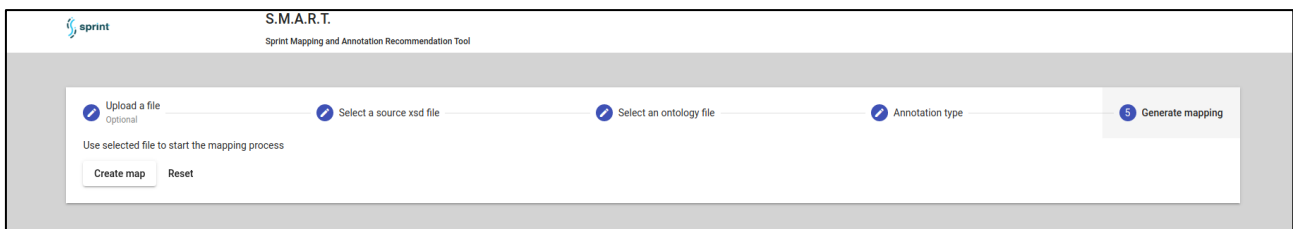


Figure 13 Start Mapping Generation

When the application generates the mapping, it represents them, pair by pair, along with the confidence level (High, Medium and Low) indicating the probability of similarity of the particular terms in the source and target standards. Figure 14 shows an example a suggested pair.

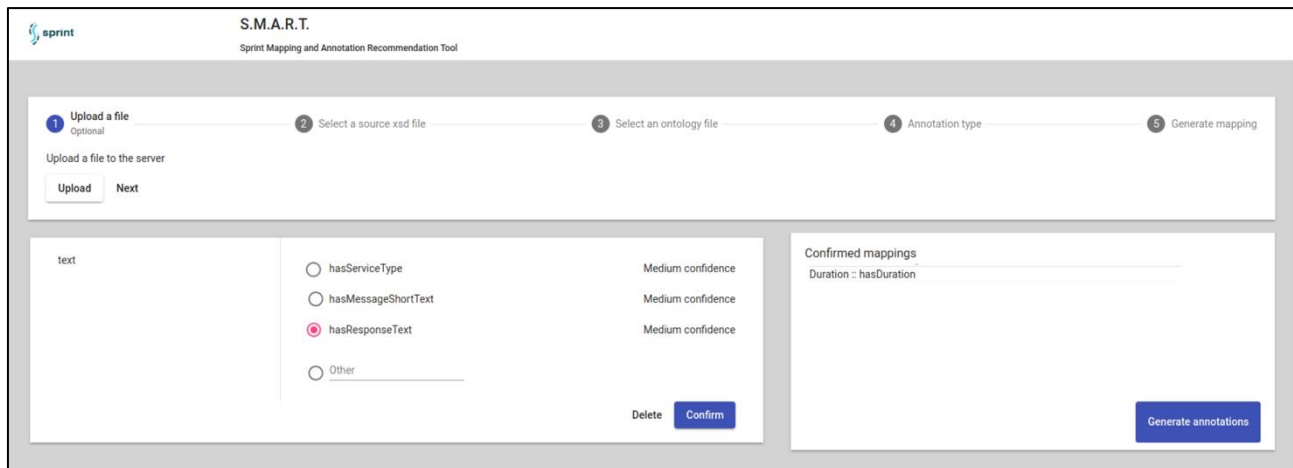


Figure 14 Mapping Suggestions

After the selection of all the suggestions, the user can proceed to generate the annotation and finally download the outputs as represented in Figure 15.

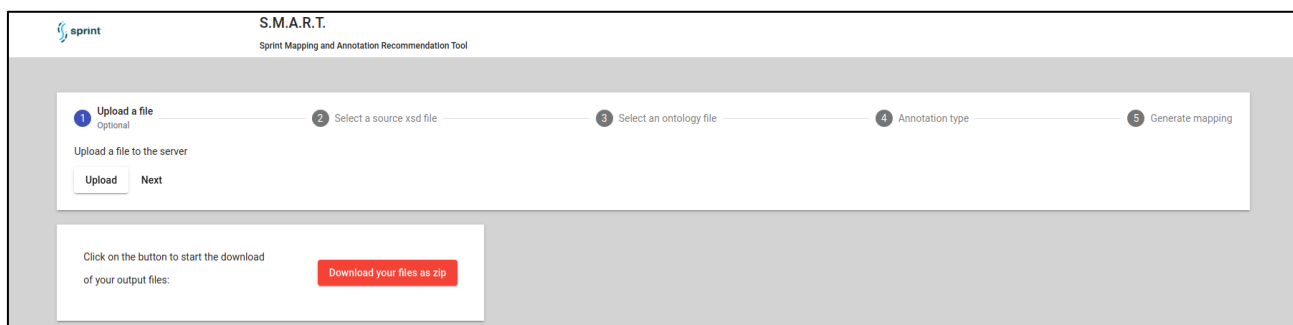


Figure 15 Output generation

2.4.2 Validation

Mapping Generation

For given input Source and Target files, the Mapping Tool suggests mappings between terms of the Source and Target standards. The formats accepted by the tool include OWL, XSD, and XML.

The accuracy evaluation has been carried out as following:

- we chose 2 pairs of {source, target} standards; we call each pair a validation “scenario”;
- for each pair of standards, we ran the Mapping Tool and produced a set of pairs of terms { source_term_i, target_term_j } whose mapping is suggested by the tool;
- for each scenario, we manually checked the mapping suggested by the tool, to evaluate the accuracy obtained for the scenario.

We use the following formula to calculate the overall accuracy of the Mapping Tool for each scenario.

$$\text{Accuracy percentage} = (\text{Number of correct mappings} / \text{Number of total mappings}) * 100$$

where the meaning of the terms used in the formula is the following:

Number of correct mappings: Number of accurately mapped pairs according to the manual evaluation for the corresponding scenario.

Number of total mappings: Number of total mappings that has been produced by tool.

The Mapping Tool has been tested against two scenarios, which are described in the rest of this section. For the chosen pairs of standards either existing manual mapping is available; or pairs for which there was sufficient knowledge to evaluate the reasonableness of the suggested mapping.

Test Case 1: Mapping between FSM to TransmodelOntology

In Case Study 1 we considered test case as FSM standard, and the Transmodel Ontology that seemed to include a number of concepts that should be available in FSM standard. In addition, the terms of the FSM standards are sufficiently well commented that our confidence in the accuracy of our manual evaluation the mappings is good. Table 6 summarizes the sets of terms retrieved from the source and target standards.

Table 6 Summary of terms in Source (FSM) and Target (TransModel) Ontology

	Source terms FSM	Target terms Transmodel Ontology
Unique terms in source	98	231
Terms remaining after filtering	82	220
Unique mapped pairs	66	

Figure 16 shows a snippet of the manual evaluation of the mappings suggested by the Mapping Tool when run on the FSM and Transmodel standards. While evaluating the suggested mappings, we realized that, for some concepts in the source standard (FSM), no equivalent concept was actually available in the target standard (Transmodel). We label those pairs as “MayBe” or “Don’t Know”.

Table 7 Description of Mapping Labels of the column "Decision" in Figure 17 and Figure 19

Decision	Description
Correct	The mapping suggested by the Mapping Tool is correct.
Incorrect	The mapping suggested by the Mapping Tool is incorrect, though a correct one seems to exist.
UnFeasible	Expected correct mapping doesn't exist in target standard
MayBe	The mapping produced by Mapping Tool is possibly accurate, or inaccurate.
Don'tKnow	The mapping produced by Mapping Tool can't be categorized in accurate or inaccurate

Table 7 shows description of these labels used in Figure 16.

source_term	mapped_term	Decision	confidence_score
0 AccessEquipmentId	accessMode	Incorrect	0.557144820690155
1 Acronym	changeOfServiceRequirements	Incorrect	0.697268545627594
2 ArrivalTime	arrivalTime	Correct	0.7699399524264865
3 BoardingPosition	BoardingPosition	Correct	0.7894401788711548
4 BoardingPositionId	vehicleModelId	Incorrect	0.7178292274475098
5 BorderPointCode	timingLinkPointSequence	Incorrect	0.7511488596598307
6 ChangePoint	BeaconPoint	Don'tKnow	0.81087327003479
7 CheckInTime	arrivalTime	Incorrect	0.7751647631327311
8 CloseOfBoarding	forBoarding	Maybe	0.734955628712972
9 CompartmentId	changeOfServiceRequirements	Incorrect	0.5254254341125488
10 Connection	endsConnection	Incorrect	0.7738086382548014
11 ConnectionTime	endsConnection	Maybe	0.7944939136505127
12 DateTime	passingTimeTimingPoint	UnFeasible	0.754897952079773
13 Description	dayTypeAssignment	Don'tKnow	0.6143476366996765
14 Destination	destinationDisplay	Maybe	0.5531281948089599
15 EffectiveDeparture	lastDepartureTime	Correct	0.7299173672993978
16 ExperiencedOverallChangeTime	waitTime	Maybe	0.7852698961893717
17 FootpathEquipmentId	vehicleModelId	Incorrect	0.6708360314369202
18 GenericStatus	DefaultServiceJourneyRunTime	UnFeasible	0.6467540264129639
19 GenericTextMessage	StopPointInJourneyPattern	UnFeasible	0.5219136476516724
20 GeoPoint	BeaconPoint	Don'tKnow	0.7960529327392578
21 ImpairedOverallChangeTime	lineNumber	Incorrect	0.6028727889060974
22 IsBorderPoint	timingLinkPointSequence	Incorrect	0.7531669934590658
23 Itinerary	Trip	Correct	0.5276892185211182
24 LanguageId	vehicleModelId	UnFeasible	0.7612857818603516
25 LengthPos	changeOfServiceRequirements	Incorrect	0.5091819763183594
26 LinkPath	LinkSequence	Don'tKnow	0.768217134475708
27 MeanOverallChangeTime	arrivalTime	Incorrect	0.7826924324035645
28 MobilityAidTypeId	holidayType	Incorrect	0.702202320098877
29 Name	lineName	Maybe	0.7234164476394653
30 NavigationPoint	BeaconPoint	Don'tKnow	0.80415940284729
31 NeighbourPassengerIdRefsList	changeOfDestinationDisplay	Incorrect	0.6001829504966736
32 OnboardServiceCategoryId	typeOfFlexibleService	Maybe	0.59874027967453
33 PathLinkTypeId	timingLink	Incorrect	0.7835566202799479
34 PedestrianPacId	vehicleModelId	Incorrect	0.6614804267883301
35 PlaceReference	TopographicPlace	Incorrect	0.7571753859519958

Figure 16 FSM to Transmodel mappings provided by MappingTool

Figure 17 shows a summary of the findings of the manual evaluation of the mappings suggested by the Mapping Tool for the FSM and Transmodel standards. From description can be seen that out of 66 unique mapped pairs around 13 are correctly mapped, 18 of those mapped are uncertain. Whereas 20 terms from source are labeled as “UnFeasible” due to the fact that their corresponding matching terms doesn’t exist in target standard, while 14 are mapped incorrect.

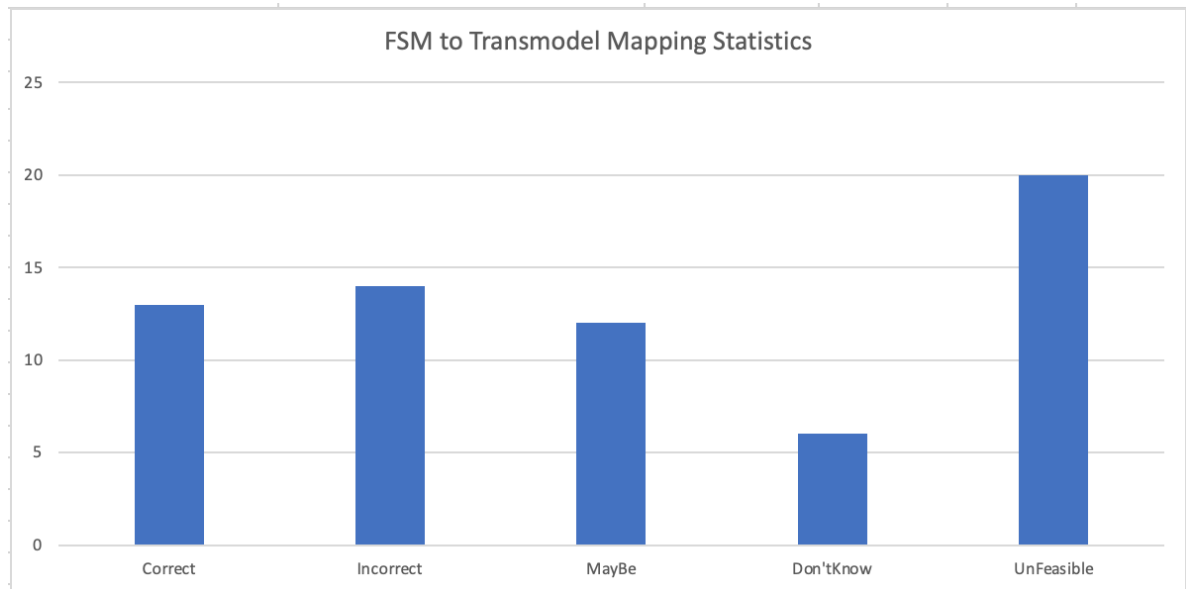


Figure 17 FSM to Transmodel Mapping Statistics

For given statistics from manual evaluation accuracy of the tool for current scenario is calculated as total number of correct mappings vs total number of feasible mappings which leads to figures as $(13/27) \times 100 = 48\%$. It can be said that for scenario 1 Mapping tool produces 38% accurate mappings.

Test Case 2: Mapping Between FSM standard and IT2Rail Ontology.

In the second test case, we considered the ontology developed within the IT2Rail project, and the FSM standard. In particular, we chose a subset of the FSM standard (similar subset that one was used in Test case 1), that seemed to include a number of concepts that should also be available in the IT2Rail ontology.

Table 8 Summary of terms in Source (FSM) and Target (IT2Rail)

Terminology	Source terms FSM	Target terms IT2Rail
Unique terms in source	83	555
Terms remaining after filtering	82	527
Unique mapped pairs	80	

Figure 17 shows a snippet of the manual evaluation of the mappings suggested by the Mapping Tool when run on the FSM and IT2Rail standards.

source_term	mapped_term	Decision	confidence_score
0 AccessEquipmentId	accessMode	Incorrect	0.557144820690155
1 Acronym	changeOfServiceRequirements	UnFeasible	0.697268545627594
2 ArrivalTime	arrivalTime	Correct	0.7699399524264865
3 BoardingPosition	BoardingPosition	Correct	0.7894401788711548
4 BoardingPositionId	vehicleModeld	Incorrect	0.7178292274475098
5 BorderPointCode	timingLinkPointSequence	Incorrect	0.7511488596598307
6 ChangePoint	BeaconPoint	Don'tKnow	0.81087327003479
7 CheckInTime	arrivalTime	UnFeasible	0.7751647631327311
8 CloseOfBoarding	forBoarding	MayBe	0.734955628712972
9 CompartmentId	changeOfServiceRequirements	UnFeasible	0.5254254341125488
10 Connection	endsConnection	Incorrect	0.7738086382548014
11 ConnectionTime	endsConnection	MayBe	0.7944939136505127
12 DateTime	passingTimeTimingPoint	UnFeasible	0.754897952079773
13 Description	dayTypeAssignment	Don'tKnow	0.6143476366996765
14 Destination	destinationDisplay	MayBe	0.5531281948089599
15 EffectiveDeparture	lastDepartureTime	Correct	0.7299173672993978
16 ExperiencedOverallChangeTime	waitTime	MayBe	0.7852698961893717
17 FootpathEquipmentId	vehicleModeld	UnFeasible	0.6708360314369202
18 GenericStatus	DefaultServiceJourneyRunTime	UnFeasible	0.6467540264129639
19 GenericTextMessage	StopPointInJourneyPattern	UnFeasible	0.5219136476516724
20 GeoPoint	BeaconPoint	Don'tKnow	0.7960529327392578
21 ImpairedOverallChangeTime	lineNumber	Incorrect	0.6028727889060974
22 IsBorderPoint	timingLinkPointSequence	Incorrect	0.7531669934590658
23 Itinerary	Trip	Correct	0.5276892185211182
24 LanguageId	vehicleModeld	UnFeasible	0.7612857818603516
25 LengthPos	changeOfServiceRequirements	UnFeasible	0.5091819763183594
26 LinkPath	LinkSequence	Don'tKnow	0.768217134475708
27 MeanOverallChangeTime	arrivalTime	Incorrect	0.7826924324035645
28 MobilityAidTypeId	holidayType	UnFeasible	0.702202320098877
29 Name	lineName	MayBe	0.7234164476394653
30 NavigationPoint	BeaconPoint	Don'tKnow	0.80415940284729
31 NeighbourPassengerIdRefsList	changeOfDestinationDisplay	UnFeasible	0.6001829504966736
32 OnboardServiceCategoryId	typeOfFlexibleService	MayBe	0.59874027967453
33 PathLinkTypeId	timingLink	Incorrect	0.7835566202799479
34 PedestrianPaceld	vehicleModeld	Incorrect	0.6614804267883301
35 PlaceReference	TopographicPlace	Incorrect	0.7571753859519958

Figure 18 FSM to IT2Rail mappings provided by MappingTool

Figure 19 shows a summary of the findings of the manual evaluation of the mappings suggested by the Mapping Tool for the FSM and IT2Rail standards. From description can be seen that out of 80 unique mapped pairs 32 are correctly mapped, 31 of those mapped are uncertain while 9 are mapped incorrect. 8 of those mappings are labeled as “UnFeasible”.

For given statistics from Figure 19 the accuracy of the tool for current scenario is calculated as total number of correct mappings vs total number of feasible mappings which leads to figures as $(32/41) \times 100 = 78\%$. It can be said that for scenario 2 Mapping tool produces 78% accurate mappings.

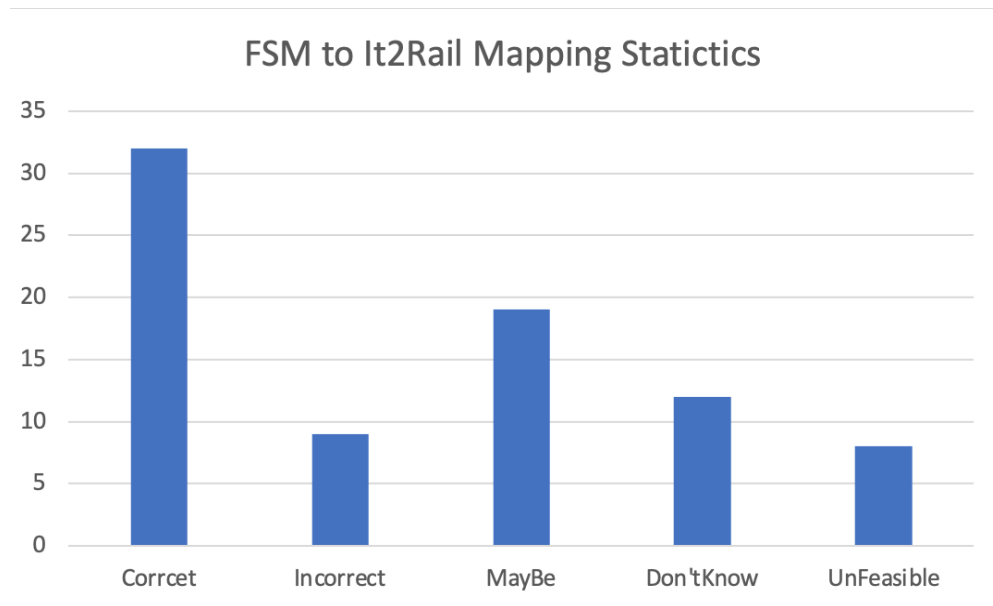


Figure 19 FSM to IT2Rail Mapping Statistics

Given above evaluation for 2 testcases for mapping Tool overall accuracy of the tool can be calculated as average of accuracy of both testcases that is:

$(\text{Scenario 1 average accuracy} + \text{Scenario 2 average accuracy})/2$ which leads to results as $(48+78)/2= 63\%$.

Therefore given evaluation against two testcases it can be concluded that overall accuracy of Mapping tool is 63%.

Annotation Generation

In this section, we start by evaluation the Java-based annotation and then we examine the RML-based annotation.

Java-based annotation

In following, we explain the procedure that we have followed to evaluate the correctness of the Java Annotation functionality, and dicuss the obtained results. We ran the tool with two runs as test cases. In each cases we followed the following steps for each mapping pairs:

1. Check if the syntax of the generated annotations is correct according to the tool's specification. Results are reported in the row #2 of Table 9 and Table 10 .
2. Search the expected Java files to see if the annotations are placed in the expected lines of the Java files or not. Results are reported in the rows #3 and #5 of Table 9 and Table 10.
3. Search all the Java files to see if any of the annotations are added in a wrong place. Results are reported in the row #4 of Table 9 and Table 10.
4. Computing the accuracy without considering the JAXB's limitation. Results are reported in the row #6 of Table 9 and Table 10.

5. Computing the accuracy by considering the JAXB's limitation. Results are reported in the row #7 of Table 9 and Table 10.

Figure 20 provides a partial snapshot of an annotated Java file. The added annotations are marked using red rectangles. As we can see the annotation mechanism has annotated the StartAndDuration which is a Class with appropriate annotation right before the start of the Class definition; and, added appropriate annotations for the start and duration properties in the correct places.

```
@RdfsClass("IT2Rail:Date")
public class StartAndDuration {

    @RdfProperty(propertyName = "IT2Rail:hasStartDateTime")
    @XmlElement(name = "Start")
    @XmlSchemaType(name = "dateTime")
    protected XMLGregorianCalendar start;
    @RdfProperty(propertyName = "IT2Rail:hasDuration")
    @XmlElement(name = "Duration")
    protected Duration duration;
```

Figure 20 Partial snapshot of an annotated Java file. The added annotations are marked by red rectangles.

Test Case 1:

The first test case, is the FSM to Transmodel mappings. Table 9 reports the results obtained from this test case.

Table 9 Java Annotation Correctness Evaluation Results for the Test Case 1

Number selected mappings :	67
Number of correct generated annotations:	67
Number of correct annotations added into the java files:	65
Number of wrong annotations added into the java files:	0
Number of missed annotations:	2
Accuracy by not considering JAXB limitation	100%
Accuracy by considering JAXB limitation	97%

Test Case 2:

The second test case is FSM to IT2Rail and Table 10 reports the results obtained from this test case.

Table 10 Java Annotation Correctness Evaluation Results for the Test Case 2

Number selected mappings:	81
Number of correct generated annotations:	81
Number of correct annotations added into the java files:	79
Number of wrong annotations added into the java files:	0
Number of missed annotations:	2
Accuracy by not considering JAXB limitation	100%
Accuracy by considering JAXB limitation	97.5%

Finally, Table 11 provides the overall the evaluation by considering both test cases.

Table 11 Java Annotation Correctness Evaluation Results for all Test Cases

Number of mappings:	148
Number of correct generated annotations:	146
Number of correct annotations added into the java files:	144
Number of wrong annotations added into the java files:	0
Number of missed annotations:	4
Accuracy by not considering JAXB limitation	100%
Accuracy by considering JAXB limitation	95.25%

In conclusion, as we can see in the results presented above, the tool is accurately capable of performing the Java Annotation task with an accuracy of 100%. The only limitation is related to the JAXB limitation. Specifically, in an XSD file which will be the input of the JAXB, if there exist any term which its type is “element”, in some cases, JAXB is not capable of generating property for that term in the corresponding Java Class. Considering this limitation, the tool could reach the accuracy of 95.28% for the java annotation generation task which still a very good results.

RML-based annotation

To test the functionality of the in generating RML based annotations (YARRRML declarations), we performed the test cases discussed in previous chapter. To test correctness of the generated outputs from syntactical point of view, we test the results using the `yarrml-parser`². All test cases could successfully pass the parsing tests.

2.5 SCENARIO S8: AUTOMATIC CONVERTER BUILDING USE CASE

Table 12 Scenario S8: Automatic converter building Use case

Actor	<p>N-rail: a rail <u>TSP</u> which just joined the Shift2Rail ecosystem.</p> <p>Y-bus and X-bus: bus <u>TSPs</u> already part of the Shift2Rail ecosystem.</p>
Target Component/Sub-system/Entity	Asset Manager, Converter
Description	A new operator is interested in establishing a new business by communicating with other operators who already joined the IF ecosystem. Since those operators are compliant with the Shift2Rail Ontology, he just needs to provide the mapping between the messages used by his IT systems and the reference ontology. The Asset Manager will then be able to assemble a Converter, composing the different mapping and the required ontologies and data sets. Such Converter will be then used by the operator to effectively connect his system to the ones provided by the other operators.
Story	Y-bus services joined the S2R ecosystem and contributed a Converter to let its clients interact with X-bus, an allied bus operator. It does so by providing a mapping which “lifts” its own data model to the Shift2Rail ontology, and also a mapping which “lowers” instances of the S2R ontology to the X-bus data model.

2.5.1 Tools configuration

The F-Rel version of S8 builds upon the results of the C-Rel implementation. The basic system behavior remains the same, so after a successful publication of a Converter, a Jenkins job takes care of building deployable artifacts.

With respect to C-Rel implementation, F-Rel S8 adds dependency tracking and Kubernetes artifacts creation.

² <https://github.com/RMLio/yarrml-parser>

As depicted in Figure 21, a Converter asset adopting the SPRINT framework relies on the existence of several other assets:

- A Lifting mapping
- A Lowering mapping
- Multiple Ontologies
- Multiple RDF Datasets

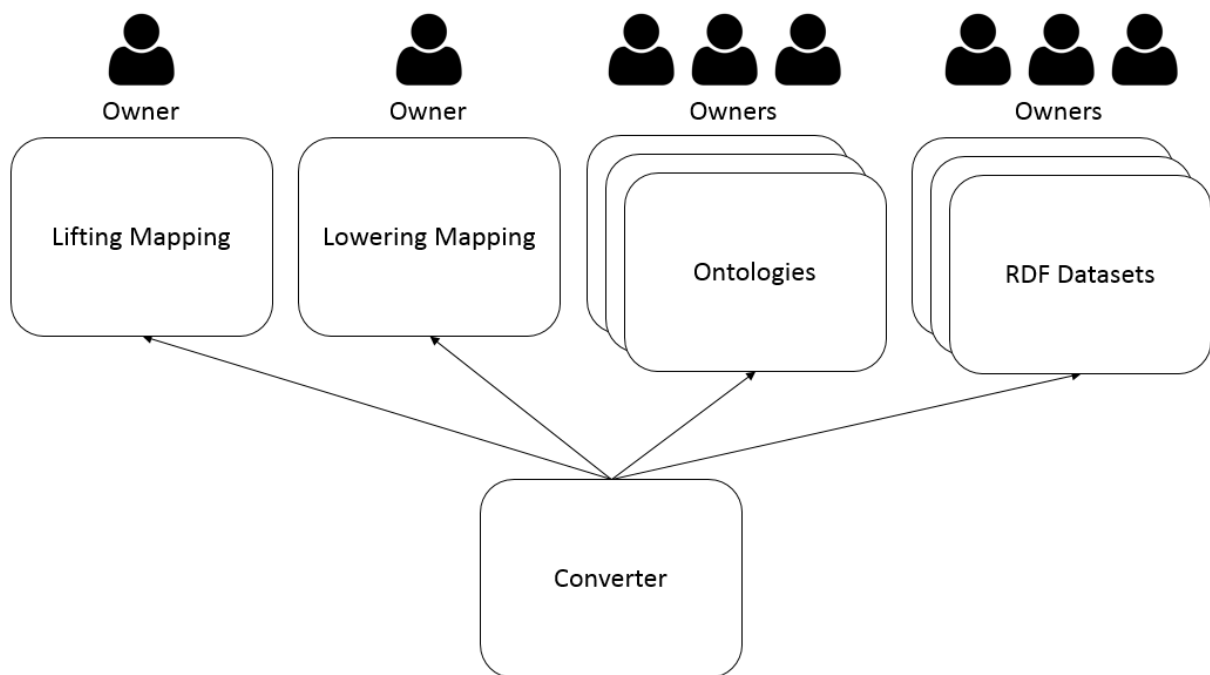


Figure 21 Dependencies between a Converter and other assets

Whenever one of such dependencies is updated, any Converter relying on them potentially should be updated to include the changes. We therefore modified the default BPMN lifecycle management process, introducing, as a last step after a successful publication of an asset, a call to an API which sends notifications to all the owners of assets depending on the current one.

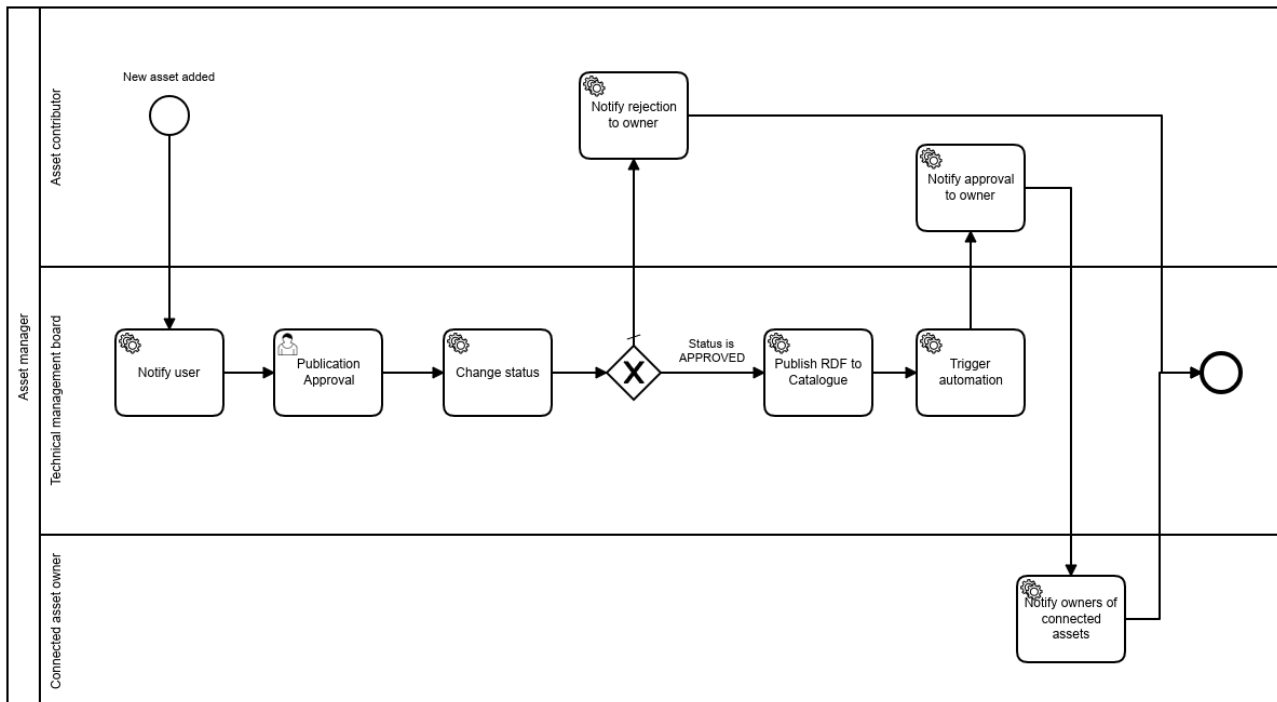


Figure 22 Default lifecycle management process with dependency tracking

To allow validation of the scenario, we created a “Linked GTFS converter” asset in the Asset Manager. This asset requires a “GTFS to Linked GTFS” mapping, which in turn contains an RML mapping file.

2.5.2 Validation

The automatic generation of the Converter artifact has already been demonstrated in the context of C-Rel and documented in D5.3. We will now focus on F-Rel improvements.

The automatic generation of the Converter artifact has already been demonstrated in the context of C-Rel and documented in D5.3. We will now focus on F-Rel improvements.

After successfully publishing the Converter, the Asset Manager triggers the Jenkins pipeline whose responsibility is to generate deployable artifacts, namely:

- a self-contained JAR with no external dependencies ready to be run from the command line;
- a Docker-oriented ZIP file containing:
 - the aforementioned JAR file;
 - a Dockerfile to allow containerization;
 - a Docker compose descriptor to allow manual scaling on a single machine;
- a Kubernetes-oriented ZIP file containing:
 - the aforementioned JAR file;

- a Dockerfile to allow containerization;
- a Kubernetes descriptor to allow running the Converter on a cluster.

The details of the Kubernetes descriptor have already been described in D4.3, and an example configuration is shown in Figure 23. Although the Docker compose and the Kubernetes descriptors can be used as-is to quickly test the Converter behavior, such artifacts are mainly intended to provide a basis for further customization by the end users and developers. The key point is that the SPRINT Converter framework allows developers to adapt an efficient but generic conversion process to a production environment without any need for code modifications. The only effort required will be related to the configuration of the conversion process (which resides in an XML file) and of the “conversion architecture” to meet the scalability requirements.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: chimera-example
  labels:
    app: chimera-converter
    chimera-infra: chimera-example-deployment
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: chimera-converter
        chimera-infra: chimera-example-pod
    spec:
      containers:
        - image: repository/chimera-example
          name: chimera-example
          ports:
            - containerPort: 8888
          resources:
            requests:
              memory: "200Mi"
              cpu: "0.2"
            limits:
              memory: "2Gi"
              cpu: "4"
```

Figure 23 Kubernetes deployment descriptor for a Chimera converter created by the Asset Manager

To validate the dependency tracking mechanism, we then modified the “GTFS to Linked GTFS” asset which is required by the “Linked GTFS converter” and approved the publication request. This action triggered the “Notify owners of connected assets” task in the BPMN process activated by the “GTFS to Linked GTFS” publication. This task performs an API call to the Asset Manager, which in turn performs a SPARQL query that retrieves all the assets depending on the asset which is being

updated, and notifies all their owners about a dependency change, as shown in Figure 24. We decided to provide a lightweight dependency update process which notifies users about a change without automatically triggering any artifact re-generation process, with the intent of ensuring the stability of the ecosystem supported by the Asset Manager. The lifting or lowering mapping contained in a “mapping” asset can cover a specific message in a specification/standard, or a whole specification/standard, and refers to a specific version of an ontology. An update in such kind of asset can be minor or major, and in case of major updates, any asset depending on such mapping is not guaranteed to work properly and must be re-tested before being re-published. Since the Asset Manager can be configured to enforce complex governance rules, we provide this feature as an example of how dependency tracking can be exploited to improve the robustness of an interoperability-oriented ecosystem.

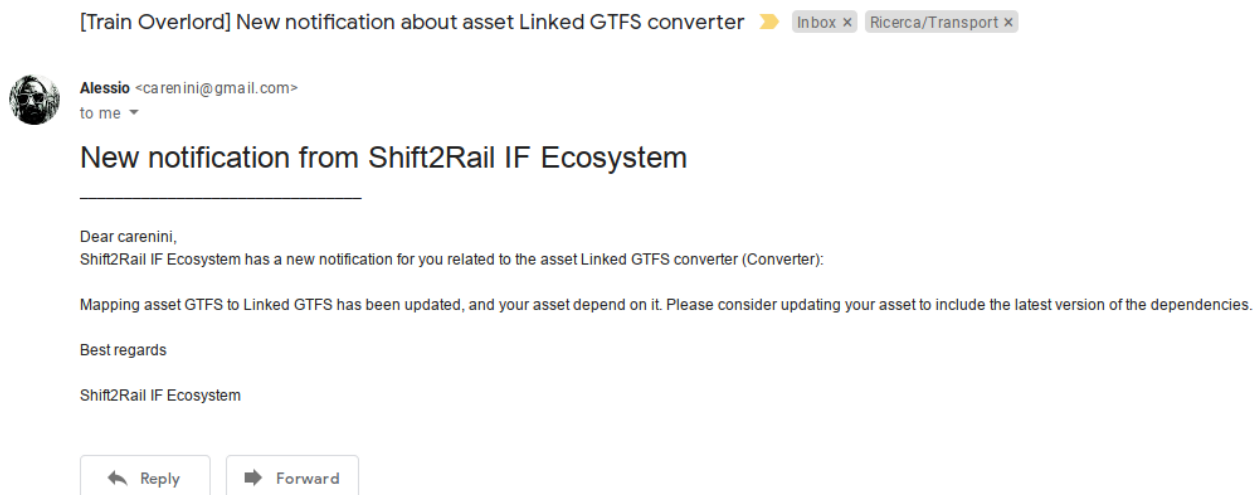


Figure 24 Dependency update notification email sent from the Asset Manager

2.6 SCENARIO S9: FAST ADAPTATION TO PEAKS USE CASE

Table 13 Scenario S9: Fast Adaptation to Peaks Use case

Actor	BE-Service: Booking Engine for land (rail, bus, etc.) travels 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.
Target Component/Sub-system/Entity	Converter, Asset Manager
Description	The infrastructure managing the converters deployed by BE-Service to interact with its partner operators need to dynamically adapt to the load. BE-Service needs to quickly replicate Converters, possibly in a cloud environment, to adapt the infrastructure and avoid denial of service.
Story	One of the cities covered by BE-Service is hosting a huge music event, and BE-Service expects a surge of booking request. BE-Service, therefore, needs to cope with two different scenarios: prepare for the first wave of requests to reach the city, and then to cope with mass requests to reach the music event before its start and to reach the homes and hotels after its end.

2.6.1 Tools configuration

BE-Service is hosting a Converter to let other companies access its booking service. The BE-Service Converter can be built using the Chimera framework proposed for the IF Converter and implemented as a self-contained component with minimal dependencies. Once configured the specific pipeline required by the BE-Service, the Asset Manager can generate different artifacts to guarantee the management of multiple replicas and the dynamic adaptation to load peaks.

The Chimera framework allows configuring the BE-Service Converter using Apache Camel³ and Spring⁴ to implement a stateless service exposing an endpoint handling conversion requests. To guarantee the portability in different deployment environments, the Asset Manager generates a Dockerfile to build a Docker image for the Converter service and to allow its execution as a container in a Container Runtime environment. Moreover, it provides artifacts allowing to scale the deployment of the Converter service. Two different configurations with Docker Compose and Kubernetes manifests are generated, allowing to instantiate multiple replicas of the Converter and to load balance the requests to the BE-Service Converter among them. Details on the validation of the

³ <https://camel.apache.org/>

⁴ <https://spring.io/>

Docker Compose approach for scalability can be found in the deliverable D5.3. Considering a Kubernetes environment, the F-Rel validation focuses on the scenario described.

2.6.2 Validation

The Github repository of the Chimera framework (<https://github.com/cefriel/chimera>) contains a complete example (<https://github.com/cefriel/chimera/tree/master/chimera-example>) that can be used to validate the proposed scenario. The repository also provides a complete description of the Docker Compose approach already validated for CREL in D5.3. In this section, assuming the *chimera-example* project has been configured with a conversion pipeline for the BE-Service, it is showcased how to dynamically adapt the Converter service in a Kubernetes environment. The validation of the scenario uses the Kubernetes package available in the repository and automatically generated by the Asset Manager for each Converter. The Kubernetes package contains the JAR file of the Converter service, the Dockerfile to build the associated Docker image and the Kubernetes configuration files (manifests). Further explanations on the approach proposed in FREL can be found in deliverable D4.3. For the proposed validation we consider a local Kubernetes environment with a single node using *minikube*⁵.

The provided Kubernetes manifests assume that a Docker image of the Converter is available in a local/remote Container Registry configured to be accessed by Kubernetes. A different Docker image tag and additional configurations can be set modifying the YAML file provided (e.g., exposed port, resources needed/limits, labels, etc.).

The provided manifest creates a Kubernetes *Deployment* using the converter image for instantiating a *Pod* and defining a related Kubernetes *Service*. We can deploy the BE-Service converter as a *Service* using the following command.

```
$ kubectl apply -f chimera-converter.yml
```

Once executed the command, it is possible to run `kubectl get pods` to check if the BE-Service converter is running in the cluster, and `kubectl get services` to visualize the related *Service*.

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
chimera-example-c6b446c8-7mbg6	1/1	Running	0	33m

```
$ kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
chimera-example	NodePort	10.101.225.118	<none>	8888:30042/TCP	33m

Depending on the cluster management, different configurations for a *Service* can then be defined to publish the defined Converter *Service* on an external IP⁶. In the example considered, the converter

⁵ <https://minikube.sigs.k8s.io/docs/>

⁶ <https://kubernetes.io/docs/concepts/services-networking/service/#publishing-services-service-types>

Service is exposed on port 30042 of each node of the cluster and it forwards requests to the *Pods* associated with the *Service* on their 8888 port.

If the BE-Service expects a load peak at a given hour, it can manually scale the *Service* using the Kubernetes API. The requests sent to the Converter *Service* will be distributed among the different *Pods* (replicas) instantiated for the *Service*. The BE-Service can also set a lower number of replicas once the expected load peak is passed.

```
$ kubectl scale --replicas=3 deployments/chimera-example
```

```
deployment.extensions/chimera-example scaled
```

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
chimera-example-c6b446c8-gb9rs	0/1	ContainerCreating	0	2s
chimera-example-c6b446c8-nnvwc	0/1	ContainerCreating	0	2s
chimera-example-c6b446c8-whnp9	1/1	Running	0	50m

Scripting solutions can be implemented to automate the scaling of a *Deployment* at a certain time of the day, and/or during days when a load peak is expected. However, to be more proactive in the adaptation to unexpected load peaks, the BE-Service can define a *Horizontal Pod Autoscaler*⁷ (HPA) within the Kubernetes cluster targeting the Converter. The HPA enables automatic scaling of replicas for a *Deployment* considering metrics exposed by the *metrics-server*, that needs to be configured in the cluster to expose metrics via the *resource metrics API* to the HPA (instructions for deploying it are on the GitHub repository of *metrics-server*⁸). The following command enables autoscaling from 1 up to 5 replicas if CPU usage is greater than 80 percent considering the available replicas and the *request* for CPU in the *Deployment* manifest⁹.

```
$ kubectl autoscale deployments/chimera-example --min=1 --max=5 --cpu-percent=80
```

To check the HPA status it is possible to run the commands `kubectl get hpa`.

```
$ kubectl get hpa
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS
chimera-example	Deployment/chimera-example	9%/80%	1	5	3

Different metrics exposed by the *metrics-server* can be used to set the scaling logic. Moreover, an HPA can be also configured using a YAML manifests to set additional configurations¹⁰. Using the command `kubectl describe hpa chimera-example` we can obtain information on the HPA

⁷ <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>

⁸ <https://github.com/kubernetes-incubator/metrics-server/>

⁹ More details on the adopted formula can be found in the HPA documentation

¹⁰ <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>

and related events. For example, considering a load test performed on the BE-Service Converter we can see that a request to scale the service is automatically instantiated.

```
$ kubectl describe hpa chimera-example
```

```
Name: chimera-example
Namespace: default
Reference: Deployment/chimera-example
Metrics:
  resource cpu on pods (as a percentage of request): 316% (632m) / 80%
Min replicas: 1
Max replicas: 5
Deployment pods: 1 current / 4 desired

Conditions:
  Type            Status Reason                               Message
  ----            -
  AbleToScale     True  SucceededRescale                       the HPA controller was able to update the
target scale to 4
  ScalingActive   True  ValidMetricFound                       the HPA was able to successfully calculate
a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  False DesiredWithinRange                     the desired count is within the acceptable
range

Events:
  Type    Reason             Age   From                                Message
  ----    -
  Normal  SuccessfulRescale  3s    horizontal-pod-autoscaler          New size: 4; reason: cpu
resource utilization (percentage of request) above target
```

Once the load peak is passed and the CPU utilization starts lowering, the HPA automatically rescales the number of replicas.

```
$ kubectl describe hpa chimera-example | grep Rescale
```

```
Normal SuccessfulRescale 4m    horizontal-pod-autoscaler New size: 4; reason: cpu
resource utilization (percentage of request) above target
Normal SuccessfulRescale 1m    horizontal-pod-autoscaler New size: 1; reason: All
metrics below target
```

The automatic scaling of the number of replicas can be also defined using application-specific metrics (custom metrics) but needs an "adapter" API server provided by a metrics solution vendor¹¹. Therefore, the configuration depends on the metrics pipeline implemented in the cluster. To enable this type of configuration, the validation is completed explaining how to expose custom metrics in a Converter implemented with Chimera.

The *chimera-example* provides an alternative Camel Context (*src/main/resources/routes/camel-context-monitor.xml*) showcasing how to use the *camel-micrometer*¹² component to expose custom metrics as an endpoint compliant with the well-established Prometheus¹³ format. The Micrometer registry configuring the endpoint can be modified in the Java class *MicrometerConfig*.

¹¹ <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#support-for-metrics-apis>

¹² <https://camel.apache.org/components/latest/micrometer-component.html>

¹³ <https://prometheus.io/>

The endpoint exposes default metrics about the *JVM* and on *Camel Routes* and *Messages*. Moreover, the Camel Context is configured to showcase how to expose some custom metrics: the number of conversions executed (num_executions as a *counter*), conversion time (processing_time as a *timer*), metrics on the *seda* queues¹⁴ enabled for the endpoint (seda_queue_size *summary*).

The endpoint is made available at <http://localhost:8888/chimera-demo/metrics> and can be easily configured using an agent for scraping. Once available in a time-series database (e.g., *Prometheus*), these metrics can be queried and visualized through a dashboard (e.g., using *Grafana*¹⁵) to drive scaling decisions, and/or accessed to implement more tailored automatisms for the scaling based on the actual deployment environment (e.g. using the *Prometheus adapter*¹⁶ to feed *Kubernetes* metrics).

2.7 SCENARIO S10: SPECIAL PURPOSE ASSET DISCOVERY PACKAGE: RESOLVER

Table 14 Scenario S10: Special Purpose Asset Discovery Package: Resolver

Actor	Travel Service Provider and/or Public Authority
Target Component/Sub-system/Entity	Resolver, Asset Discovery, Registry, Converter
Description	<p>Special-purpose Asset Discovery components, or Resolvers, are packaged as deployable units and used to perform discovery/retrieve of specific categories of resources such a Locations or Travel Expert services. These Resolvers can be deployed equally internally to the Interoperability Framework, or in any external runtime environment, e.g. at the Travel Service Provider.</p> <p>In this scenario, Resolvers are deployed externally to the Interoperability Framework and are used by a Travel Service Provider or Public Authority in the execution of a shopping/booking or trip tracking process, where there is a need to identify and access resources that are unknown to the requesting application at runtime. These resources may be the geographical coordinates of some Point Of Interest, the Stop Places closest to these geographical coordinates, the web service interface specification of a remote system that can compute offers for an itinerary starting and ending at specified Stop Places, or of a remote system that can perform bookings for an offer.</p>

¹⁴ <https://camel.apache.org/components/latest/seda-component.html>

¹⁵ <https://grafana.com/>

¹⁶ <https://github.com/directxman12/k8s-prometheus-adapter>

Story	<p>Requesting Actor, e.g. Travel Service Provider application needs access to resources that may be distributed over the network and unknown to it at runtime. To locate and get access to these resources:</p> <ol style="list-style-type: none">1. The requesting Actor calls the service interface of special-purpose Resolver component with a specific query2. The Resolver validates and analyses the query3. If the query is valid, it is passed to the Asset Discovery component for processing. The Process Request activity may use the Distributed SPARQL endpoint to access assets semantic annotations meta-data to determine the nature of the assets being requested<ol style="list-style-type: none">a. If data assets are being requested, such as Stop Places or geographical coordinates, the Asset Discovery initiates the Retrieve Data Assets activity in the Registry through a call to the registry's interface.b. If a web service interface is being requested, such as a Travel Expert or Booking Engine, the Asset Discovery initiates the Retrieve Service Descriptor activity in the Registry through a call to the registry's interface.4. Data Assets or Service Descriptors obtained in steps 3a or 3b, respectively, are associated with their format specification. This specification is used to call a dynamic Converter, which will locate the relevant ontologies, datasets and mappings inside the Asset Manager, in the case where this is needed by the requestor Actor to map the data asset or service descriptor to a target different specification5. The Resolver then builds a response to be returned to the requestor Actor. The response contains the requested data asset or the service descriptor, depending on the specific request, and the electronic link to the associated converter where applicable. This link may be used by the requestor to access the converter to be used with the returned data asset or service descriptor6. The response thus build is returned to the requestor Actor
-------	---

2.7.1 Tools configuration

This scenario demonstrates the possibility to use the Asset Manager in the runtime phase of the conversion process with the aid of a Converter Resolver. In particular, it shows how the SPRINT Converter framework can be configured in order to become a “universal converter”, which is able to convert a potentially unlimited number of message types by relying on mappings, ontologies and datasets obtained by invoking a Converter Resolver that in turns queries the Asset Manager.

This conversion architecture was described in D4.3 in the “Asset Manager-Converter runtime integration”, and it relies on the assumption that a message can be converted by pre-loading any master data datasets and ontologies, and by then applying exactly one lifting mapping and one

lowering mapping. Among the different alternatives discussed in D4.3, for the validation of the scenario, we implemented the set of interactions shown in Figure 25.

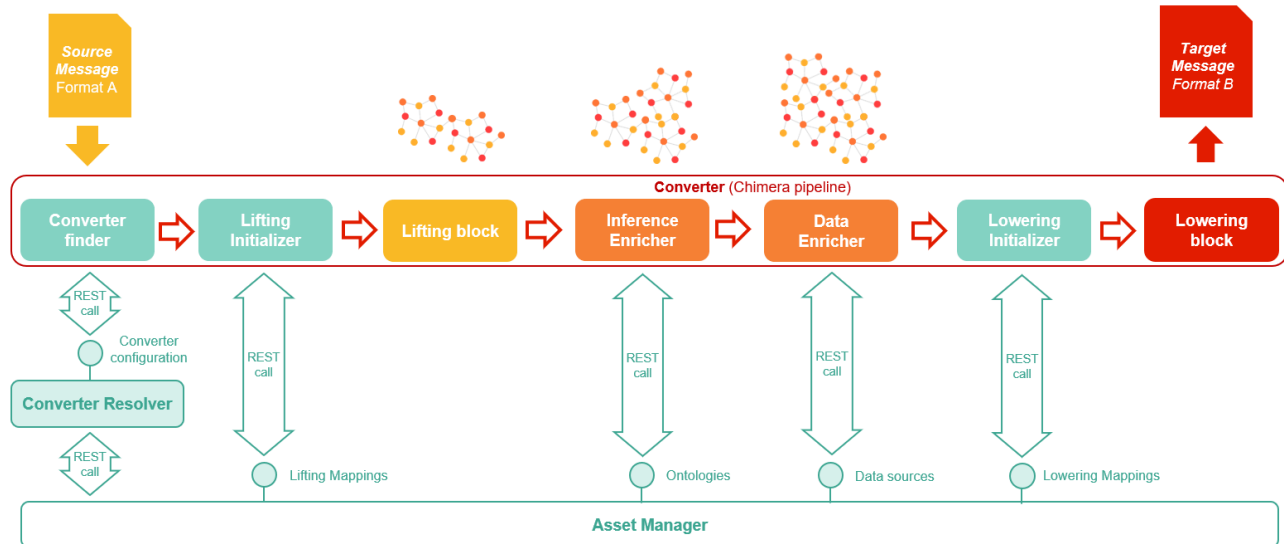


Figure 25 The Converter Resolver enables a “universal converter” to access needed resources from the Asset Manager at runtime.

The integration between the Asset Manager and a Converter is made possible by three “components”: (i) an *API* exposed from the *Asset Manager* to obtain all the resources required to perform a conversion between a source and a destination message type, (ii) a *Converter Resolver* capable of invoking the mentioned API and process it to produce a converter configuration, (ii) a *Converter* capable of processing an external configuration accessing resources needed on-the-fly.

Asset Manager

Since dependencies between assets are represented in RDF, the Exploration API mechanism provided a perfect solution to implement the Asset Manager API. We, therefore, implemented an Exploration API named `converter_dependencies_resources` (whose Swagger documentation is shown in Figure 26), based on the SPARQL query listed in Figure 27. The output of the SPARQL CONSTRUCT query is then returned to the requester after applying the JSON-LD Frame shown in Figure 28. An example output is shown in Figure 29.

GET /assets-api/exploration_api/converter_dependencies_resources/execute

Parameters Try it out

Name	Description
target string (query)	<input type="text" value="target"/>
converter_name string (query)	<input type="text" value="converter_name"/>
source string (query)	<input type="text" value="source"/>

Figure 26 Converter dependencies resources API exposed by the Asset Manager via the Exploration API

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dcat: <https://www.w3.org/ns/dcat#>
PREFIX cef: <http://www.cefriel.com/knowledge_tech/ontologies/asset_manager#>

CONSTRUCT {
  ?s dct:title ?title_str;
  rdf:type cef:Converter;
  cef:requiresOntology ?local_ontology_url;
  cef:requiresDataset ?local_dataset_url;
  cef:requiresMapping {
    cef:mappingSerialisation ?format;
    cef:mappingDirection ?direction;
    dcat:downloadURL ?local_mapping_url;
  };
  cef:requiresOntology ?remote_ontology_url;
  cef:requiresDataset ?remote_dataset_url;
  cef:requiresMapping {
    dcat:downloadURL ?remote_mapping_url;
  };
}
WHERE {
  ?s dct:type cef:Converter;
  dct:title ?converter_name;
  cef:conversionSource ?source;
  cef:conversionTarget ?target.

  OPTIONAL { ?s cef:requiresLocalMapping ?mapping.
    ?m dct:type cef:Mapping;
    dct:title ?mapping;
    cef:mappingSerialisation ?mapping_tech;
    cef:mappingDirection ?mapping_direction;
    dcat:distribution {
      dcat:downloadURL ?local_mapping
    }.
    BIND(STR(?local_mapping) as ?local_mapping_url)
    BIND(strafter(STR(?mapping_direction), "#") as ?direction)
    BIND(STR(?mapping_serialisation) as ?format)
  }
  OPTIONAL { ?s cef:requiresLocalOntology ?ontology.
    ?o dct:type cef:Ontology;
    dct:title ?ontology;
    dcat:distribution {
      dcat:downloadURL ?local_ontology
    }.
    BIND(STR(?local_ontology) as ?local_ontology_url)
  }
  OPTIONAL { ?s cef:requiresLocalDataset ?dataset .
    ?d dct:type cef:Dataset;
    dct:title ?dataset;
    dcat:distribution {
      dcat:downloadURL ?local_dataset
    }.
    BIND(STR(?local_dataset) as ?local_dataset_url)
  }
  OPTIONAL {
    ?s cef:requiresRemoteMapping ?remote_mapping
    BIND(STR(?remote_mapping) as ?remote_mapping_url)
  }
  OPTIONAL {
    ?s cef:requiresRemoteOntology ?remote_ontology
    BIND(STR(?remote_ontology) as ?remote_ontology_url)
  }
  OPTIONAL {
    ?s cef:requiresRemoteDataset ?remote_dataset
    BIND(STR(?remote_dataset) as ?remote_dataset_url)
  }
  values { ?converter_name } { ( { {converter_name or 'UNDEF' } } ) }
  values { ?source } { ( { {source or 'UNDEF' } } ) }
  values { ?target } { ( { {target or 'UNDEF' } } ) }
  BIND(STR(?converter_name) as ?title_str)
}

```

Figure 27 SPARQL query to retrieve all the resources required to run a specific conversion

Converter Resolver

A Converter Resolver can be easily implemented as a service using Apache Camel to expose an API and process the requests. Once invoked, the resolver accesses the Exploration API exposed by the Asset Manager and retrieves the available resources to perform a conversion between the source and the destination message type as requested by the caller. The Converter Resolver needs (i) a valid token provided by the requester to dialogue with the Asset Manager, and (ii) the required information to compose the request to the Asset Manager (source, target and, optionally, the converter name). In the implemented solution, the Converter Resolver expects these data to be

contained in the headers of the incoming request. The Converter Resolver may adopt cache mechanisms to avoid invoking the *Asset Manager* for every request and to improve performances.

The Converter Resolver invokes the Asset Manager and processes the obtained JSON-LD fragment querying the resulting RDF. It extracts the required information to produce a converter configuration processable by the SPRINT Converter¹⁷ and serializes it in the response as JSON.

```
{
  "@context": {
    "cef": "http://www.cefriel.com/knowledge_tech/ontologies/asset_manager#",
    "dct": "http://purl.org/dc/terms/",
    "dcat": "https://www.w3.org/ns/dcat#",
    "requiresMapping": { "@id": "cef:requiresMapping", "@container": "@set" },
    "requiresOntology": { "@id": "cef:requiresOntology", "@container": "@set" },
    "requiresDataset": { "@id": "cef:requiresDataset", "@container": "@set" },
    "mappingDirection": "cef:mappingDirection",
    "downloadURL": "dcat:downloadURL",
    "title": "dct:title"
  },
  "title": {},
  "@embed": {}
}
```

Figure 28 JSON-LD Frame applied to the dependency tracking SPARQL query

```
{
  "@context": {
    "cef": "http://www.cefriel.com/knowledge_tech/ontologies/asset_manager#",
    "dct": "http://purl.org/dc/terms/",
    "dcat": "https://www.w3.org/ns/dcat#",
    "requiresMapping": { "@id": "cef:requiresMapping", "@container": "@set" },
    "requiresOntology": { "@id": "cef:requiresOntology", "@container": "@set" },
    "requiresDataset": { "@id": "cef:requiresDataset", "@container": "@set" },
    "mappingDirection": "cef:mappingDirection",
    "downloadURL": "dcat:downloadURL",
    "title": "dct:title"
  },
  "@graph": [
    {
      "@id": "http://www.shift2rail.eu/instances#http-localhost-8000-publisher-assets-converter-vbb-to-trias-converter",
      "@type": "cef:Converter",
      "title": "VBB to TRIAS converter",
      "requiresMapping": [
        {
          "mappingDirection": "lifting",
          "downloadURL": "http://localhost:8000/publisher/media/assets_media/mapping/VBB-to-IT2Rail/vbb_it2rail.rml.ttl"
        },
        {
          "mappingDirection": "lowering",
          "downloadURL": "http://localhost:8000/publisher/media/assets_media/mapping/IT2Rail-to-TRIAS/it2rail_trias.vm"
        }
      ]
    }
  ]
}
```

Figure 29 Converter dependencies resources API result example

¹⁷<https://github.com/cefriel/chimera/blob/master/chimera-core/src/main/java/com/cefriel/chimera/util/ConverterConfiguration.java>

Converter

The Converter defines a generic conversion pipeline adopting the blocks of the SPRINT Chimera framework (cf. D5.5) for lifting (RMLProcessor), data enrichment (DataEnricherProcessor), inference enrichment (InferenceEnricherProcessor) and lowering (TemplateProcessor). The Converter exposes an API to request the conversion indicating the source and target message type (`/converter/convert/<source>/<target>`) and providing the message in the request body. In the pipeline execution, an initial REST call to the Converter Resolver is performed to retrieve the Converter Configuration. The configuration is attached to the message¹⁸ going through the pipeline and accessed by the various components to process the input data and produce the response. External resources are accessed through an authenticated GET request to the URL provided in the configuration. In the considered case, the Converter should be configured with a username/password to obtain a valid JWT token from the Asset Manager and access the needed resources.

2.7.2 Validation

To validate the described scenario, we defined a Converter asset in the Asset Manager to convert a response message of the HaCon VBB endpoint¹⁹ to TRIAS. The defined Converter depends on a resource containing RML lifting mappings from the VBB format to the IT2Rail ontology, and a resource containing an Apache Velocity template from the IT2Rail ontology to TRIAS. For our validation, we considered a subset of the VBB and TRIAS specification. An example of the input message is shown in

Figure 30, and the related output message in Figure 31.

```
<?xml version="1.0" encoding="UTF-8" ?>
<TripList serverVersion="1.6" dialectVersion="1.0" scrB="1|OB|MTÂµ11Âµ16314Âµ16314Âµ16321Âµ16321Âµ0Â
  <Trip idx="0" tripId="C-0" duration="PT7M">
    <ServiceDays sDaysR="Sa, Su" sDaysI="also 20. Apr until 1. May 2020, 8., 21. May, 1. Jun"
      <LegList> ... </LegList>
      <TariffResult> ... </TariffResult>
    </Trip>
    <Trip idx="1" tripId="C-1" duration="PT7M"> ... </Trip>
    <Trip idx="2" tripId="C-2" duration="PT7M"> ... </Trip>
    <Trip idx="3" tripId="C-3" duration="PT7M"> ... </Trip>
    <Trip idx="4" tripId="C-4" duration="PT7M"> ... </Trip>
  </TripList>
```

Figure 30 Example message obtained from the HaCon VBB endpoint

¹⁸ Enrichment using a custom AggregationStrategy <https://github.com/cefriel/chimera/blob/master/chimera-core/src/main/java/com/cefriel/chimera/processor/aggregate/ConverterAggregationStrategy.java>

¹⁹ <http://fahrinfo.vbb.de>

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<Trias xmlns="http://www.vdv.de/trias" xmlns:ns2="http://www.siri.org.uk/siri" xmlns:ns3="
http://www.ifopt.org.uk/acsb" xmlns:ns4="http://www.ifopt.org.uk/ifopt" xmlns:ns5="
http://datex2.eu/schema/1_0/1_0">
  <ServiceDelivery>
    <DeliveryPayload>
      <TripResponse>
        <TripResponseContext> ... </TripResponseContext>
        <TripResult> ... </TripResult>
        <TripResult> ... </TripResult>
        <TripResult> ... </TripResult>
        <TripResult> ... </TripResult>
        <TripResult> ... </TripResult>
      </TripResponse>
    </DeliveryPayload>
  </ServiceDelivery>
</Trias>
```

Figure 31 Example message in TRIAS obtained through the defined conversion

To validate the scenario, we deployed the Asset Manager, the Converter Resolver, and the “universal” Converter. The following request is performed to the Converter

```
curl -i -X POST -H "Content-type:text/xml" -T "./vbbresponse.xml"
'http://localhost:8888/converter/convert/vbb/trias'
```

As shown in the logs of the Converter and Converter Resolver containers (Figure 32), the “universal” converter: (i) retrieves the VBB-Trias converter configuration from the Converter Resolver, and (ii) applies it to perform the lifting and the lowering processes using the defined blocks and accessing resources from the Asset Manager.

```
chimera-converter | INFO Conversion Started
chimera-converter | INFO Converter Configuration requested
chimera-resolver | INFO Source: vbb
chimera-resolver | INFO Destination: trias
chimera-resolver | INFO Resolving Converter Request: https://localhost:8000/assets-
api/exploration_api/converter_dependencies_resources/execute?source_standard=vbb&destination_standard=trias
chimera-resolver | INFO Configuration of converter 'VBB-TRIAS Converter' retrieved
chimera-converter | INFO Creating In-Memory RDF4J store
chimera-converter | INFO Lifting from vbb
chimera-converter | INFO Converter configuration found in the exchange, lifting mappings extracted
chimera-converter | INFO Loading resource http://localhost:8000/publisher/media/assets_media/mapping/VBB-to-
It2Rail/vbb.ttl
chimera-converter | INFO RML Initializer created
chimera-converter | INFO RML processing completed
chimera-converter | INFO Lifting completed
chimera-converter | INFO Lowering to trias
chimera-converter | INFO Converter configuration found in the exchange, lowering mappings extracted
chimera-converter | INFO Loading resource http://localhost:8000/publisher/media/assets_media/mapping/It2Rail-
to-TRIAS/trias.vm
chimera-converter | INFO Template Lowerer Initializer created
chimera-converter | INFO Lowering completed
```

Figure 32 Logs of the Converter Resolver and the “universal” Converter for a VBB to TRIAS request

2.8 SCENARIO 11: (COLLABORATIVE) ONTOLOGY MANAGER

Table 15 Scenario S11: (Collaborative) Ontology Manager

Actor	EU-Tram: A European TSP, in specific a tram service provider that aims to represent its data semantically.
Target Component/Sub-system/Entity	OnToology
Description	This scenario illustrates validation and documentation during the collaborative ontology development lifecycle
Story	EU-Tram is interested in generating, extending and reviewing a set of ontologies that have developed to semantically represent its data. As part of the feedback process during the ontology development lifecycle, EU-Tram proposes its ontology developers to use OnToology in the generation and revision of its ontologies. To build each ontology, developers must interact with domain experts that are not familiar with ontologies. Thus, these experts need to easily understand vocabulary involved in the transport domain by means of diagrams in order to elicit their requirements. After that, they can provide feedback to the ontology developers which will extend and improve the already created ontologies and re-start the documentation process. In addition, the ontology developers must evaluate the quality of their generated ontologies by means of OnToology, and identify and fix potential errors before publishing the ontology.

2.8.1 Tools configuration

For this scenario, we use the ontology module Facilities, which is defined as part of the transmodel ontology²⁰. Before running the evaluation over an input ontology, we have to configure the Ontology tool, which allows to the users to evaluate the model, generate its documentation or to generate a graphical visualization. In general, this step, gives an overview of the quality of the input ontology. Note that although we are using the Facilities module from Transmodel ontology, these steps can be run over any ontology described in OWL. We use Jenkins as a efficient and maintainable way to integrate all the steps performed by Ontology.

The requirements for configuring the engine are:

- Fork, and clone locally the repository located in: <https://github.com/oeg-upm/sprint>
- Move to the folder *collaborative_ontology_fv*

²⁰ <https://github.com/oeg-upm/transmodel-ontology/blob/master/ontology/tm-facilities.owl>

- Locate the ontology file inside the directory called Ontology. The Jenkins file has to be located in the root folder as it is shown in Figure 33.
- Edit the Jenkins file (see Figure 34), changing the name of the ontology file that wants to be evaluated.
- Push the changes over your repository.

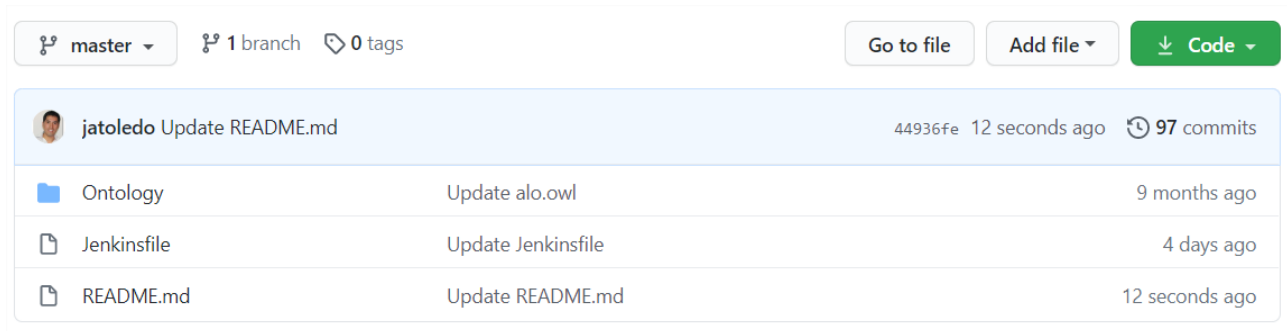


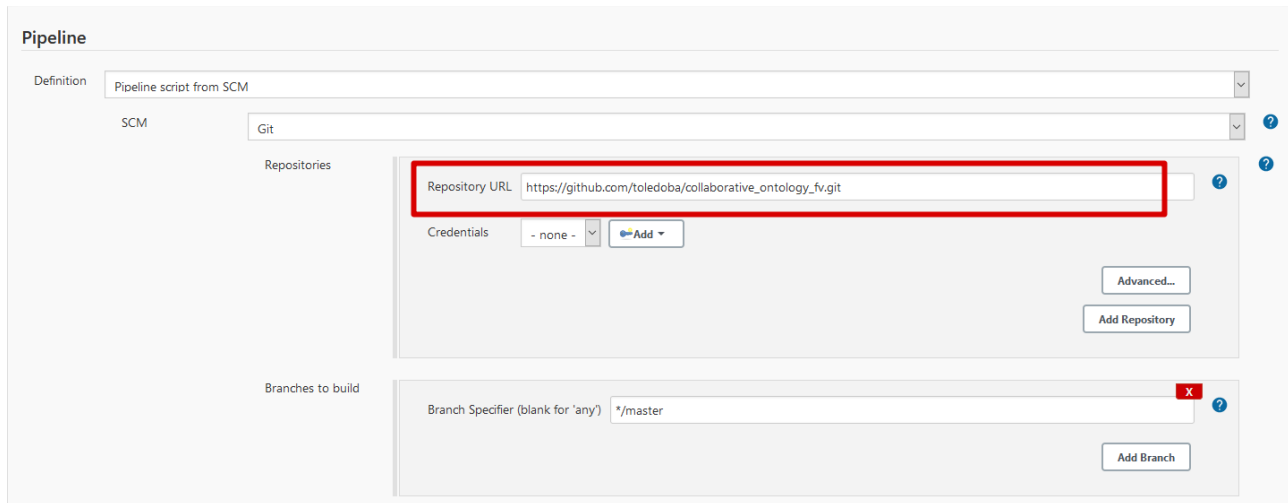
Figure 33 Repository structure



Figure 34 Jenkinsfile environment setup

Finally, for configuring Jenkins:

- Create New Item on your Jenkins home page, enter a name for your (pipeline) job, select Pipeline, and click OK.
- In the pipeline section add your Github repository path and save it (see Figure 35).
- The pipeline can be run using the build tasks of Jenkins



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories:

Repository URL: https://github.com/toledoba/collaborative_ontology_fw.git

Credentials: - none - Add

Advanced...

Add Repository

Branches to build:

Branch Specifier (blank for 'any'): */master

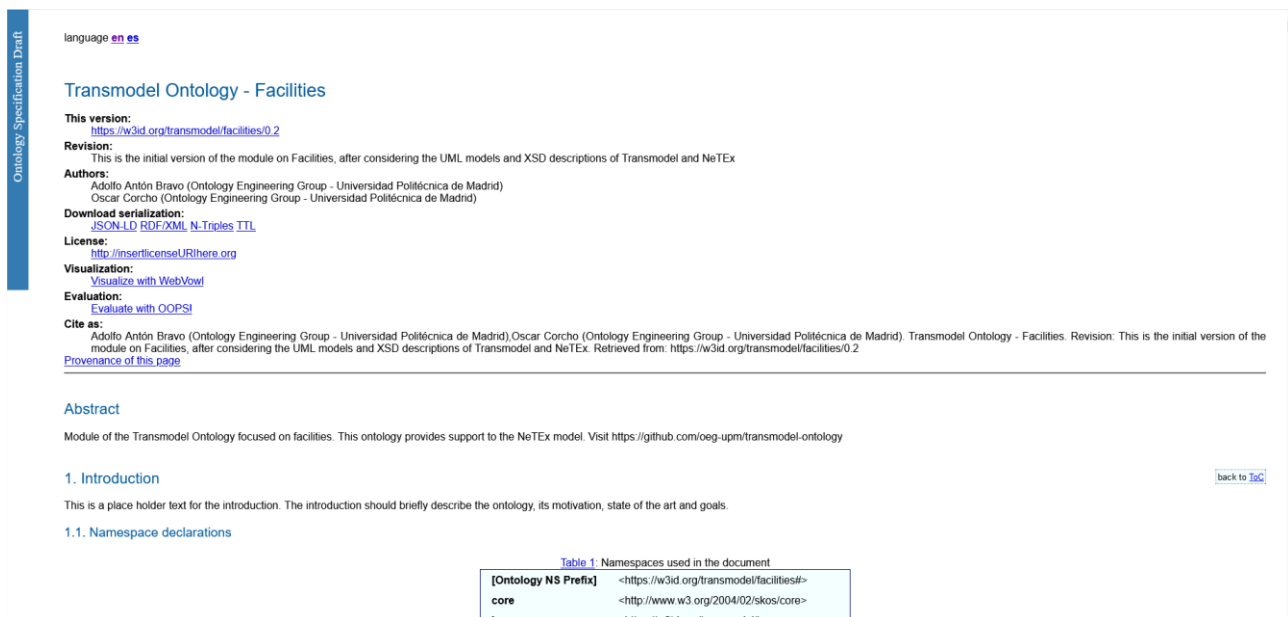
Add Branch

Figure 35 Repository configuration in Jenkins

2.8.2 Validation

In the Jenkins workspace, the HTML files are generated where the documentation, evaluation and landing page can be explored.

As can be seen in Figure 36, the generated documentation allows the user to understand the ontology with the specification of the requirements. Figure 37 shows the evaluation of the ontology and reflect common pitfalls during the development of an ontology. In this example, OOPS! suggests how the ontology elements could be modified to improve the ontology quality. However, not all of the pitfalls identified should be interpreted as factual errors but as suggestions that must be manually revised in some cases. Finally, Figure 38 shows a landing page with the ontologies that exist in our Github repository, in the case there are multiple.



language [en](#) [es](#)

Transmodel Ontology - Facilities

This version:
<https://w3id.org/transmodel/facilities/0.2>

Revision:
This is the initial version of the module on Facilities, after considering the UML models and XSD descriptions of Transmodel and NeTeX

Authors:
Adolfo Antón Bravo (Ontology Engineering Group - Universidad Politécnica de Madrid)
Oscar Corcho (Ontology Engineering Group - Universidad Politécnica de Madrid)

Download serialization:
[JSON-LD](#) [RDF/XML](#) [N-Triples](#) [TTL](#)

License:
<http://insertlicenseURIhere.org>

Visualization:
[Visualize with WebVowl](#)

Evaluation:
[Evaluate with OOPS!](#)

Cite as:
Adolfo Antón Bravo (Ontology Engineering Group - Universidad Politécnica de Madrid), Oscar Corcho (Ontology Engineering Group - Universidad Politécnica de Madrid). Transmodel Ontology - Facilities. Revision: This is the initial version of the module on Facilities, after considering the UML models and XSD descriptions of Transmodel and NeTeX. Retrieved from: <https://w3id.org/transmodel/facilities/0.2>

[Provenance of this page](#)

Abstract

Module of the Transmodel Ontology focused on facilities. This ontology provides support to the NeTeX model. Visit <https://github.com/oeg-upm/transmodel-ontology>

1. Introduction

This is a place holder text for the introduction. The introduction should briefly describe the ontology, its motivation, state of the art and goals.

1.1. Namespace declarations

Table 1: Namespaces used in the document

[Ontology NS Prefix]	<https://w3id.org/transmodel/facilities#>
core	<http://www.w3.org/2004/02/skos/core>
inurnavc	<https://w3id.org/transmodel/inurnavc>

Figure 36 Documentation generated by widoco

Transmodel Ontology - Facilities

Title Transmodel Ontology - Facilities
URI <https://w3id.org/transmodel/facilities#>
Version This is the initial version of the module on Facilities, after considering the UML models and XSD descriptions of Transmodel and NeTeX

The following evaluation results have been generated by the [RESTful web service](#) provided by [OOPS! \(Ontology Pitfall Scanner!\)](#).

[OOPS! logot](#) it is obvious that not all the pitfalls are equally important; their impact in the ontology will depend on multiple factors. For this reason, each pitfall has an importance level attached indicating how important it is. We have identified three levels:

- Critical** It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- Important** Though not critical for ontology function, it is important to correct this type of pitfall.
- Minor** It is not really a problem, but by correcting it we will make the ontology nicer.

Evaluation results

P02. Creating synonyms as classes1 case detected. **Minor**

P04. Creating unconnected ontology elements3 cases detected. **Minor**

P08. Missing annotations59 cases detected. **Minor**

P11. Missing domain or range in properties35 cases detected. **Important**

P13. Inverse relationships not explicitly declared58 cases detected. **Minor**

P21. Using a miscellaneous class1 case detected. **Minor**

P22. Using different naming conventions in the ontology ontology * **Minor**

P24. Using recursive definitions2 cases detected. **Important**

P31. Defining wrong equivalent classes1 case detected. **Critical**

P32. Several classes with the same label1 case detected. **Minor**

References:

- [1] Gómez-Pérez, A. Ontology Evaluation. Handbook on Ontologies. S. Staab and R. Studer Editors. Springer. International Handbooks on Information Systems. Pp: 251-274. 2004.
- [2] Noy, N.F., McGuinness, D. L. Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics. 2001.
- [3] Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R.; Wang, H., Wroe, C. "Owl pizzas: Practical experience of teaching owl-dl: Common errors and common patterns". In Proc. of EKAU 2004, pp: 63-81. Springer. 2004.
- [4] Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A. Weaving the Pedantic Web. Linked Data on the Web Workshop LDOW2010 at WWW2010 (2010).

Figure 37 Evaluation generated by Oops!

Vocabularies Vocabulary report

Repository landing page

Here you can find the list of vocabularies that have been found on Repository.

Ontology	Serialization	License	Language	Description
Transmodel Ontology - Facilities	RDF/XML	Attribution 4.0 International...	en es	Module of the Transmodel Ontology focused on facilities. This ontology provides support to the NeTeX model. Visit https://github.com/oeg-upm/transmodel-ontology
Vocabulario para la representación de datos sobre alojamiento	RDF/XML	CC-BY	es	

Page created with [VocabLite](#) (Ontology Engineering Group)
 Built with [Bootstrap](#)
 Latest revision November, 2020




Figure 38 Documentation generated by vocablite

2.9 SCENARIO 12: ONTOLOGY CREATION USING NON-ONTOLOGICAL SOURCES

Table 16 Scenario S12: Ontology Creation using non-ontological sources

Actor	EU-Bus: A European TSP, in specific a bus service provider
Target Component/Sub-system/Entity	XSD2Ontology
Description	This scenario illustrates an ontology creation using non-ontological sources
Story	EU-Bus needs to register itself as a company with the IF node but its information follows the XML format designed for the efficient, updateable exchange of complex transport data among distributed systems. For not starting from scratch, EU-Bus proposed to transform its XML documents to ontologies by means of automatization of the ontology creation. The Ontology generation is done using XML instances and validating XSD files. The generated ontology can be added to our transport ecosystem.

2.9.1 Tools configuration

To exemplify the configuration of the tool, we are supported by the automatic transformation from NeTEx XSD²¹ files to RDF/OWL. We have selected NeTEx because it is an standard for the exchange of timetable data related to public transport. Notice that for this case, the application needs maven to be installed in the computer, which is a software tool for the management and construction of Java projects. Using maven, as it is detailed in the tool's repository²², the installation generates a Java executable file that will be used to generate the ontology from the NeTEx file. Specifically, the requirements are:

- Clone or download the repo: <https://github.com/oeg-upm/sprint>
- Go to the folder `xsd2owl`
- Run `mvn clean install` and copy from the target folder the `xsd2owl-1.0.0.jar` to the root one
- Use the jar executable file using the options defined in Figure 39

²¹ https://github.com/NeTEx-CEN/NeTEx/tree/master/xsd/netex_framework/netex_reusableComponents

²² <https://github.com/oeg-upm/sprint/tree/main/xsd2owl>

Usage

CLI

The following options are the most important.

- `-x, --xsd <arg>` : path to XML Schema file
- `-o, --outputfile <arg>` : path to output file in OWL format

All options can be found when executing `java -jar xsd2owl-1.0.0.jar --help`, that output is found below.

```
usage: java -jar xsd2owl-1.0.0.jar <options>
options:
-h,--help                show help info
-v,--verbose              show more details in debugging output
-o,--outputfile <arg>    path to output file (default: output.rdf)
-x,--xsd <arg>           path to XML Schema file
```

Figure 39 Usage XSD2OWL

2.9.2 Validation

For our example defined in the previous section, the transformation of the XML Schema into OWL files has been tested and the results can be checked in the repository directory called `NeTEx_output`²³. For the particular case of `netex_facility_support.xsd`, in Figure 40 we show the result obtained. These results allow domain experts and ontology developers to see the ontology extracted from the XSD files.

We also show the differences between an automatically created ontology (Figure 41-1) from the XSD files and an ontology created manually (Figure 41-2). We have created the documentation to visualize better the ontology. We can observe that the automatic generation has "Error" classes while the manual model has very well defined classes. This is why the creation of ontologies from non-ontological sources could help a lot in the first steps to have an overview of the domain but the manual work of is essential to create robust and representative models

²³ https://github.com/oeg-upm/sprint/tree/main/xsd2owl/NeTEx_output

Contract No. H2020 – 826172

```

1 <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:owl="http://www.w3.org/2002/07/owl#"
4   xmlns:dtype="http://www.srdc.com.tr/ontalmizer#"
5   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6   xmlns:j.0="http://www.netex.org.uk/netex#"
7   xmlns="http://www.w3.org/2001/XMLSchema" >
8   <rdf:Description rdf:nodeID="A0">
9     <rdf:rest rdf:nodeID="A1"/>
10    <rdf:first rdf:resource="http://www.netex.org.uk/netex#MedicalFacilityEnumeration_unknown"/>
11  </rdf:Description>
12  <rdf:Description rdf:about="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_Enumeration">
13    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_internet"/>
14    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_audioEntertainment"/>
15    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_businessServices"/>
16    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_publicWifi"/>
17    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_freeWifi"/>
18    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_unknown"/>
19    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_postBox"/>
20    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_videoEntertainment"/>
21    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_telephone"/>
22    <rdf:type rdf:resource="http://www.srdc.com.tr/ontalmizer#Enumeration"/>
23    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_postOffice"/>
24    <dtype:hasValue rdf:resource="http://www.netex.org.uk/netex#PassengerCommsFacilityEnumeration_powerSupplySockets"/>
25  </rdf:Description>
26  <rdf:Description rdf:nodeID="A2">
27    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
28    <rdf:first rdf:resource="http://www.netex.org.uk/netex#HireFacilityEnumeration_recreationDeviceHire"/>
29  </rdf:Description>
30  <rdf:Description rdf:nodeID="A3">
31    <rdf:rest rdf:nodeID="A4"/>
32    <rdf:first rdf:resource="http://www.netex.org.uk/netex#PassengerInformationFacilityEnumeration_realTimeConnections"/>

```

Figure 40 Result netex_facility_support.xsd to OWL file

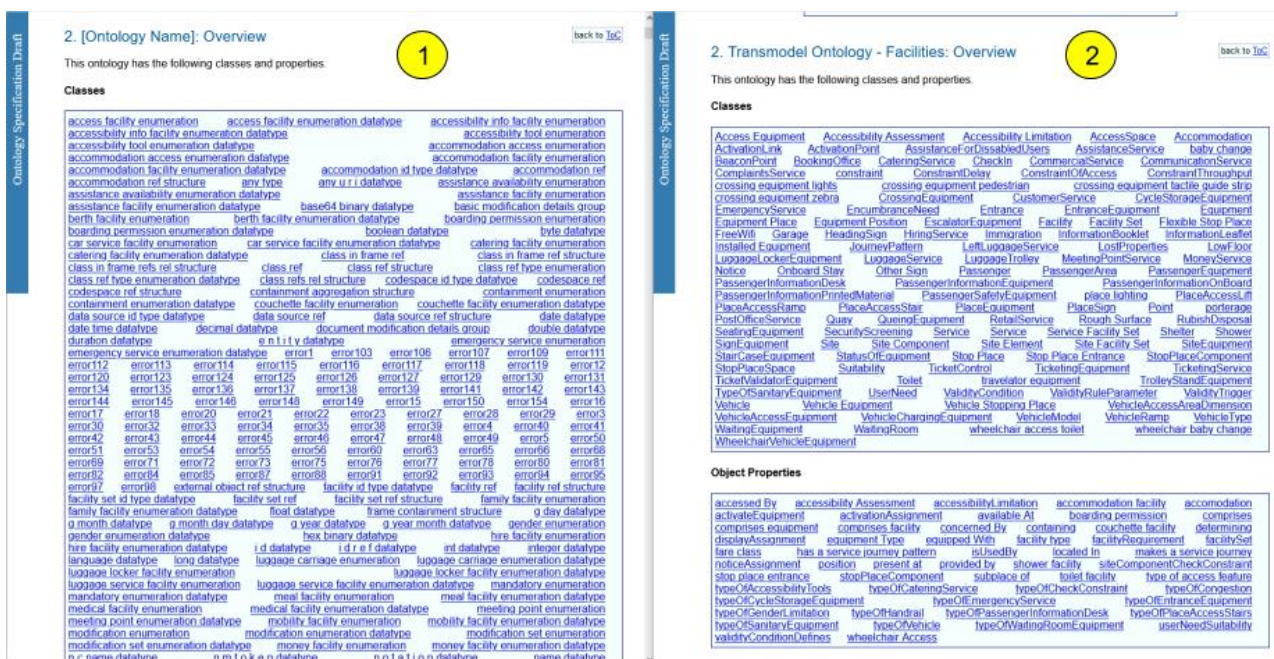


Figure 41 Differences between an automatically created and an ontology created manually

2.10 SCENARIO S13: ASSET MANAGER AS NATIONAL ACCESS POINTS AGGREGATOR

Table 17 Scenario S13 Asset Manager as National Access Points aggregator

Actor	EuroTrain, a rail TSP focused on cross-border travels.
Target Component/Sub-system/Entity	Asset Manager, Converter
Description	This scenario demonstrates the usage of the Asset Manager as an aggregator of National Access Points.
Story	EuroTrain is planning a new cross-border service connecting France and Netherlands through Belgium. They need to access the Journey planning data to analyse the competitors' offers and create their offer. Since their planned service crosses three EU countries, they would need to access three different National Access Points. Instead, they deploy and configure an IF Asset Manager to let it aggregate the metadata coming from those countries, and get access to the desired assets using a single interface.

2.10.1 Tools configuration

To validate S13 we added a “NAP metadata ingester” service in the default deployment of the Asset Manager. The ingestion service is triggered by the Asset Manager every three hours, and it performs the following set of operations on the National Access Points provided by France, Belgium and The Netherlands:

- Access to NAP API to obtain metadata of all the relevant assets
- RML-based lifting from the country-specific JSON format to DCAT-AP 2.0 RDF metadata schema
- Upload of the resulting RDF graph to the RDF repository used by the Asset Manager

The details of the workflow are described in D5.5, together with an explanation of the mappings and the description of the overall metadata ingestion architecture.

2.10.2 Validation

The integration of NAP metadata in the Asset Manager can be verified by accessing either the Store or the Publisher application. We mapped the assets coming from multimodal National Access Points onto “Journey planning” asset, in Figure 42 we can see the page showing the list of assets belonging to such asset type.



[Home](#) > [Journey_Planning](#)

JOURNEY PLANNING assets

NEW

Description of the network of a Transport Service Provider for journey planning purposes, comprising stop places, lines, and timetables.

🔗 Journey_Planning assets are managed by the following lifecycle processes:

⚙️ [Default Lifecycle](#)



Velopark - all bicycle parkings in Belgium

Published by: [Fietsberaad Vlaanderen](#)



GTFS

Published by: [Stichting OpenGeo](#)



Table of CO2 emission

Published by: [Ministerie van Infrastructuur en Waterstaat](#)



MaaS Service Provider table

Published by: [Ministerie van Infrastructuur en Water](#)

Figure 42 Asset Manager Publisher showing results from multiple NAPs

Figure 43 shows how the metadata of an asset belonging to a NAP are rendered and shown to the Publisher users. Differently from “local” assets, the page showing remote assets details will not provide any versioning information nor the list of attachments. Anyway the Journey Planning asset type allows users to define a “Remote GTFS URL” as part of the provided metadata, therefore any GTFS URL will be shown together with all the other metadata.



Réseau urbain Vbus

Asset name

Réseau urbain Vbus

Description

Réseau urbain Vbus

Version**Author****Author Email****Institution**

Horaires théoriques du réseau Vbus - Communauté d'Agglomération de Vesoul (GTFS)

Resource type

Remote GTFS

URL<https://www.data.gouv.fr/fr/datasets/r/fd60af93-7e31-4666-99fc-560780336732>

Figure 43 NAP asset details

The manual analysis of a subset of the RDF triples obtained by ingesting NAP metadata showed a significant data quality problem. NAPs are meant to be repositories for timetables and real-time information, therefore being repositories for structured data. We noticed that many countries implemented their solutions using open data portals, without any check about the contents of the published information and without any distinction between structured or unstructured data and without any dataset categorization. As a result, accessing for example metadata about the “Rail” mode in the Belgian NAP shows both timetables and statistics about punctuality.

There are two strategies to mitigate such problem. The former is to improve the NAP ingester by leveraging on the API of the specific NAP, whenever they are available. This strategy can help reducing the number of assets which are incorrectly shown in the Asset Manager. Another strategy is represented by using the BPMN features of the Asset Manager itself to implement an approval

system. In this case, only the users of the Publisher application (belonging to the “Providers” group) would be allowed to see all the unfiltered metadata directly coming from the NAP. Publisher users would then be allowed to candidate an asset to be “promoted” to the Asset Manager Store.

2.11 SCENARIO S14: ASSET MANAGER AS A TOOL FOR CONTRIBUTING TO A NATIONAL ACCESS POINT

Table 18 Scenario S14: Asset Manager as a tool for contributing to a National Access Point

Actor	EuroTrain, a rail TSP focused on cross-border travels.
Target Component/Sub-system/Entity	Asset Manager, Converter
Description	This scenario demonstrates the usage of the Asset Manager as a tool for contributing to National Access Points.
Story	Following the activities related to S13, EuroTrain has planned his new cross-border service connecting France and Netherlands through Belgium. According to the NAP regulations, they must contribute their Journey Planning data to the NAP of each country. They publish their GTFS dataset just once, informing the Asset Manager that the dataset is related to a cross-border operator and including the names of the countries the dataset is related to. The Asset Manager takes care of activating the publication processes required by each NAP, and in case a conversion is required performs data conversion using an existing Converter.

During our analysis of the available National Access Points, we realized that while the automation features are able to cover all the requirements of this scenario, it is not possible to validate it attaching to a running NAP. Being able to add new datasets to a NAP requires being a registered TSP, and none of the partners of the SPRINT consortium is an authorized TSP in a country with an existing NAP (Ferrovie dello Stato is a transport operator but Italy currently has no multimodal National Access Point). We therefore believe that our high-level description of a NAP publication workflow provided in D4.3 remains valid, we believe that the current set of tools provided by the SPRINT project cover all its requirement, but we cannot provide a demonstration to validate our claims.

3. PERFORMANCE AND SCALABILITY EVALUATION

In this section, we evaluate the performance and scalability of the final release of the Interoperability Framework considering the Testing Infrastructure defined in deliverable D3.4. For each test case defined in D3.4, we follow the test checklist to specify the tests executed. The section presents the results of the testing activities, discussing the performance and scalability of the SPRINT IF, the improvements with respect of C-Rel, and the strengths and the limits of the various components.

3.1 COLLABORATIVE ONTOLOGY MANAGEMENT

3.1.1 TC1 - Performance Testing for Collaborative ontology manager

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.

Table 19 TC1 Performance Testing for Collaborative ontology manager

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
Preparation	S.O. Ubuntu, OnToology pipeline tool ²⁴ , Java
Testing Environment (setup)	The experiment configurations are loaded into a machine with the following characteristics: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, 40 cores, 32RAM, 50GB HDD with Ubuntu 18.04 as its operating system, Jenkins v2.222.4, Apache Maven 3.6.3, Java 8
Related Scenarios	Scenario 11 - D5.4
Testing Scripts	Jenkins pipeline with steps for each tool from OnToology

²⁴ https://github.com/oeg-upm/sprint/blob/main/collaborative_ontology/Jenkinsfile

Execution	Deployed and executed with the Jenkins server
Analysing results	The average of the execution time for each generation of the documentation and evaluation.

Preparation of the test environment and setup TC1

First, we select all software needed to perform tests on our IF component, the Collaborative Ontology tool. In our test case, we have chosen the following software: Ubuntu as operating system, Jenkins as test automation tool and the set of tools that currently work with OnToology²⁵. The software versions are detailed in the table above (see Table 19).

Update the Jenkinsfile²⁶ and add the name of the directory and the path where the OWL file of repository is located, as you can see in Figure 44.

```

27     environment {
28         VERSION      = '1.0'
29         WIDOCO        = '1.4.14'
30         AR2TOOL       = 'v.1.0'
31         VOCABLITE     = '1.0.2'
32         Ontology_dir  = 'Ontology' // -- name of the directory where the ontology is located
33         Ontology_path = 'Ontology/al.o.owl' // -- path where the ontology is located
34     }
35 }
36

```

Figure 44 Constants defined in the Jenkinsfile

Validating performance test TC1

We performed rehearsals to ensure that the tests execute correctly and validate test correctness. We verified that the HTML is correctly generated with the documentation and evaluation from the ontology of received as input.

Tool : The steps for running Ontology are defined in a Groovy script and running through a Jenkinsfile defining a Jenkins pipeline.

Input: A test ontology that is currently used for testing OnToology, the alo.owl²⁷. Using this ontology we ensure the coverage of relevant parameters that can affect the performance of our tool.

Output: The result is HTML files with the documentation, evaluation and visualization of the input ontology.

²⁵ <https://ontology.linkeddata.es/>

²⁶ https://github.com/oeg-upm/sprint/blob/main/collaborative_ontology/Jenkinsfile

²⁷ <https://github.com/oeg-upm/sprint/blob/main/CollaborativeOntologyManager/Ontology/al.o.owl>

Execution TC1

The automatation of the execution of test steps over the collaborative ontology tool ensures that each individual test or subsequent re-test are run in the same manner. Figure 45 shows a set of equal tests that have executed the pipeline. Jenkins automatically is able to calculate the average of the tests, giving relevant information to ensure the identification of bottlenecks in this process.

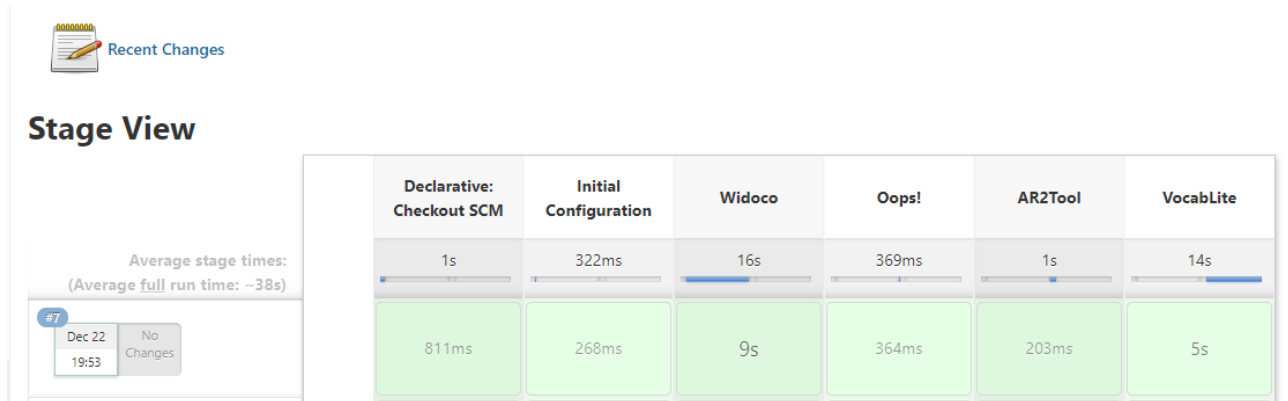


Figure 45 Pipelines execution

Analysis of the results TC1

For collaborative ontology tool, the documentation and evaluation also imply versioning of the ontology in the user's repository. The activity of documentation, generation and validation for one ontology with the reference to the User Story US-9, where we define the use case scenario for F-REL implementation, is the execution time of the whole process of both the evaluation and the generation of documentation.

Figure 46 shows the results obtained in **milliseconds** of the principal steps after the execution of the tool over an ontology, these average times can also be visualized in the Jenkins server that gives by default the execution times of the different steps executed over the Collaborative Ontology Tool. Therefore, as it can be seen in Figure 46, that the average time of generation of the documentation and evaluation of the ontology is approximately 16 seconds.

average time, VocabLite, Widoco and Oop!

■ average time ■ VocabLite ■ Widoco ■ Oop!

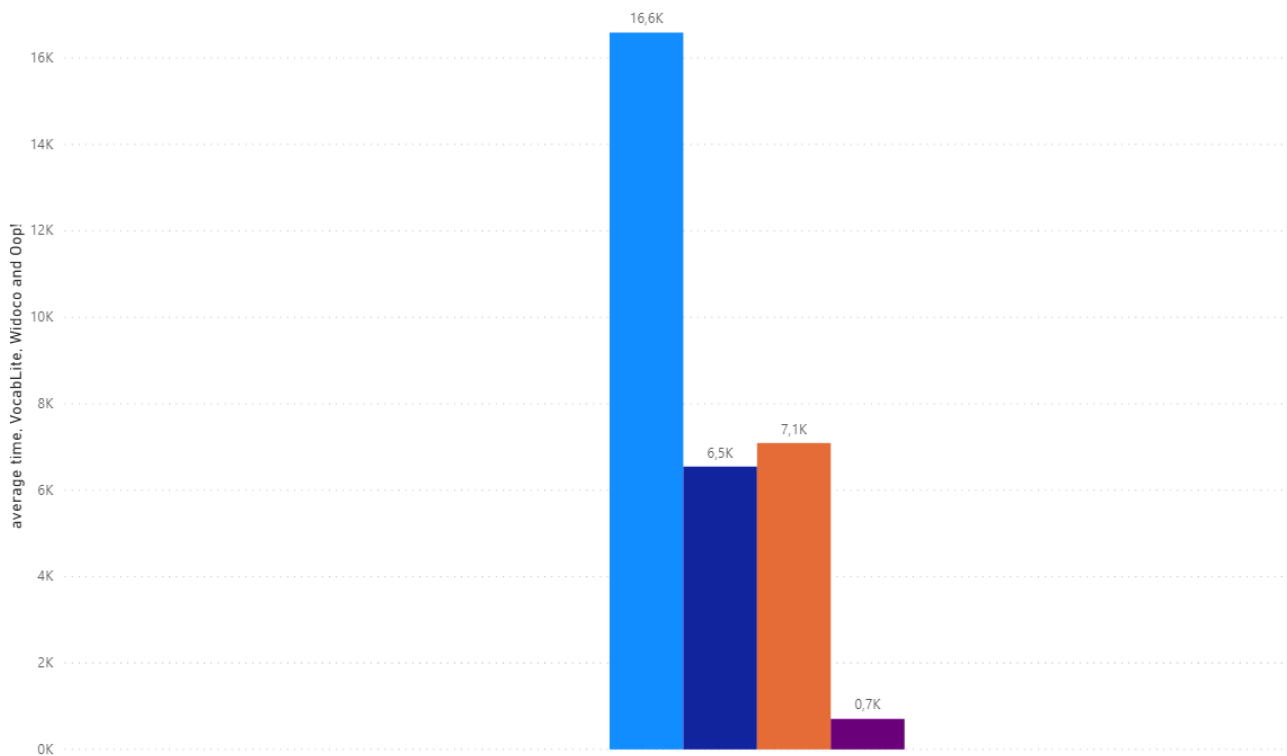


Figure 46 Results of the execution of the Collaborative Ontology Tool

3.1.2 TC2 - Scalability Testing for Collaborative ontology manager

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

Table 20 TC2 Scalability Testing for Collaborative ontology manager

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
Preparation	S.O. Ubuntu, OnToology pipeline tool ²⁸ , Java
Testing Environment (setup)	The experiment configurations are loaded into a machine with the following characteristics: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, 40 cores, 32RAM, 50GB HDD with Ubuntu 18.04 as its operating system, Jenkins v2.222.4, Apache Maven 3.6.3, Java 8
Related Scenarios	Scenario 11 - D5.4
Testing Scripts	Jenkins pipeline with steps for each tool from OnToology ²⁹
Execution	Deployed and executed with the Jenkins server
Analysing results	The execution time for each generation of the documentation and evaluation.

²⁸ <https://github.com/oeg-upm/sprint/blob/main/CollaborativeOntologyManager/Jenkinsfile>

²⁹ <https://github.com/oeg-upm/sprint/tree/main/CollaborativeOntologyManager>

Preparation of the test environment and setup TC2

The software needed to perform tests on the Collaborative Ontology tool is, in our case, the same as in TC1: Ubuntu as operating system, Jenkins as test automation tool and the set of tools that currently work with OnToology³⁰. The software versions are detailed in the table above (see Table 20).

For this test, a new GitHub repository has been created where the ontologies are uploaded and the Jenkinsfile was updated (see in Figure 47). All these files are in the ontology directory and our pipeline is prepared to execute and generate the documentation and evaluation in each of them in separate directories. We take the advantage of the transformation that the XSD2OWL tool performs on NeTEx³¹ files to perform this test.

```
161 def ontologies() {
162     return ["alo.owl",
163         "netex_accounting_version.xsd.owl",
164         "netex_address_support.xsd.owl",
165         "netex_availabilityCondition_support.xsd.owl",
166         "netex_availabilityCondition_version.xsd.owl",
167         "netex_country_support.xsd.owl",
168         "netex_dayType_propertiesOfDay.xsd.owl",
169         "netex_dayType_support.xsd.owl",
170         "netex_dayType_version.xsd.owl",
171         "netex_equipment_support.xsd.owl",
172         "netex_equipment_version.xsd.owl",
173         "netex_facility_support.xsd.owl",
174         "netex_facility_version.xsd.owl",
175         "netex_facilityUic_support.xsd.owl",
176         "netex_mode_support.xsd.owl",
177         "netex_mode_version.xsd.owl",
178         "netex_notice_support.xsd.owl",
179         "netex_notice_version.xsd.owl",
180         "netex_otherOrganisation_support.xsd.owl"]
181 }
182
```

Figure 47 Ontologies generated from NeTEx XML schema files

Validating performance test TC2

We performed rehearsals to ensure that the tests execute correctly and validate test correctness. We verified that the HTML is correctly generated with the documentation and evaluation from the ontology of received as input.

³⁰ <https://ontology.linkeddata.es/>

³¹ https://github.com/NeTEx-CEN/NeTEx/tree/master/xsd/netex_framework/netex_reusableComponents

Tool: The steps for running Ontology are defined in a Groovy script and running through a Jenkinsfile defining a Jenkins pipeline.

Input: Set of ontologies described in OWL.

Output: The results are HTML files with the documentation, evaluation, and visualization for each ontology in different directories.

Execution TC2

For this case, the test automation of the collaborative ontology over several ontologies is done in the same way as the performance test, since it is also deployed using a Jenkins server (see Figure 48).

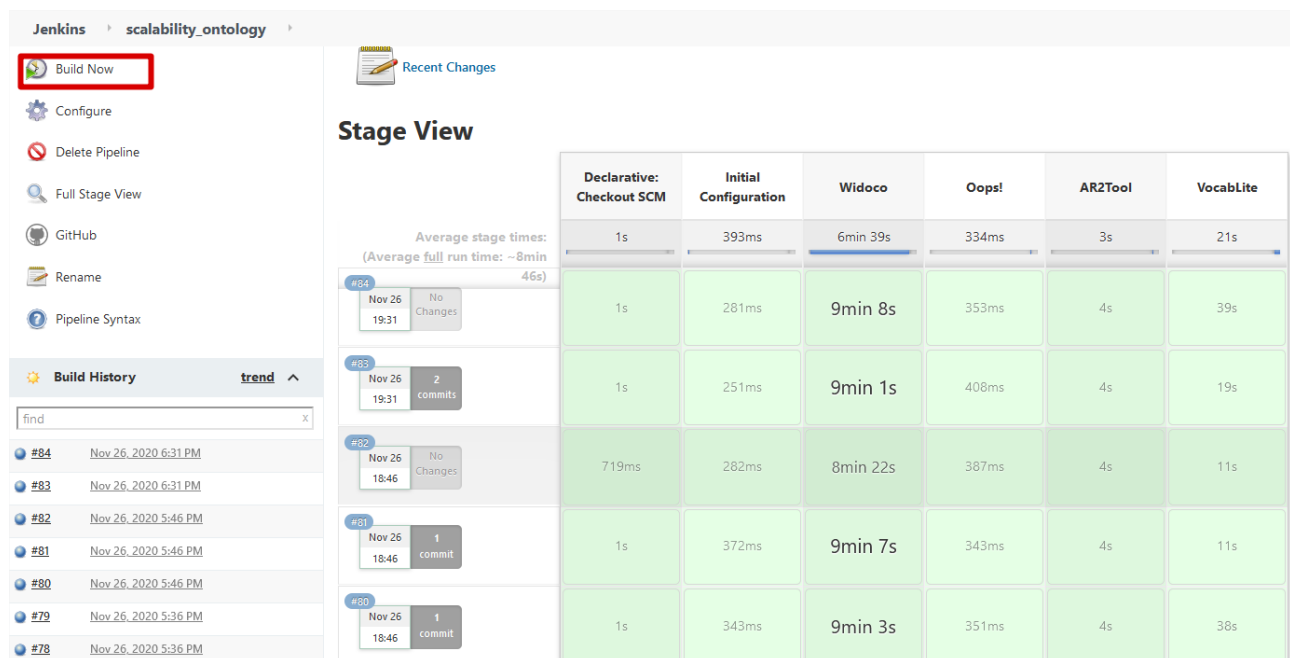


Figure 48 Pipelines execution over several ontologies

Analysis of the results TC2

We generate documentation and evaluate quality for multiples ontologies to measure how the input of multiple ontologies and their growth in the number of concepts affects the performance of the documentation generation and evaluation process.

Figure 49 shows an example of an ontology obtained from the transformation of an XSD file from NeTeX XSD to OWL file. In this figure, we can see the number of nodes displayed through the Webvowl³² tool to see the number of nodes in the ontology files. In the repository, we have images of how a file has a low number of nodes(*netex_accounting_version*³³), and another that has a high

³² <http://www.visualdataweb.de/webvowl/>

³³ https://github.com/oeg-upm/sprint/blob/main/collaborative_ontology/images/netex_accounting_version.owl.svg

number of nodes(*netex_facility_version*³⁴) from XSD files transformed. This allows us to make an approximate measure of the time of generation of documentation and evaluation as the file grows at the level of classes and relationships.



Figure 49 View of the netex_facility_support.owl file

Finally, Figure 50 shows the times in **seconds** of the generation of the documentation and evaluation from ontologies with minimum nodes/relations to the largest file in terms of concept and relations already explained in the previous paragraph where the minimum time corresponds to the file with the least relations and the maximum time the file with more relations/nodes. For this evaluation, the ontologies generated from the NeTEx files have been used again.

³⁴ https://github.com/oeg-upm/sprint/blob/main/collaborative_ontology/images/netex_facility_version.owl.svg

elapsed_time by file

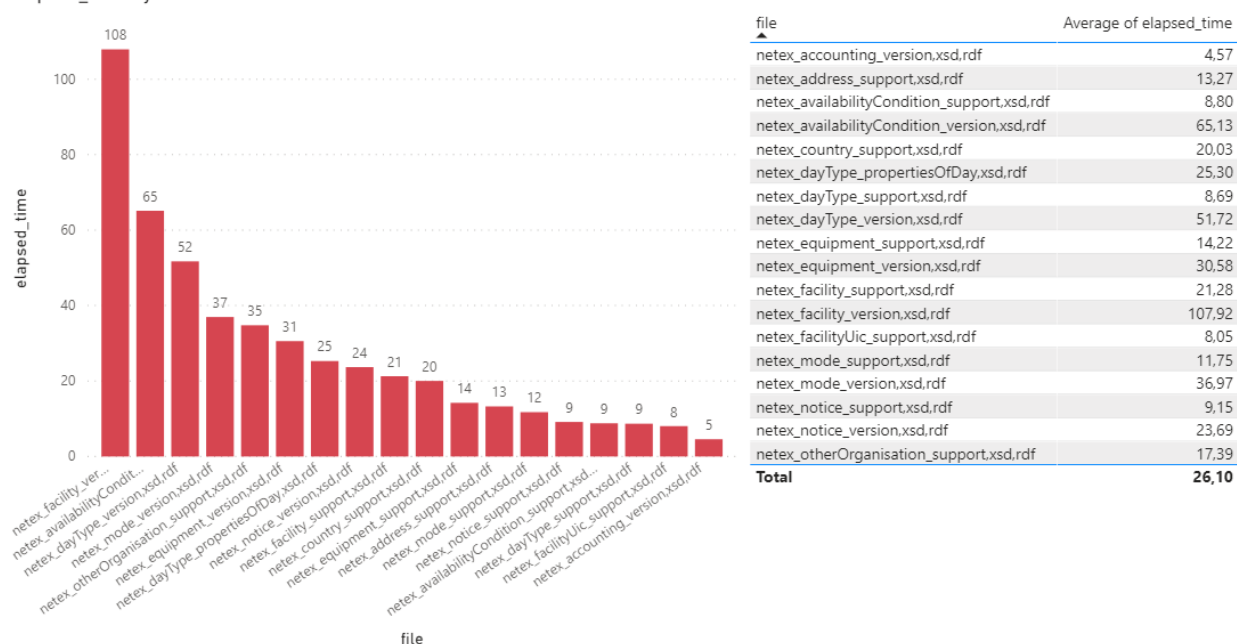


Figure 50 Results from scalability test

3.2 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.

3.2.1 TC3 - Performance Testing for Automatic generation of ontologies from non-ontological sources

For the TC3 performance test we are going to obtain times in the transformation of an XSD file to an OWL file. For them we have summarized the processes, performance criteria and environment in the following table:

Table 21 TC3 Performance Testing for Automatic generation of ontologies from non-ontological sources

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
Preparation	S.O. Ubuntu, XSD2OWL tool ³⁵ , Java , Apache Maven
Testing Environment (setup)	The experiment configurations are loaded into a machine with the following characteristics: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, 40 cores, 32RAM, 50GB HDD with Ubuntu 18.04 as its operating system, Jenkins v2.222.4, Apache Maven 3.6.3, Java 8
Related Scenarios	Scenario 12 - D5.4
Testing Scripts	Bash script with a test plan from each case ³⁶
Execution	The generated script plan is deployed and executed with the Jenkins server
Analysing results	The time , maximum memory usage(KB), CPU consumption, of the execution.

Preparation of the test environment and setup TC3

We need to create a test environment as a close as the production environment to ensure that the process of the application is run under similar characteristics (see Figure 51). The test plans are determined: converting XSD to OWL using NeTEx XSD. The scripts are created to automate the planned tests and each script, defined in bash, together with Jenkins automatize each test case. To setup the test environment in Jenkins, we have to create a new item. First, we go to the Jenkins

³⁵ <https://github.com/oeg-upm/sprint/tree/main/xsd2owl>

³⁶ <https://github.com/oeg-upm/sprint/blob/main/xsd2owl/performanceTest.sh>

dashboard and create new item (see Figure 52). Then, we define the source based on our GitHub³⁷ repository, following the instruction we shown in Section 2.9 (Figure 53).

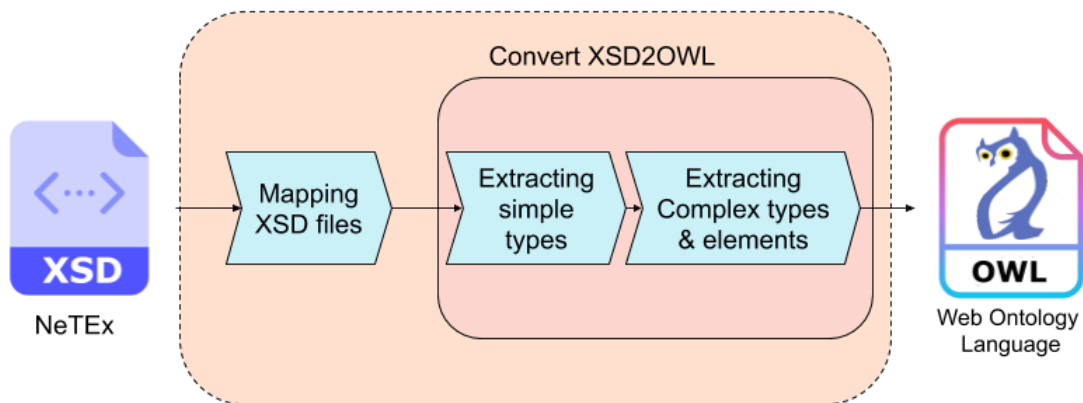


Figure 51 NeTeX XSD to OWL

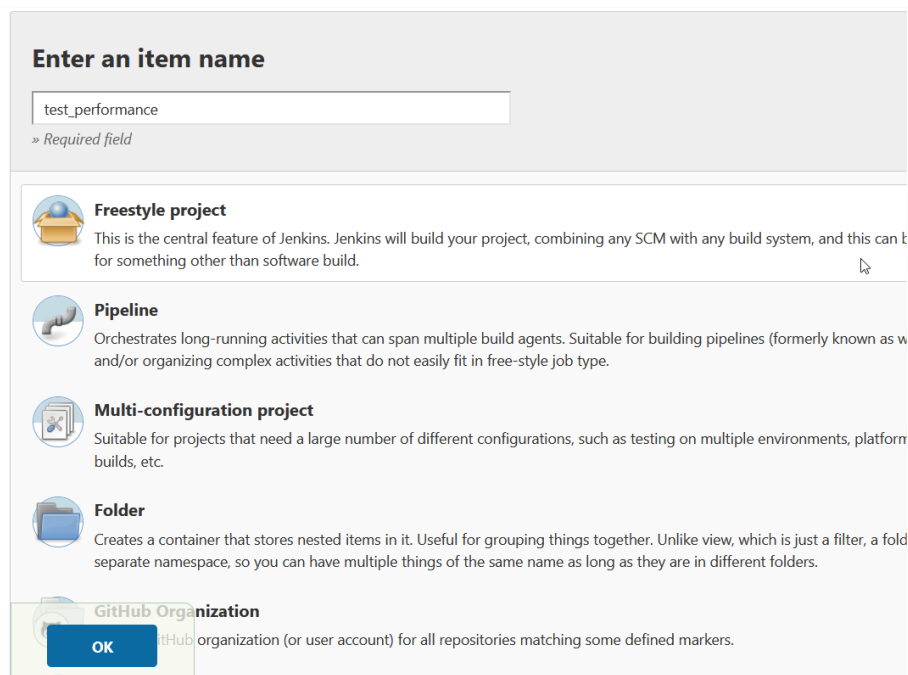
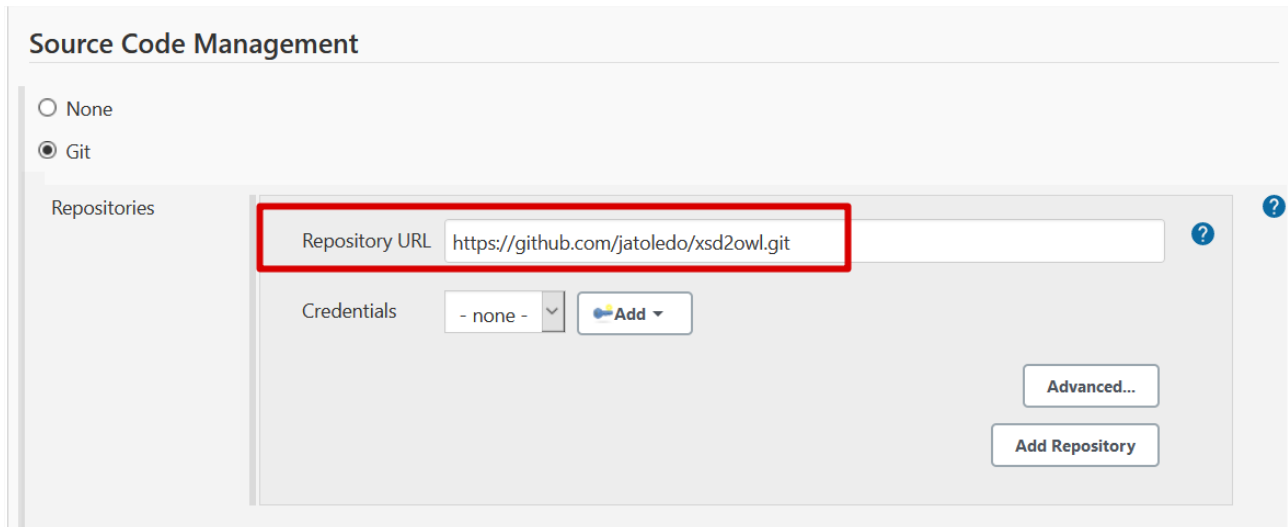


Figure 52 Create new Item Jenkins

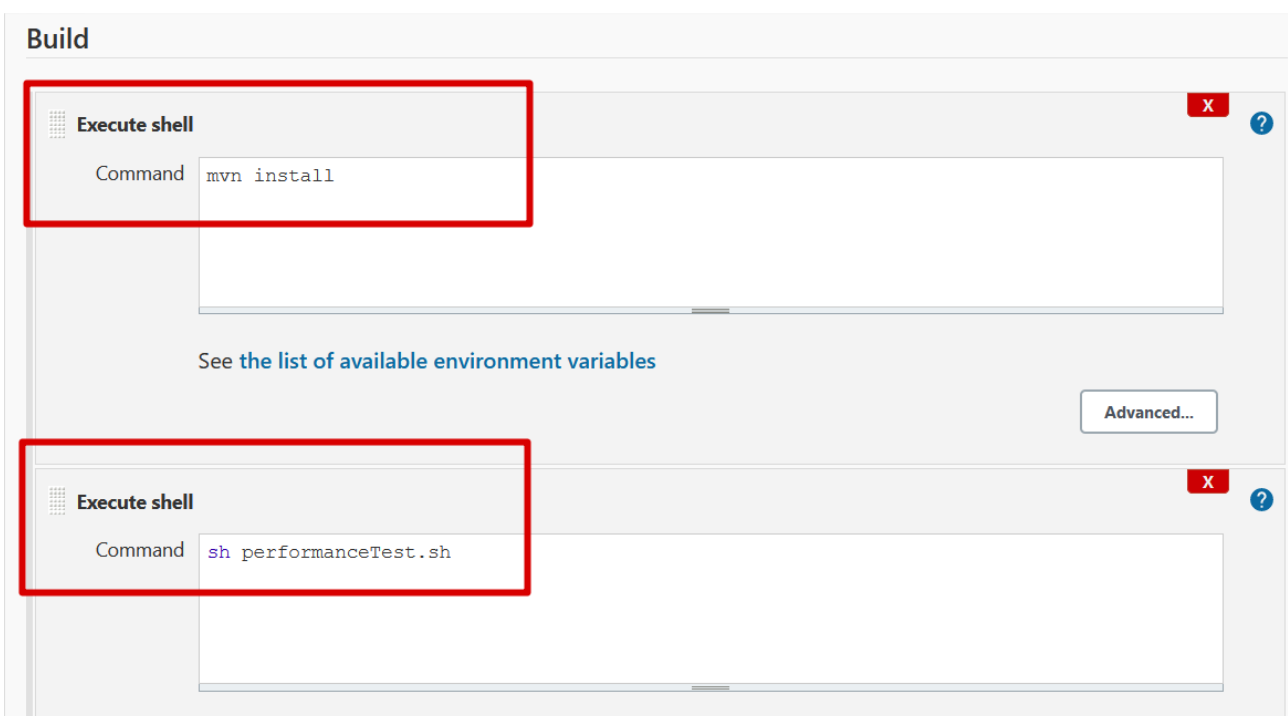
³⁷ <https://github.com/oeg-upm/sprint/tree/main/xsd2owl>



The image shows the 'Source Code Management' configuration page in Jenkins. The 'Git' radio button is selected. Under the 'Repositories' section, a red box highlights the 'Repository URL' field, which contains the text 'https://github.com/jatoledo/xsd2owl.git'. Below this, the 'Credentials' dropdown is set to '- none -' with an 'Add' button next to it. At the bottom right of the configuration area, there are buttons for 'Advanced...' and 'Add Repository'.

Figure 53 Setting up GitHub repository on Jenkins

Over the run section, we have to add two steps using bash command line instructions. The first builds the application using Maven and the second runs the test script³⁸ as shown in Figure 54. This test script is created to run each test case allowing the definition of the input parameters.



The image shows the 'Build' configuration page in Jenkins. Two 'Execute shell' steps are added, each highlighted with a red box. The first step has the command 'mvn install'. The second step has the command 'sh performanceTest.sh'. Between the two steps, there is a link that says 'See the list of available environment variables'. At the bottom right of each step's configuration area, there is an 'Advanced...' button.

Figure 54 Build configuration on Jenkins

³⁸ <https://github.com/oeg-upm/sprint/blob/main/xsd2owl/performanceTest.sh>

Validating performance test TC3

We have validated the tests with some examples and verified that the initial results are the ones we want to achieve. Performance test execution principally involved multiple executions of the number of XSD files and obtaining an average execution time to transform an XML schema to an ontology.

Tool : XSD2OWL is a tool that allows to transform XML schema files to owl.

Input: NeTEx Schema. NeTEx is a CEN Technical Standard for exchanging Public Transport schedules and related data³⁹.

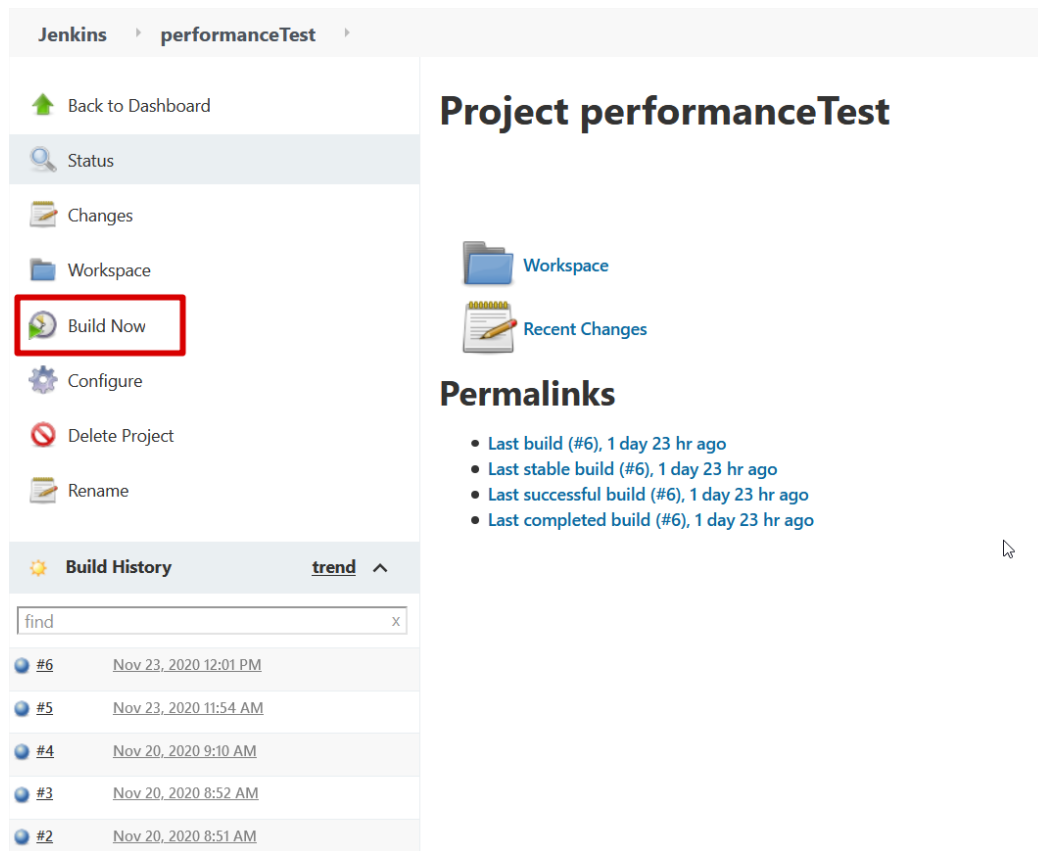
Output: OWL file that allows being used in applications that need to process the information content facilitating a better interpretability

Script: The script developed for this test is based on a bash script with loops that allow us to perform several executions and measure the execution times when performing an XSD to OWL transform.

Execution TC3

Once the environment configuration and validations is prepared, we can see the task in Jenkins' dashboard. To run the test we can access the task and run "Build now" (see Figure 55), this will execute what we have defined in the previous step. In each build, we can see the log that the code has been completed correctly and we can also see the log of the test execution (see Figure 56 and Figure 57).

³⁹ <http://netex-cen.eu/>



Jenkins ▶ **performanceTest** ▶

Back to Dashboard

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

Build History trend ^

find X

#	Time
#6	Nov 23, 2020 12:01 PM
#5	Nov 23, 2020 11:54 AM
#4	Nov 20, 2020 9:10 AM
#3	Nov 20, 2020 8:52 AM
#2	Nov 20, 2020 8:51 AM

Project performanceTest

Workspace

Recent Changes

Permalinks

- Last build (#6), 1 day 23 hr ago
- Last stable build (#6), 1 day 23 hr ago
- Last successful build (#6), 1 day 23 hr ago
- Last completed build (#6), 1 day 23 hr ago

Figure 55 Example build execution

```
[@[1;34mINFO@] Replacing original artifact with shaded artifact.
[@[1;34mINFO@] Replacing /var/lib/jenkins/workspace/performanceTest/target/xsd2owl-1.0.0.jar with /var/lib/jenkins/workspace/performanceTest/target/xsd2owl-1.0.0-shaded.jar
[@[1;34mINFO@] Dependency-reduced POM written at: /var/lib/jenkins/workspace/performanceTest/dependency-reduced-pom.xml
[@[1;34mINFO@] Dependency-reduced POM written at: /var/lib/jenkins/workspace/performanceTest/dependency-reduced-pom.xml
[@[1;34mINFO@]
[@[1;34mINFO@] @[1m--- @[0;32mmaven-install-plugin:2.4:install@] @[1m(default-install)@] @[36mxsd2owl@]@[0;1m ---@]
[@[1;34mINFO@] Installing /var/lib/jenkins/workspace/performanceTest/target/xsd2owl-1.0.0.jar to /var/lib/jenkins/.m2/repository/tr/com/srdc/xsd2owl/1.0.0/xsd2owl-1.0.0.jar
[@[1;34mINFO@] Installing /var/lib/jenkins/workspace/performanceTest/dependency-reduced-pom.xml to /var/lib/jenkins/.m2/repository/tr/com/srdc/xsd2owl/1.0.0/xsd2owl-1.0.0.pom
[@[1;34mINFO@] @[1m-----@]
[@[1;34mINFO@] @[1;32mBUILD SUCCESS@]
[@[1;34mINFO@] @[1m-----@]
[@[1;34mINFO@] Total time: 4.289 s
[@[1;34mINFO@] Finished at: 2020-11-20T09:10:21Z
[@[1;34mINFO@] @[1m-----@]
```

Figure 56 Build success Log

```
**** netex_dayType_propertiesOfDay.xsd ****  
AUTOMATIC GENERATION OF ONTOLOGIES FROM NON-ONTOLOGICAL SOURCES  
Time 1197 ms  
-----Process DONE-----  
**** netex_dayType_support.xsd ****  
AUTOMATIC GENERATION OF ONTOLOGIES FROM NON-ONTOLOGICAL SOURCES  
Time 756 ms  
-----Process DONE-----  
**** netex_dayType_version.xsd ****  
AUTOMATIC GENERATION OF ONTOLOGIES FROM NON-ONTOLOGICAL SOURCES  
Time 1479 ms  
-----Process DONE-----
```

Figure 57 Test execution log

Analysis of the results TC3

We obtained the results in a CSV file where the following information is saved for each execution when transforming an XSD file to OWL:

- *Elapsed_time*: Elapsed real (wall clock) time used by the process, in seconds.
- *Kernel_mode*: Total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
- *User_mode*: Total number of CPU-seconds that the process used directly (in user mode), in seconds.
- *Memory_max(Kbytes)*: Maximum resident set size of the process during its lifetime, in Kbytes.

In Figure 58 we show the results of transforming time of each XML Schema file from NeTEx to OWL. The largest evaluated file is `netex_facility_version.xsd`⁴⁰ and contains 68 simpleType and 36 complexType, we know that an element is complex (complexType) when it contains one or more elements and/or attributes and the smallest XSD is `netex_accounting_version.xsd`⁴¹ which is a basic representation of the Schema. With respect to the other measured parameters, these are also within our permitted ranges and we do not see an excessive consumption of memory and time in any execution (see Figure 59).

⁴⁰ https://github.com/oeg-upm/sprint/blob/main/xsd2owl/src/test/resources/netex_reusableComponents/netex_facility_version.xsd

⁴¹ https://github.com/oeg-upm/sprint/blob/main/xsd2owl/src/test/resources/netex_reusableComponents/netex_accounting_version.xsd

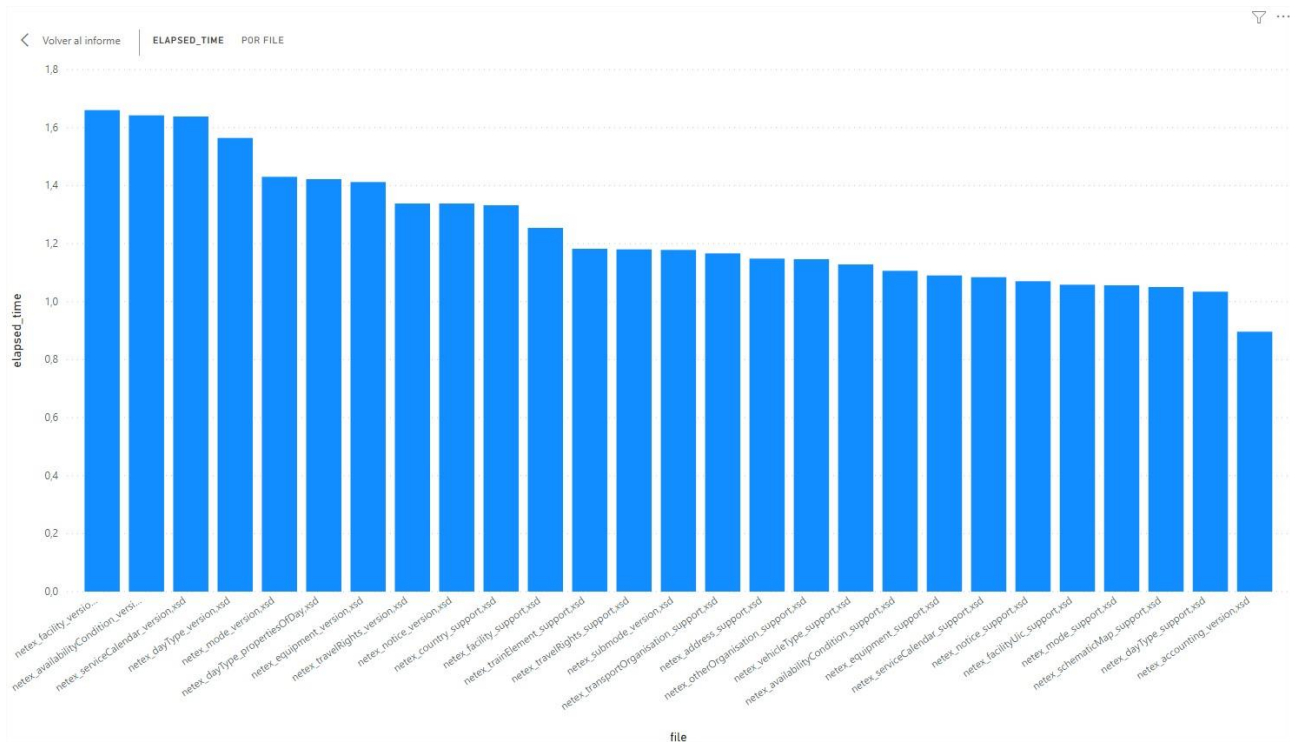


Figure 58 Elapsed time for each XSD file

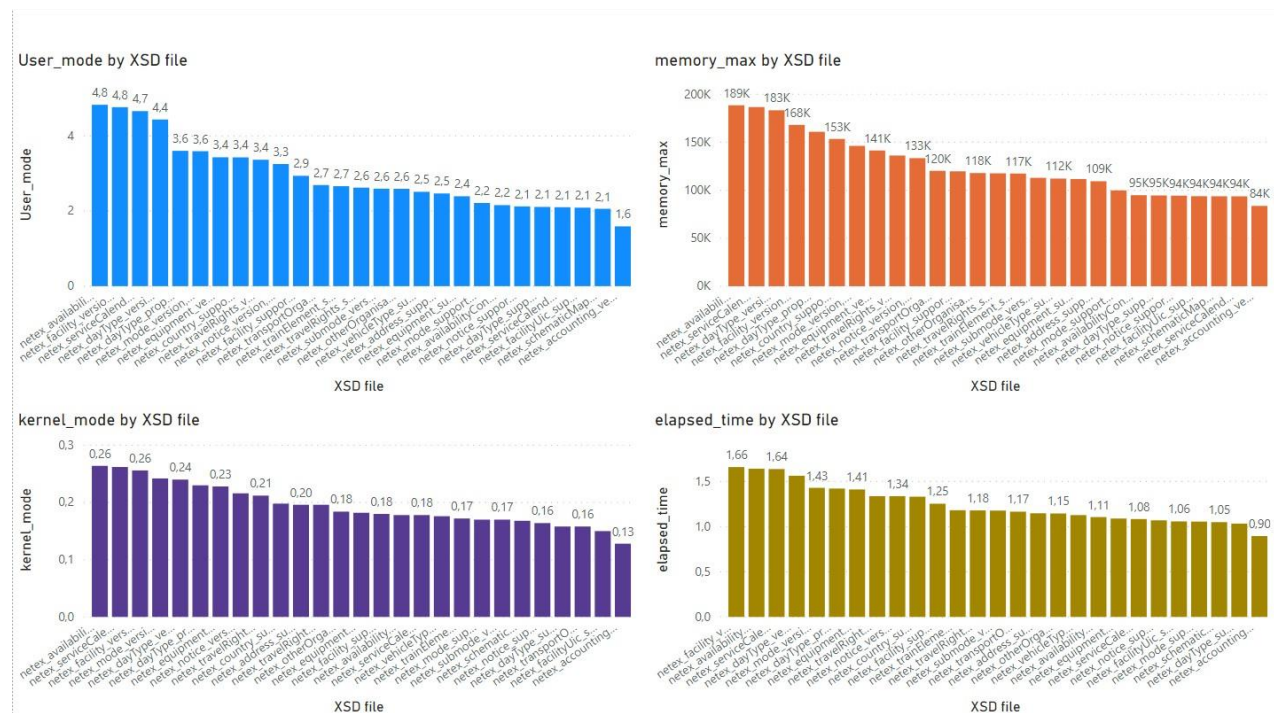


Figure 59 Results with all measured parameters

3.2.2 TC4 - Scalability Testing for Automatic generation of ontologies from non-ontological sources

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.

Table 22 TC4 Scalability Testing for Automatic generation of ontologies from non-ontological sources

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
Preparation	S.O. Ubuntu, XSD2OWL tool ⁴² , Java , Apache Maven
Testing Environment (setup)	The experiment configurations are loaded into a machine with the following characteristics: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, 40 cores, 32RAM, 50GB HDD with Ubuntu 18.04 as its operating system, Jenkins v2.222.4, Apache Maven 3.6.3, Java 8
Related Scenarios	Scenario 12 - D5.4
Testing Scripts	Bash script with a test plan from each case
Execution	The generated script plan is deployed and executed with the Jenkins server
Analysing results	The time, maximum memory usage(KB), CPU consumption, of the execution.

⁴² <https://github.com/oeg-upm/sprint/tree/main/xsd2owl>

Preparation of the test environment and setup TC4

For the scalability tests, we use three XSD files from NeTEx with different simple and complex types. Figure 60, is shown the first file, `netex_facility_version.xsd` which the more bigger and has 36 complex types and 38 simple types in the definition of its scheme. Figure 61 is shown the second file `netex_accounting_version.xsd` that does not have simple or complex types and finally `netex_facility_support.xsd`⁴³ file that has intermediate values of complex and simple types. These experiments will measure the times when increasing the types (simple and complex) in the XML Schema.

@xmlns	http://www.netex.org.uk/netex
@xmlns:netex	http://www.netex.org.uk/netex
@xmlns:siri	http://www.siri.org.uk/siri
@xmlns:xsd	http://www.w3.org/2001/XMLSchema
@xmlns:ifopt	http://www.ifopt.org.uk/ifopt
@targetNamespace	http://www.netex.org.uk/netex
@elementFormDefault	qualified
@attributeFormDefault	unqualified
@version	1.0
@id	netex_facility_version

Figure 60 Netex_facility_version.xsd

Data Source: netex_accounting_version.xsd **Well-Formed XML** ✓

edited with XMLSpy v2013 (x64) (http://www.altova.com) by Nicholas JS Knowles (Trapeze Group Limited)

@xmlns	http://www.netex.org.uk/netex
@xmlns:netex	http://www.netex.org.uk/netex
@xmlns:siri	http://www.siri.org.uk/siri
@xmlns:xsd	http://www.w3.org/2001/XMLSchema
@targetNamespace	http://www.netex.org.uk/netex
@elementFormDefault	qualified
@attributeFormDefault	unqualified
@version	0.1
@id	netex_journey_version

Figure 61 Netex_accounting_version.xsd

Validating performance test TC4

Scalability test execution principally involved the transformation of an XML schema into an ontology for multiple numbers of elements and types and obtaining an average execution time

Tool : XSD2OWL is a tool that allows to transform XML schema files to owl.

Input: NeTEx Schema. NeTEx is a CEN Technical Standard for exchanging Public Transport schedules and related data (<http://netex-cen.eu/>).

⁴³ https://github.com/NeTEx-CEN/NeTEx/blob/master/xsd/netex_framework/netex_reusableComponents/netex_facility_support.xsd

Output: OWL file that allows being used in applications that need to process the information content facilitating a better interpretability

Script: The script developed for this test is based on a bash script with loops that allow us to perform several executions and measure the execution times when performing an XSD to OWL transform from files with different elements and types.

Execution TC4

Since this test is executed with a bash script on a Jenkins server the execution is the same as TC3.

Analysis of the results TC4

In the top graphs of Figure 62 is shown the number of simple and complex types that has each XML Schema and at the bottom it is shown a set of parameters measured in the transformation to an OWL file the number of types in the file schema are increased. As we can observe, the schema with more types in our test does not exceed 2 seconds in the transformation to an OWL file and the memory consumption of our test has not exceeded 1MB of memory. The results are interpreted in the same way as in the TC3 since it is measured in a similar way but by increasing the types (simple or complex) in the XML Schema.

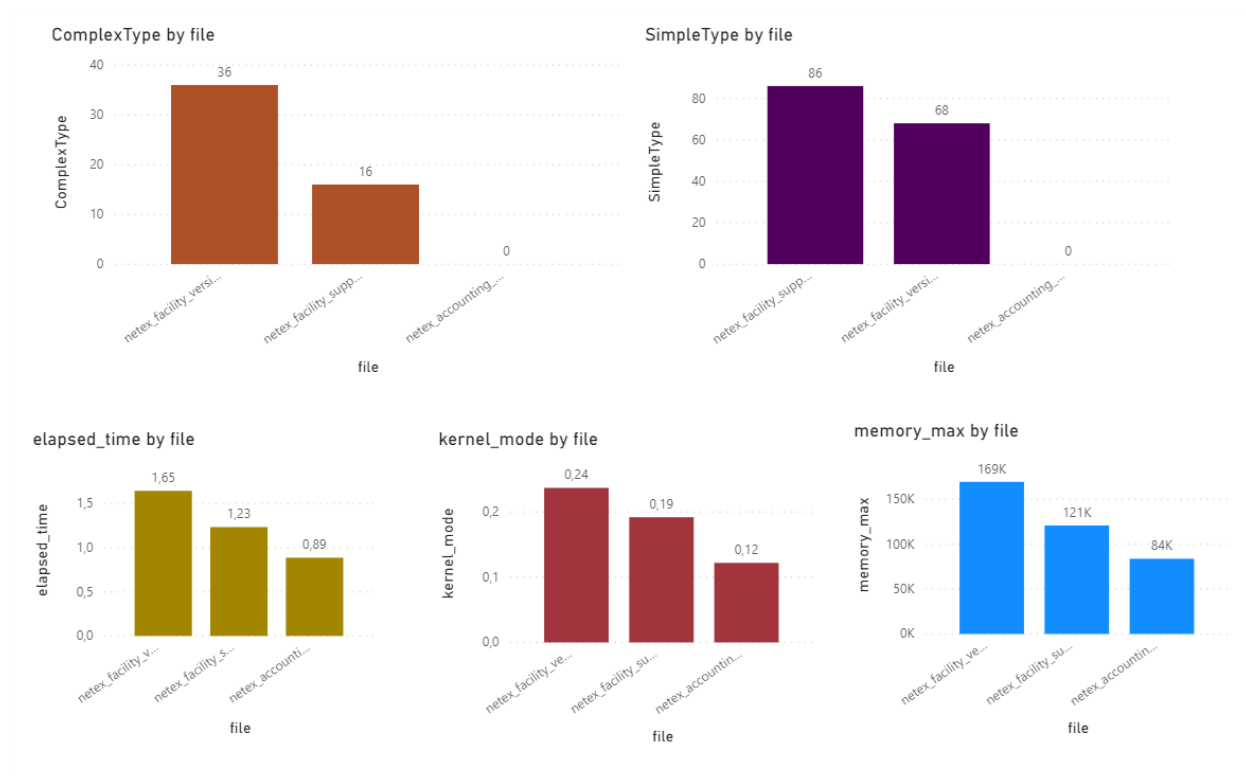


Figure 62 Scalability results

3.3 MAPPING TOOL

In this section, we evaluate the performance and usability of the mapping tool according to Table 23 and Table 24.

3.3.1 TC5 - Performance Testing Mapping Tool

Table 23 TC5 Performance Testing 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
Preparation	Three different standards (FSM, IT2Rail and Transmodel) has been choosen and prepared to create two test cases
Testing Environment (Setup)	CentOS/RHEL based Amazon Linux AMI 2018.03 x64 8 vCPUs 2.3 GHz, Intel Broadwell E5-2686v4 32 GB RAM 32GB internal disk storage Docker and docker-compose Npm v6.14.x and angular CLI v9.1.x installed
Testing Scripts	-
Execution	The program has been run four times; To generate two types of output (RML and Java based annotation) for two test cases.
Analysing Results	Total Average Execution Time, Average Execution Time for Mapping Generation, Average Execution Time for Java-based Annotation, Average Execution Time for RML-based Annotation, overall Execution Time of per Terms.

3.3.2 TC6 - Usability Testing Mapping Tool

Table 24 TC6 Usability Testing Mapping Tool

TC6 - Usability 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
Preparation	Three different standards (FSM, IT2Rail and Transmodel) has been choosen and prepared to create two test cases
Testing Environment (setup)	CentOS/RHEL based Amazon Linux AMI 2018.03 x64 8 vCPUs 2.3 GHz, Intel Broadwell E5-2686v4 32 GB RAM 32GB internal disk storage Docker and docker-compose Npm v6.14.x and angular CLI v9.1.x installed
Testing Scripts	-
Execution	Application has been run and we have counted the number of steps right from initiation of the program to the generation of the output.
Analysing Results	Number of steps

3.3.3 TC5 and TC6 Testing Activities

Preparation of the test environment and setup TC5-TC6

As represented in Table 25, the test cases include those introduced and used for functional validation of the Mapping and Annotation functionality in Section 2.4.2, however, each test case has been used to generate two types of output, i.e., Java Based annotation files and RML mappings.

Table 25 Mapping Tool Test Case Description

No.	Test Case	Source	Target	Annotation Type
1	Test Case 1-Round1 (TC5-1-R1)	FSM	Transmodel	Java-based
2	Test Case 1-Round2 (TC5-1-R2)	FSM	Transmodel	RML-based
3	Test Case 2-Round1 (TC5-2-R1)	FSM	IT2Rail	Java-based
4	Test Case 2-Round2 (TC5-2-R2)	FSM	IT2Rail	RML-based

Validating performance test TC5-TC6

The **Usability (TC6)** of the tool is straightforward to test, we need to run the program and count the number of steps. Then, to evaluate the **Execution Time (TC5)**, of the Mapping tool, we have tested the application through two test cases in four runs as explained above. The goal was to measure the overall execution time, as well as the execution time of various intermediate steps.

Tool: Mapping Tool

Input: Standard pairs (FSM-Transmodel, and, FSM-IT2Rail)

Output: Two types of outputs are expected based on the users choice. Either the RML or Java-files annotated based on the mapping results.

Execution TC5-TC6

represents the Execution Time evaluation results. The upper diagram represents the Mapping Generation Execution Time per each run of the application for test cases described in Table 25. Accordingly, we have measured the Annotation Execution Time for two types of the output, i.e., the Java-based and RML-based annotation. Therefore, TC5-1-R1 and TC5-2-R1 represent the execution time for Java-based annotation which is the sum of two steps, first to create the Java files (using JAXB) and second, to annotate them. Similarly, TC5-1-R2 and TC5-2-R2 represent the testing of the application for the RML-based annotation.

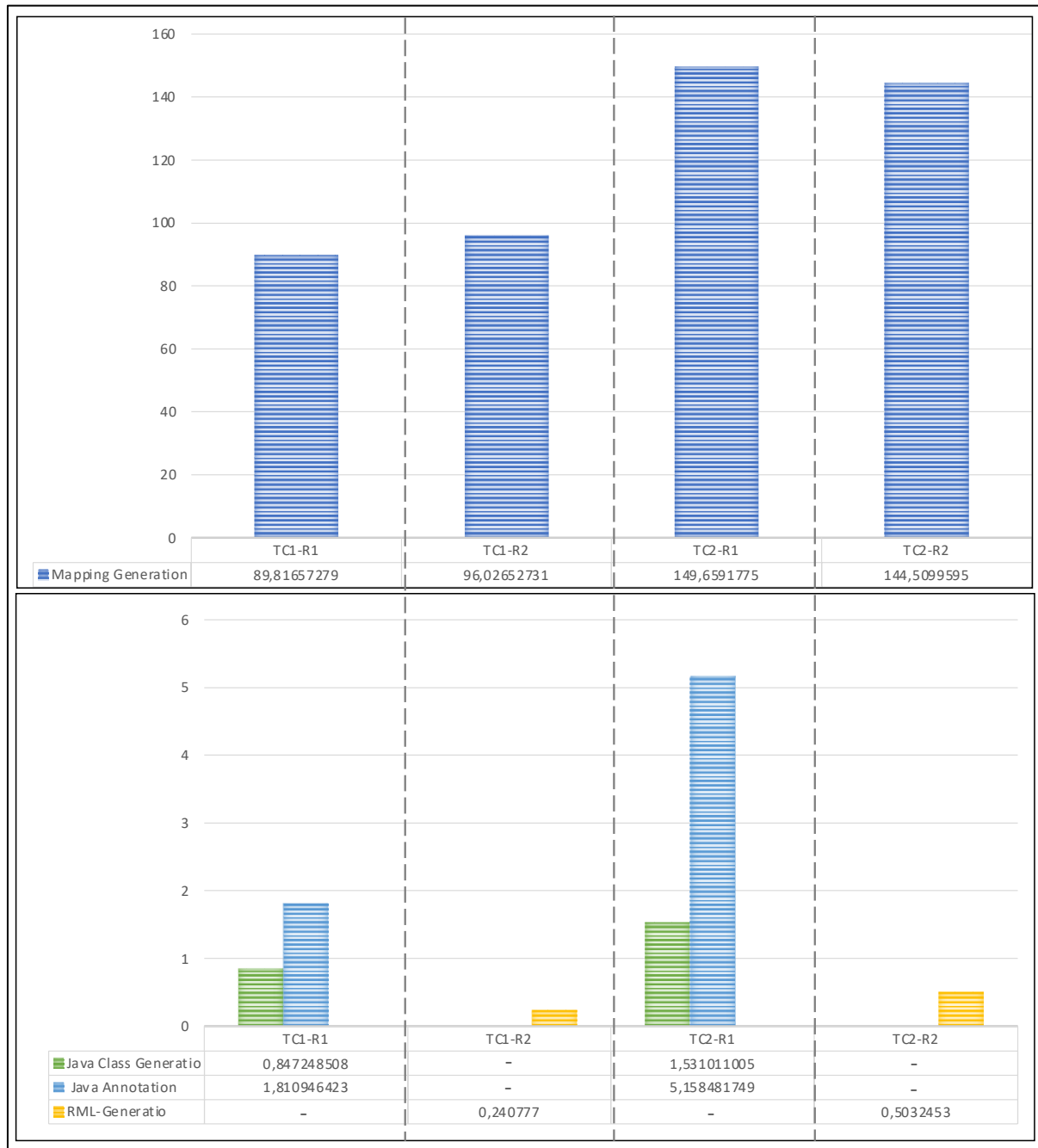


Figure 63 Mapping Tool Execution Time Evaluation

Analysis of the results TC5-TC6

As per TC6, as already demonstrated in Section 2.4.1, the overall steps to generate the mapping is below 10 steps. For the TC5, as summarized in Table 26, the analysis of , shows that the **average Mapping Execution Time** is **120,00 s**, **average Annotation Generation (Java and Rml) Time** is **3,85 s**, and finally the **total average Execution Time of the tool** is **122,5953 s**. Furthermore, given the overall unique terms in FSM, Transmodel and IT2Rail (Please see Section 23 Mapping Generation) are 83, 220 and 555 hence the test cases TC5-1-R1 and TC5-1-R2 have 303 terms, and, TC5-2-R1 and TC5-2-R2 have 638 terms. Accordingly, we can conclude that tool exhibits average execution time 0,6229 s/per term for TC5-1-R1 and TC5-1-R2, and 0,4723s/per term for TC5-2-R1 and TC5-2-R2 which gives an **overall Execution time of 0,5476 s/per terms**.

Table 26 Summary of Analysis of the Results of Mapping Tool Evaluations

No.	KPI	Value
1	Total Average Execution Time	122,5953 s
2	Average Execution Time for Mapping Generation	120 s
3	Average Execution Time for Java-based Annotation (Java-class generation included)	4,67 s
4	Average Execution Time for Java-based Annotation (Java-class generation excluded)	3,48 s
5	Average Execution Time for RML-based Annotation	0,37 s
6	Average Execution Time of per Terms for TC1 (R1 and R2)	0,6229 s
7	Average Execution Time of per Terms for TC2 (R1 and R2)	0,4723 s
8	Overall Execution Time of per Terms	0,5476 s

3.4 ASSET MANAGER

In this section, we evaluate the performance and usability of the Asset Manager Publisher as described in Table 27 and Table 29.

3.4.1 TC7 - Performance Testing for Asset manager

Table 27 TC7 Performance Testing for 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-2. Performance requirements for Exploration API: DCAT-AP metadata retrieval
Performance Criteria	Average page loading time.
Preparation	A deployed instance of the Asset Manager populated with assets. Jenkins with the Performance plugin. JMeter tool.
Testing Environment (setup)	Asset Manager deployment: CentOS Linux 7, Intel® Xeon 8-core CPU D-1521 @ 2.40GHz and 64 GB Memory, Docker v19.01.13, docker-compose v1.18.0 Jenkins and Jmeter running on a separated machine with the following specifications: Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz, 4 core; 12 GB RAM; CentOS Linux release 7.8.2003 (Core), Jenkins v2.235.2, Apache JMeter v5.3.
Testing Scripts	A JMeter test plan is defined to perform requests to the Asset Manager simulating the average expected workload.
Execution	The JMeter test plan is executed through Jenkins and results are gathered.

Analysing Results	Response times and the number of performed/failed requests are collected.
-------------------	---

Preparation and Setup TC7

The testing machine is configured with Jenkins and the JMeter tool, while the Asset Manager is deployed on a different server. The hardware, software specification of the test environment are described in Table 27 (Setup).

The Apache JMeter tool, open-source software designed to load test functional behaviour and measure performance⁴⁴, is used for configuring test plans (*jmx* files) and running them for the different converters. The test plans were uploaded and set up to be executed through a Jenkins pipeline. The Performance plugin was installed in Jenkins's plugin manager to gather and visualize performance data on the tests executed.

Validating performance test TC7

To perform TC7, a specific JMeter test plan was created⁴⁵. The JMeter test calls the “get assets” Exploration API, which retrieves the list of all the published assets performing a SPARQL query on the RDF repository. This represents the heaviest request which can be made to the Asset Manager, since it requires receiving the request, converting it into SPARQL, storing the results from the RDF repository and then forwarding them to the caller.

Tool: Asset Manager

Input: HTTP request

Output: The results were captured as JMeter output. JConsole provided the graphs of resource usage.

Script: The tests are defined as a test plan using the JMeter tool and executed through Jenkins pipelines.

Execution TC7

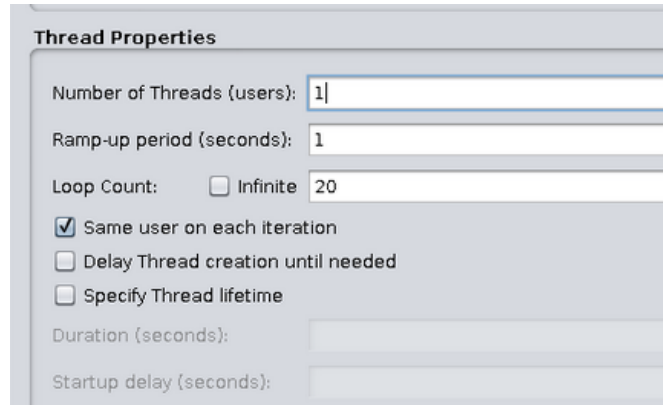
Exploration API was called in cycle of 20 repetitions in series. The configuration for testing the Asset Manager in JMeter was the following:

- Number of Threads (NoT): 1 (number of requests)
- Ramp-up period (R): 1 second
- Loop Count: 20

⁴⁴ <https://jmeter.apache.org/>.

⁴⁵ <https://jmeter.apache.org/>.

Number of iterations in which 1 threads are executed. For this performance test we used 20 iterations.



The screenshot shows the 'Thread Properties' dialog box in JMeter. The 'Number of Threads (users):' field is set to 1. The 'Ramp-up period (seconds):' field is set to 1. The 'Loop Count:' is set to 20, with the 'Infinite' checkbox unchecked. The 'Same user on each iteration' checkbox is checked. The 'Delay Thread creation until needed' and 'Specify Thread lifetime' checkboxes are unchecked. The 'Duration (seconds):' and 'Startup delay (seconds):' fields are empty.

Figure 64 TC7 JMeter test plan properties

This configuration guarantees that requests were not overlapping. The tests based on JMeter test plans (jmx files) were executed through Jenkins pipelines. The results were captured into *jtl* files.

Analysis of results TC7

The data obtained from the JMeter tool (*jtl* files) were analysed. Although 20 repetitions of the Asset Manager HTTP call were performed, we counted the average times considering the last ten requests. Figure 65 and

Table 28 show the response times as reported by JMeter.

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	09:50:11.441	Thread Group 1-1	Explore	566	✓	92275	435	516	91
2	09:50:12.007	Thread Group 1-1	Explore	524	✓	92275	435	475	0
3	09:50:12.531	Thread Group 1-1	Explore	425	✓	92275	435	400	0
4	09:50:12.956	Thread Group 1-1	Explore	436	✓	92275	435	412	0
5	09:50:13.392	Thread Group 1-1	Explore	403	✓	92275	435	379	0
6	09:50:13.796	Thread Group 1-1	Explore	495	✓	92275	435	470	0
7	09:50:14.291	Thread Group 1-1	Explore	441	✓	92275	435	417	0
8	09:50:14.733	Thread Group 1-1	Explore	461	✓	92275	435	456	0
9	09:50:15.214	Thread Group 1-1	Explore	501	✓	92275	435	476	0
10	09:50:15.715	Thread Group 1-1	Explore	436	✓	92275	435	411	0
11	09:50:16.151	Thread Group 1-1	Explore	416	✓	92275	435	391	0
12	09:50:16.567	Thread Group 1-1	Explore	428	✓	92275	435	404	0
13	09:50:16.996	Thread Group 1-1	Explore	485	✓	92275	435	460	0
14	09:50:17.481	Thread Group 1-1	Explore	454	✓	92275	435	429	0
15	09:50:17.935	Thread Group 1-1	Explore	496	✓	92275	435	471	0
16	09:50:18.431	Thread Group 1-1	Explore	417	✓	92275	435	392	0
17	09:50:18.848	Thread Group 1-1	Explore	441	✓	92275	435	416	0
18	09:50:19.289	Thread Group 1-1	Explore	478	✓	92275	435	453	0
19	09:50:19.767	Thread Group 1-1	Explore	437	✓	92275	435	413	0
20	09:50:20.205	Thread Group 1-1	Explore	483	✓	92275	435	459	0

Figure 65 TC7 Asset Manager response times from JMeter

Table 28 TC7 Asset Manager average response time

Msg. order	Total response time [ms]
1	416
2	428
3	485
4	454
5	496
6	417
7	441
8	478
9	437
10	483
Average	453,5

3.4.2 TC8 - Scalability Testing for Asset manager

Table 29 TC8 Scalability Testing for Asset manager

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-2. 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.
Preparation	A deployed instance of the Asset Manager populated with assets. Jenkins with the Performance plugin. JMeter tool.
Testing Environment (Setup)	Asset Manager deployment: CentOS Linux 7, Intel® Xeon 8-core CPU D-1521 @ 2.40GHz and 64 GB Memory, Docker v19.01.13, docker-compose v1.18.0 Jenkins and Jmeter running on a separated machine with the following specifications: Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz, 4 core; 12 GB RAM; CentOS Linux release 7.8.2003 (Core), Jenkins v2.235.2 and the Performance plugin v3.18, Apache JMeter v5.3.
Testing Scripts	Different JMeter test plans are defined to perform an increasing number of concurrent requests (load test) to Asset Manager.
Execution	The JMeter test plans are executed through Jenkins and results are gathered and visualized using the Performance plugin.
Analysing Results	The trend of response times over time, the average conversion time (with related statistics) and the number of performed/failed requests are collected considering the different test configurations.

Preparation of the test environment and Setup TC8

The preparation of the test environment is similar to the one same described for TC7. The only difference is the usage of the Performance Plugin in Jenkins to analyse data collected.

In addition to the minimum testing setup described, we monitored containers of the Asset Manager in execution using the Telegraf⁴⁶ agent to gather data from the Docker Daemon, InfluxDB⁴⁷ to store time-series and Grafana⁴⁸ to plot them.

Validating performance test TC8

The test itself consists of a single Thread Group which performs calls to the “get_assets” Exploration API. Such API is used by the “Explore” page in the Asset Manager Publisher and returns the list of assets whose status is “published”.

Tool: Asset Manager

Input: HTTP request

Output: The responses provided by the Asset Manager to the requests made. JTL files containing results of the load test execution.

Script: The tests are defined as test plans using the JMeter tool and executed through Jenkins pipelines.

Execution TC8

The test plans in JMeter define a *Thread Group* performing HTTP requests to the Asset Manager endpoint and several *Listeners* to collect performance data. The parameters configured for the Thread Group are:

- *Number of Threads (NoT):* **10, 50, 100, 150, 200, 500, 1000, 2500, 5000** (number of concurrent requests). Each thread executes the action defined in its entirety and completely independently of other test threads⁴⁹.
- *Ramp-up period (R):* **1 second**. The ramp-up period tells JMeter how long to take to "ramp-up" to the full number of threads chosen. Each thread starts *R/NoT* seconds after the start of the previous thread⁵⁰. For example, if 100 threads are executed with a 1s ramp-up period, it means that one thread is started every 10ms.
- *Loop Count:* **1**. Number of iterations in which N threads are executed. For scalability test, we use only one iteration.

The tests based on JMeter description (jmx files) were executed from Jenkins's. The results were captured into jtl files.

⁴⁶ <https://www.influxdata.com/time-series-platform/telegraf/>

⁴⁷ <https://www.influxdata.com/products/influxdb-overview/>

⁴⁸ <https://grafana.com/>

⁴⁹ https://jmeter.apache.org/usermanual/test_plan.html

⁵⁰ https://jmeter.apache.org/usermanual/test_plan.html

Analysis of results TC8

The data obtained from the JMeter tool (*jtl* files) were analysed after the execution of each configuration using the Performance Plugin of Jenkins. Core results obtained are reported in this section. Section 7.1 in Annex B contains complete reports on the trend of response times over time, the average conversion time (with related statistics) and the number of performed/failed requests considering the different test configurations.

As shown in Table 30, the Asset Manager was able to successfully answer to more than 200 requests in parallel. This means that over 200 users can view the “Explore” page at the same time on the Asset Manager Publisher application.

Table 30 Asset Manager TC8: average response time under increasing load

Number of HTTP requests (<i>N</i>)	Avg Time of processing <i>N</i> HTTP Requests [ms]
10	819
50	2 874
100	5 685
150	8 999
200	11 297
500	Not completely processed

When looking at the CPU and RAM usage during the test (shown in Figure 65 and Figure 66), we can see that the most stressed component is the CMS (Django), and that the failures in answering to more than 200 parallel requests are caused by high CPU usage (over 350%). This behavior is caused by the tight integration between Django and Blazegraph, which hampers caching. Since there is no way for the caller (Django) to know when the content of the RDF repository has changed, each time a new request comes, it is sent to the SPARQL endpoint of the RDF repository and the result is parsed and converted into JSON-LD according to a specific JSON LD Frame. This task of parsing the result into RDF and converting it in JSON requires creating a potentially big JSON document in memory, therefore requiring high CPU usage. We can nonetheless notice that such results are related to a deployment where all the components of the Asset Manager run on the same single server, and that 200 users accessing the Publishing application of the Asset Manager at the same time to perform a search would be quite a high number. According to the UIC list of operators in Europe⁵¹, in 2020 there were 281 operators, and so 200 users actively using the Asset Manager at the same time would mean that nearly a representative of each passenger rail operator in Europe would need to connect to the Asset Manager before starting to obtain errors due to high load.

⁵¹ <https://uic.org/support-activities/it/rics>

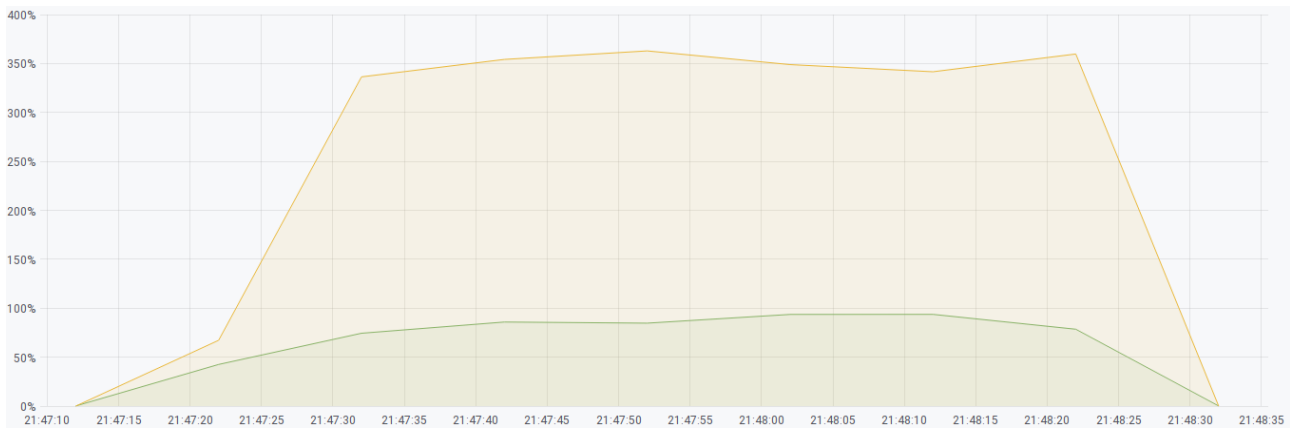


Figure 66 Asset Manager TC8 CPU usage during scalability test (yellow=Django, green=Blazegraph)

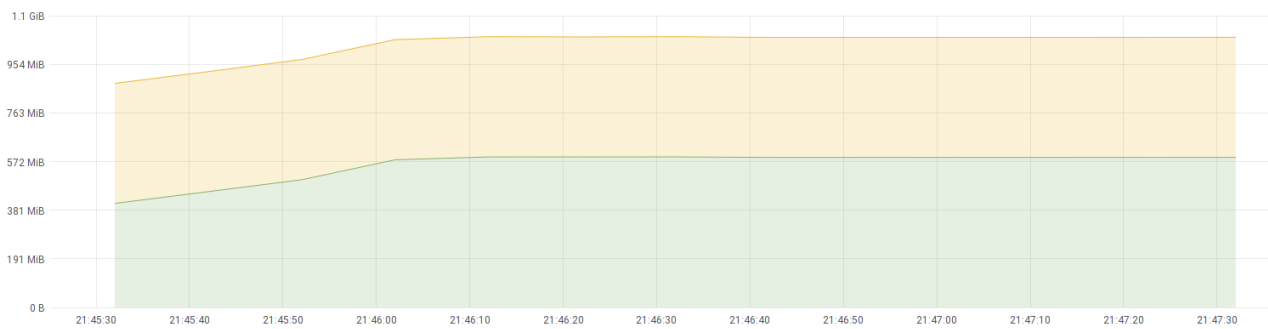


Figure 67 Asset Manager TC8 RAM usage during scalability test (yellow=Django, green=Blazegraph)

3.5 CONVERTER

To test performance and scalability of the IF Converter, we defined test cases considering the two main scenarios identified for this component: *batch data conversion* and *runtime data/message conversion*. Moreover, we defined a further test case to evaluate the scalability of the deployment solution chosen for the Converter.

As showcased in D5.3, performances of a conversion procedure not only depend on the software artifact implementing it, but also 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 in the lifting, the complexity of the queries in the lowering, etc.) and the size/density of the intermediate knowledge graph. For this reason, to obtain comparable results between different Converters (or versions of the same Converter), it is important to consider the same pipeline defined between the same formats, with the same input data and the same mappings. For the F-Rel validation, we considered the test cases defined in D3.4 designing accordingly the performance and scalability evaluation of the Converter. Moreover, we took into account the tests performed for C-Rel (D5.3) to test comparable pipelines and investigate differences in the results obtained for the final release.

For the batch data conversion scenario, we considered the GTFS-Madrid-Bench datasets and mappings as described in D5.3. In particular, we focused on the three most common format, i.e., CSV, JSON and XML. As done for C-Rel, we considered the *declarative approach* based on RML⁵² and Apache Velocity⁵³ templates for the conversion of batch data.

For the runtime data/message conversion scenario, we considered the tests designed in the ST4RT project for the conversion FSM/918 to test the final release of the *annotation approach* implemented in the F-Rel Converter. Moreover, we defined a pipeline converting a response message of the HaCon VBB endpoint to TRIAS to test how the *declarative approach*, based on RML and Velocity templates, performs in this scenario (pipeline and messages described in Section 2.7).

⁵² <https://rml.io/>

⁵³ <https://velocity.apache.org/>

3.5.1 TC9 - Performance Testing for Batch Data Conversion

The T9 performance test evaluates the Converter in the batch data conversion scenario considering the average expected workload.

Table 31 TC9 Performance Testing for Batch Data Conversion

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.
Preparation	Docker, docker-compose, Converter JAR, Unix shell
Testing Environment (setup)	The Converter JAR, the Dockerfile and docker-compose file for the Converter, different pipeline configurations, and the GTFS-Madrid-Bench input datasets are loaded into a machine with the following characteristics: CentOS Linux 7, Intel® Xeon 8-core CPU D-1521 @ 2.40GHz and 64 GB Memory, Docker v19.01.13, docker-compose v1.18.0
Testing Scripts	Bash script building the Docker Image using the Jar and the Dockerfile, and launching the container with different configurations.
Execution	The script is deployed and executed. Each test configuration is executed 5 times.
Analysing results	The average lifting time, lowering time, conversion time and number of triples in the knowledge graph are collected considering the 5 executions of the test configuration from the Converter. CPU usage and memory consumption of the container are monitored.

Preparation of the test environment and Setup TC9

The Docker image of the Converter is built to run the JAR using a Java 11 capable base image (openjdk:11-jdk on Dockerhub⁵⁴). The JAR file of the Converter is configured to access resources from the filesystem. A parametric *docker-compose* file is used to mount different input files and configurations in the Converter container filesystem before launching it. As shown in Figure 68, environment variables are used to specify:

- The *Camel Context* file, i.e., the conversion pipeline and the related configuration to be executed
- The lifting and lowering mappings to be used for the conversion
- The input dataset to be converted
- The output data and additional data gathered are mounted to a host folder to guarantee persistence after the container has been stopped.

The hardware, software specification of the test environment is described in Table 31 (Setup). For the *batch* tests of the Converter it was decided to employ the same machine used for the C-Rel testing activities to obtain comparable results. Memory constraint is set to 24GB using the Docker – `memory-limit` option on containers in execution, no limits are set on CPU usage. A timeout of 24 hours is set for each conversion executed.

Performance data are retrieved using UNIX-provided statistics on running processes and collecting timestamps through the instrumentation of the Chimera pipeline. In addition to the minimum testing setup described, we monitored containers in execution using the Telegraf⁵⁵ agent to gather data from the Docker Daemon, InfluxDB⁵⁶ to store time-series and Grafana⁵⁷ to plot them.

```
version: '2.2'

services:
  chimera-converter:
    image: chimera/batch-converter
    container_name: ${TEST_ID}
    mem_limit: 24g
    volumes:
      - ./src/main/resources/routes/${ROUTE_ID:-camel-context}.xml:/home/routes/camel-context.xml
      - ./src/main/resources/lifting/${MAPPINGS:-gtfs-csv}.rml.ttl:/home/lifting/gtfs-mappings.rml.ttl
      - ./src/main/resources/lowering:/home/lowering
      - ./inbox/${TEST_ID}.zip:/home/inbox/${TEST_ID}.zip
      - ./outbox/${ROUTE_ID}-${TEST_ID}-${MAPPINGS}/${REP_ID:-1}:/home/out
```

Figure 68 Parametric docker-compose file for the Chimera Converter.

⁵⁴ <https://hub.docker.com/>

⁵⁵ <https://www.influxdata.com/time-series-platform/telegraf/>

⁵⁶ <https://www.influxdata.com/products/influxdb-overview/>

⁵⁷ <https://grafana.com/>

Validating performance test TC9

To validate the batch conversion performance test case, we leveraged the pipeline already specified and tested for C-Rel and relying on the datasets from the GTFS-Madrid-Bench. The pipeline implements a roundtrip conversion GTFS-Linked GTFS-GTFS with Chimera, using materialization lifting via RML mappings and Apache Velocity template for the lowering over materialized RDF.

In the implemented pipeline, a GTFS feed is read from the filesystem, the feed is unzipped, lifting RML mappings are executed to materialize the RDF graph, a set of templates (one for each GTFS file) is executed in parallel to populate the resulting files querying the RDF graph, results of the lowering phase are written to the filesystem.

For performance tests we considered as average expected workload the datasets of the GTFS-Madrid-Bench corresponding to size 1, i.e., the original data provided by the Consorcio Regional de Transporte de Madrid (CRTM) and converted from CSV also in the JSON and XML formats.

As a result of the C-Rel validation, the main bottleneck in the performance of the Converter was related to the lifting portion, in particular with respect to XML input data sources. As described in D5.5, in the final release of the Converter we tried to address these issues. The tests performed for TC9 analyse the different strategies implemented to introduce concurrency in the lifting procedure, and the performances of the new parsers for JSON and XML.

Tool: Converter built using the Chimera framework. RMLProcessor used for lifting, TemplateProcessor for the lowering.

Input: 1-csv, 1-json and 1-xml GTFS feeds from the GTFS-Madrid-Bench

Output: A CSV GTFS feed containing the same information of the original source. A dump of the intermediate knowledge graph built during the conversion process.

Script: The script developed for this test is based on a bash script. The script sets for each configuration the environment variables and launches the Converter using the provided parametric docker-compose. As documented for other test cases it is possible to automatize tests in Jenkins using a bash script. Moreover, environmental variables can be set in the pipeline⁵⁸ to define the configuration of the different tests.

Execution TC9

The tested configurations are:

- **frel-0:** final release of the Converter without concurrency options (cf. D5.5) and using an in-memory repository;
- **frel-1:** as frel-0, plus Concurrent Executor (CR);

⁵⁸ <https://www.jenkins.io/doc/pipeline/tour/environment/>

- **frel-2**: as *frel-0*, plus Concurrent Executor (CR), concurrency in the lifting process on triple maps (LPC-TM), using a single records factory to access data from the sources (SRF);
- **frel-3**: as *frel-0*, plus concurrency in the lifting process on triple maps (LPC-TM), using a single records factory to access data from the sources (SRF);
- **frel-4**: as *frel-0*, plus Concurrent Executor (CR), concurrency in the lifting process on logical sources (LPC-LS), using a single records factory to access data from the sources (SRF);
- **frel-5**: as *frel-0*, plus Concurrent Executor (CR), concurrency in the lifting process on triple maps (LPC-TM), using a single records factory to access data from the sources (SRF).

The different configurations were run for the 1-csv dataset using both the mappings of the GTFS-Madrid-Bench and the optimized mappings (avoiding *joins* between Triple Maps) developed for C-Rel. Finally, the configuration performing better was tested also for the 1-json and 1-xml datasets considering both typologies of mappings.

Each test was executed 5 times.

Analysis of the results TC9

Values reported in this section are computed as averages on the data of the 5 executions performed for each configuration. The results are compared with the ones of C-Rel presented in D5.3 (rows filled with grey). TO stands for timeout (exceeding the 24 hours timeout set), OM stands for out-of-memory (exceeding the 24GB memory limit).

Table 32 reports the tests performed with the 1-csv dataset for each possible configuration using the GTFS-Madrid-Bench lifting mappings (565,489 materialized triples). As described in D5.5, the *TemplateProcessor* showcased good performances already in the C-Rel implementation and for F-Rel we focused on improving performances of the lifting portion. Therefore, as expected, the lowering time is similar to the one obtained for C-Rel and almost constant for every configuration.

The obtained statistics showcase a consistent performance improvement of *frel-0* (not introducing concurrency) with respect to *crel*. Despite similar memory consumption and CPU usage, *frel-0* results in a lower conversion time, mainly due to improvements in the time required for the lifting portion of the pipeline. The obtained improvements can be explained considering the refactoring implemented in updating Chimera from Camel 2.x to Camel 3.x, and the simplified mechanism to write triples generated during the lifting procedure to the graph used in the conversion pipeline. The introduction of the different concurrency strategies lowers, even more, the time required for conversion. However, the introduction of multiple threads naturally increases CPU usage and memory consumption. Conversion times considering the different concurrency strategies are similar, however, *frel-4* registered the best execution time.

Table 32 Tests for TC9 with the 1-csv dataset

	CR	LPC	SRF	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
1-csv-crel	-	-	-	22.77	20.95	1.82	3.6	251.05
1-csv-frel-0				15.77	13.89	1.87	3.38	269.78
1-csv-frel-1	X			12.31	10.42	1.89	4.19	380.58
1-csv-frel-2	X	TM	X	11.25	9.38	1.87	4.37	426.93
1-csv-frel-3	X	TM		11.21	9.43	1.78	4.49	424.84
1-csv-frel-4	X	LS	X	10.83	8.98	1.85	4.56	436.33
1-csv-frel-5		TM		11.52	9.64	1.88	5.19	370.54

Table 33 reports the tests performed with the 1-csv dataset for each possible configuration using the optimized lifting RML mappings with no *join* conditions defined for C-Rel (326,538 materialized triples but consistent output after lowering). The results obtained confirm the trends observed for the GTFS-Madrid-Bench mappings: the F-Rel Converter performance outweighs the ones of the C-Rel version, and the *frel-4* configuration is the one performing better in terms of *conversion time*.

Analysing more in details the results reported in Table 32 and Table 33 we can draw additional conclusions on the performances of the different concurrency strategies. Using the concurrent executor option to process in parallel each record provides a considerable performance advantage (cf. *frel-0* and *frel-1*). Moreover, in the case analysed, the concurrency in the lifting process on logical sources performs better with respect the one on triple maps. That is probably because in the second case different threads try to concurrently access the same input source that acts as a bottleneck. However, it is important to point out that different RML mappings may result in different performance considerations, for example, considering how many triple maps are defined for each logical source in the mapping file.

Finally, the usage of a single records factory (shared among the different threads) to process and cache the records already extracted from the input sources does not seem to introduce delays due to concurrent access. Therefore, it is useful to adopt this option to exploit caches and avoid duplication of data structures in memory.

Table 33 Tests for TC9 with the 1-csv dataset and optimized lifting mappings

	CR	LPC	SRF	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
1-csv-o-crel	-	-	-	12.03	10.35	1.69	1.99	298.09
1-csv-o-frel-0				8.49	6.78	1.71	1.74	8.49
1-csv-o-frel-1	X			7.8	6	1.79	1.56	394.52
1-csv-o-frel-2	X	TM	X	7.58	5.71	1.86	1.7	434.58
1-csv-o-frel-3	X	TM		7.58	5.65	1.93	1.71	434.5
1-csv-o-frel-4	X	LS	X	7.54	5.69	1.85	1.85	428.03
1-csv-o-frel-5		TM		7.58	5.81	1.78	1.72	357.02

Table 34 compares the tests performed with the 1-csv, 1-json and 1-xml datasets considering the *crel*, *frel-0* and *frel-4* configurations with the GTFS-Madrid-Bench mappings. The F-Rel Converter outperforms the C-Rel one for each format. Notably, the newly introduced parser for XML had the effect of drastically improve the performances of the conversion. For C-Rel it was not possible to complete the conversion within the 24 hours timeout for the 1-xml dataset with GTFS-Madrid-Bench mappings, meanwhile, the F-Rel Converter completed it performing even better than on the JSON dataset. The conversion of the JSON dataset is the one registering the more limited improvements, this is mainly because the actual bottleneck is related to the JSON parser library and not to the actual processing to materialize the triples. Indeed, JSON is the only format not taking advantage of the concurrency (*1-json-frel-0* performs better than *1-json-frel-4*) probably because concurrent threads queue waiting to get access records from the slow JSON parser. We tested also *frel-1*, *frel-2*, *frel-3*, and *frel-5* for the 1-json dataset but *frel-0* resulted as the best configuration using the GTFS-Madrid-Bench mappings.

Table 34 Tests for TC9 with 1-csv, 1-json and 1-xml dataset

	CR	LPC	SRF	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
1-csv-crel	-	-	-	22.77	20.95	1.82	3.6	251.05
1-csv-frel-0				15.77	13.89	1.87	3.38	269.78
1-csv-frel-4	X	LS	X	10.83	8.98	1.85	4.56	436.33
1-json-crel	-	-	-	50.41	48.34	2.07	5.53	189.23
1-json-frel-0				30.89	28.96	1.93	6.23	217.59
1-json-frel-4	X	LS	X	35.17	33.42	1.75	5.31	328.22
1-xml-crel	-	-	-	TO	-	-	-	-
1-xml-frel-0				31.04	29.15	1.89	5.43	211.30
1-xml-frel-4	X	LS	X	16.26	14.43	1.83	13.21	455.28

Table 35 reports the tests performed with the 1-csv, 1-json and 1-xml datasets considering the *crel*, *frel-0* and *frel-4* configurations with the optimized lifting RML mappings with no *join* conditions defined for C-Rel. These tests confirm the performance enhancements of the F-Rel version and let us quantify the improvement on the XML case (three orders of magnitude) with respect to the C-Rel implementation. Using the optimized mappings with no *join* conditions, i.e. with fewer accesses to the logical sources, also for JSON *frel-4* performed better than *frel-0*.

Table 35 Tests for TC9 with 1-csv, 1-json and 1-xml dataset with optimized lifting mappings

	CR	LPC	SRF	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
1-csv-o-crel	-	-	-	12.03	10.35	1.69	1.99	298.09
1-csv-o-frel-0				8.50	6.77	1.73	1.64	321.72
1-csv-o-frel-4	X	LS	X	7.54	5.69	1.85	1.85	428.03
1-json-o-crel	-	-	-	17.39	15.53	1.86	3.39	276.16
1-json-o-frel-0				11.38	9.59	1.79	2.43	308.36
1-json-o-frel-4	X	LS	X	10.24	8.55	1.7	2.95	414.02
1-xml-o-crel	-	-	-	35279.34	35277.59	1.76	6.81	118.49
1-xml-o-frel-0				11.60	9.75	1.85	3.31	301.06
1-xml-o-frel-4	X	LS	X	8.79	6.86	1.93	4.11	446.23

3.5.2 TC10 - Performance Testing for Runtime Data/Message Conversion

The TC10 performance test evaluates the Converter adopting the annotation and declarative approaches in the runtime data/message conversion scenario and considering the average expected workload.

Table 36 TC10 Performance Testing for Runtime Data/Message Conversion

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.
Preparation	The Converter JARs with different pipeline configurations exposing an endpoint for message conversion. The input messages (ST4RT test cases, VBB endpoint example response). Unix shell
Testing Environment (setup)	Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz, 4 core; 12 GB RAM; CentOS Linux release 7.8.2003 (Core), Java (Java 8 for annotations approach, Java 11 for the declarative approach).
Testing Scripts	For each configuration, the related Converter JAR is executed. A bash script is defined to perform requests at regular interval to the running Converter using the input messages provided.
Execution	The script is deployed and executed. Each test configuration executes 20 requests to the converter.
Analysing Results	The average lifting time, lowering time, conversion time and number of triples in the knowledge graph are collected considering the different test configurations. CPU usage and memory consumption of the running Java processes are monitored.

Preparation of the test environment and Setup TC10

The different converters are built as executable JARs using Maven⁵⁹ to test the different pipelines and configurations. Each Converter is configured to expose an endpoint accepting requests for message conversion. To reduce the possible overhead introduced by additional components deployed in the testing environment, it was decided to use a simple bash script to run the performance tests.

The hardware and software specifications of the test environment are described in Table 36 (Setup). Performance data about lifting, lowering and execution time for a request are retrieved collecting timestamps through the instrumentation of the converters' pipeline.

The `curl` tool, available by default in many operating systems, was selected to perform the HTTP requests to the Converter endpoint. The `JConsole` tool, provided within the Java JDK, was selected to monitor CPU and memory usage of the running Converter. For these reasons, there was no need to install extra tools.

Validating performance test TC10

For the runtime data/message conversion scenario, the testing activities were performed considering the two different approaches developed within SPRINT.

For the *annotation approach*, we considered the Converter blocks developing further the ST4RT converter and using Java Annotations for the lifting and the lowering. As done for C-Rel, we considered the messages designed for the ST4RT project testing activities from FSM to 918 and viceversa [1].

For the *declarative approach*, we considered the blocks adopting declarative rules for the mappings: materialization lifting via RML mappings and Apache Velocity templates for the lowering. To test this approach in the runtime data/message conversion scenario, we exploited the pipeline described in Section 2.7 and converting an example message obtained from the HaCon VBB endpoint to TRIAS.

Tool: Converter built using the Chimera framework.

Annotation approach: *ST4RTLiftingProcessor* and *ST4RTLoweringProcessor*

Declarative approach: *RMLProcessor* for the lifting, *TemplateProcessor* for the lowering

Input

Annotation approach

The same test messages defined in the ST4RT project were executed for the annotation approach evaluation. All the 5 types of FSM requests (`Booking_PreBookRequest`) were considered as input for the FSM-to-918 request conversion. Similarly, all the 5 types of 918 replies (`uic_reservationbookrp`) were considered as input for the 918-to-FSM response

⁵⁹ <https://maven.apache.org/>

conversion. The messages differ in several attributes (like *Accommodation type*, *Class* or *Number of passengers*) to test different aspects of the mappings defined.

Declarative approach

An example message obtained from the HaCon VBB endpoint, like the one shown in Figure 30, is used as input for the conversion process.

Output: The responses provided by the Converter to the requests made were recorded: the resulting 918 request (*uic_reservationbookrq*) and FSM response (*Booking_PreBookResponse*) for the *annotation approach*, and the resulting TRIAS message for the *declarative approach* (as the one shown in Figure 31).

Script: The script developed for this test is based on a bash script. The script launches the Converter JAR for a given configuration and uses the command `curl` with necessary options to perform the requests.

Run the converter

Converter JARs are executed by command:

```
java -jar converter-*.jar
```

where `[*]` stands different postfix of the converters.

Perform the requests

The following command is used within a loop to perform the HTTP POST requests to the Converter:

```
curl --header 'Content-Type: application/xml' --request POST --  
data-binary 'message.xml' -o ./conversion-result.xml  
http://localhost:<port>/<path>
```

The first option (`--data-binary`) is the input file (FSM request, 918 reply or VBB message to be converted)

The second file after the option (`-o`) is the output file (converted 918 request, FSM response or TRIAS message)

The last parameter is the endpoint exposed by the Converter:

- Annotation approach, FSM-to-918 Converter
<http://localhost:7080/conversion/fsm>
- Annotation approach, 918-to-FSM Converter
<http://localhost:7080/conversion/918reply>
- Declarative approach, VBB-to-TRIAS
<http://localhost:7080/conversion/vbb/trias>

Execution TC10

Before executing the performance tests, we performed pre-testing sessions to ensure the correctness of the configurations. Every configuration was run using the defined scripts and sending to the corresponding converter 20 requests, each one executed upon the completion of the previous. We considered the second subset of requests (11th to 20th) to obtain average performance data in usual conditions over 10 requests. However, for each Converter JAR, it is reported also the result obtained for the 1st request to measure the impact of cold start on the Converter performances.

Annotation approach

For the *annotation approach*, we run 10 test configurations using; (i) the same Converter JAR (*ST4RTLiftingProcessor* and *ST4RTLoweringProcessor*), (ii) the 5 types of FSM requests and 5 types of 918 replies from the ST4RT project.

Declarative approach

For the *declarative approach*, the same example message from the VBB endpoint was used with different configurations of the pipeline. Despite providing different options each pipeline was configured to use an in-memory repository, the *RMLProcessor* block for the lifting, and the *TemplateProcessor* block for the lowering.

The tested configurations are:

- **crel-m**: core release of the Converter without concurrency, with the default XML parser and without optimizations in the *TemplateProcessor* block (cf. D5.5);
- **fre1-0-m**: final release of the Converter with new XML parser implementation (XP);
- **fre1-1-m**: final release of the Converter with new XML parser implementation (XP), Concurrent Executor and optimizations in the *TemplateProcessor* block (OTP);
- **fre1-2-m**: final release of the Converter with new XML parser implementation (XP), Concurrent Executor (CR), concurrency in the lifting process on triple maps (LPC-LS), using a single records factory to access data from the sources (SRF) and with optimizations in the *TemplateProcessor* block (OTP).

Analysis of results TC10

The data obtained through pipeline instrumentation during the execution were processed to obtain lifting, lowering and execution times. Then, average times were computed considering the requests performed. The JConsole report was analysed after the execution of each configuration (considering the 20 requests) to extract the max Memory and CPU usage values.

Core results obtained are reported in this section. Section 7.2 in Annex B contains the detailed data for each request and the complete JConsole report for each configuration.

Annotation approach

Table 37 reports the results of the tests performed for each configuration considering the *annotation approach*. It is possible to notice how the conversion time changes with the two pipelines but also

considering the different types of messages. In particular, the execution times obtained converting 918 replies to FSM are higher than the ones obtained to convert FSM requests to 918 response. This is due to the fact that in the *annotation approach* the lowering part is more demanding because the SPARQL queries defined to access the knowledge graph materialized during the lifting are not optimized for the specific conversion from one format to another, but generic enough to support the annotation mechanism. Therefore, since FSM is more complex than 918, the lowering time in the pipeline converting triples to FSM is higher than the one in the pipeline lowering triples to 918.

Related to this aspect, it is important to point out that in Table 37 the number of triples reported only considers the result of the lifting procedure but the knowledge graph used for lowering comprises more than 20 thousands triples. As detailed in the ST4RT project, the conversion pipeline defined enriches that triples with a set of master data (e.g., code lists) to obtain the needed information for the lowering. Finally, considering the usage of resources, we can notice that memory consumption is not negligible given the size of messages processed.

Table 37 TC10 Results of tests for the annotation approach

		Conversion time (ms)	Lifting time (ms)	Lowering time (ms)	Triples	Max Mem (GB)	Max CPU Usage (%)
Type 1	FSM-to-918 request	161.4	14.3	147.1	75	0.5	65
	918-to-FSM response	1286.3	4.8	1281.5	58	1.2	45
Type 2	FSM-to-918 request	94.1	8.3	85.8	75	1.3	30
	918-to-FSM response	1187.3	4.2	1183.1	58	1.6	40
Type 3	FSM-to-918 request	92.4	7.8	84.6	75	2.0	40
	918-to-FSM response	1306.6	3.6	1303.3	58	2.2	40
Type 4	FSM-to-918 request	101	8.1	92.9	75	2.5	25
	918-to-FSM response	1187.5	3.7	1183.8	58	2.5	40
Type 5	FSM-to-918 request	95.3	11.3	84	79	2.8	55
	918-to-FSM response	1941.9	4.3	1937.6	67	2.9	60

Table 38 compares the results of the tests performed for the first configuration (Type 1 – FSM-to-918 request) considering the *annotation approach* and a cold start scenario (first request after launching the Converter JAR). As expected, the time obtained for the cold start is higher than the average because of the time required for the initialization of objects associated with the conversion pipeline.

Table 38 TC10 Cold start results for the annotation approach

Type 1 – FSM-to-918 request	Conversion time (ms)	Lifting time (ms)	Lowering time (ms)
Average	1176	626	550
Cold start	161.4	14.3	147.1

Table 39 compares the results obtained in SPRINT (C-Rel and F-Rel) with the ones of the ST4RT project. Comparing the results, it is evident the improvement obtained in performances for the annotation-based converter. The request average processing time in the SPRINT project is almost 4 seconds faster than the average processing time of the request in the ST4RT project. The response average processing time in the SPRINT project is almost 3,5 seconds faster than the average processing time of response in the ST4RT project. The results also showcase how the F-Rel developments in SPRINT, introducing concurrency in the annotation-based blocks of the Converter, significantly improved performances.

Table 39 TC10 Comparison ST4RT and SPRINT

SPRINT F-Rel	Type 1	Type 2	Type 3	Type 4	Type 5
Avg request time [ms]	161,4	94,1	92,4	101	95,3
Avg response time [ms]	1286,3	1187,3	1306,9	1187,5	1941,9
SPRINT C-Rel	Type 1	Type 2	Type 3	Type 4	Type 5
Avg request time [ms]	980	1000	910	930	1030
Avg response time [ms]	2250	2290	2250	2110	2260
ST4RT/Message type	Type 1	Type 2	Type 3	Type 4	Type 5
Avg request time [ms]	4180,17	3994,29	3840,05	4623,20	4255,17
Avg response time [ms]	4901,13	4896,45	4969,34	4802,70	5097,23

Declarative approach

Table 40 reports the results of the tests performed for each configuration considering the *declarative approach* (466 materialized triples). It is possible to notice that *frel-m-1* is the configuration performing better with conversion times in the order of 100ms and, thus, introducing a perfectly acceptable overhead in a runtime scenario. The results obtained for *frel-m-0* showcase how the introduction of the new parser for XML, as noticed also for TC9, is responsible for a consistent improvement in conversion time. The results for *frel-m-1* demonstrate the impact of concurrency and additional optimizations implemented in F-Rel. The results for *frel-m-2*, having a similar configuration to *frel-4* defined in TC9 (performing best for batch messages), demonstrate how for small messages is preferable not to introduce excessive concurrency since the initialization of structures (e.g., threads) to handle it are not compensated by the obtained speedup. Finally, it is important to point

out the very small usage of resources of the Converter, especially memory, in the different configurations tested.

Table 40 Tests for TC10 for the Declarative approach

	Configuration		Conversion time (ms)	Lifting time (ms)	Lowering time (ms)	Max Mem (GB)	Max CPU Usage (%)
crel-m		Average	739	711	28	0.09	40
		Cold start	2305	1979	326	-	-
frel-m-0	XP	Average	166	137	30	0.04	30
		Cold start	2228	1884	344	-	-
frel-m-1	XP/CR/OTP	Average	138	107	31	0.04	25
		Cold start	2232	1881	351	-	-
frel-m-2	XP/CR/OTP/ LPC-LS/SRF	Average	166	125	41	0.04	30
		Cold start	1883	1491	392	-	-

3.5.3 TC11 - Scalability Testing for Batch Data Conversion

The TC11 scalability test evaluates the Converter in the batch data conversion scenario considering different increasing workloads.

Table 41 TC11 Scalability Testing for Batch Data 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.
Preparation	Docker, docker-compose, Converter JAR, GraphDB Docker image, Unix shell
Testing Environment (setup)	The Converter JAR, the Converter Dockerfile, the docker-compose file for the Converter and GraphDB (v9.0.0-Free), different pipeline configurations, and the GTFS-Madrid-Bench input datasets are loaded into a machine with the following characteristics: CentOS Linux 7, Intel® Xeon 8-core CPU D-1521 @ 2.40GHz and 64 GB Memory, Docker v19.01.13, docker-compose v1.18.0
Testing Scripts	Bash script building the Docker Image using the Jar and the Dockerfile, and launching the Converter container with different configurations. The script can be launched with an option to initialize the GraphDB container before executing the Converter.
Execution	The script is deployed and executed. Each test configuration is executed 5 times.
Analysing results	The average lifting time, lowering time, conversion time and number of triples in the knowledge graph are collected considering

	the 5 executions of the test configuration from the Converter. CPU usage and memory consumption of the container are monitored.
--	---

Preparation of the test environment and Setup TC11

The preparation of the test environment is similar to the one same described for TC9. The only difference is the modification of the docker-compose to launch also a Docker container running the GraphDB (Free version 9.0.0) Triplestore in the same network.

Validating scalability test TC11

To validate the batch conversion scalability test case, we leveraged the pipeline already specified and tested for C-Rel and relying on the datasets from the GTFS-Madrid-Bench as described for TC9.

For the scalability tests, we considered a workload of increasing size leveraging the datasets of the GTFS-Madrid-Bench corresponding to size 5-10-50-100-500 and available in CSV, JSON and XML formats.

As a result of the C-Rel validation, the main bottleneck in the performance of the Converter was related to the lifting portion. In addition to the work done to reduce the conversion time (commented in TC9), to improve the scalability of the Converter the F-Rel version implemented the possibility of leveraging an external repository to minimize the memory consumption (cf. D5.5). To test this option we validated the Converter using the GraphDB Triplestore⁶⁰. We selected GraphDB because it is the Triplestore solution already adopted by other IP4 projects and because it fully supports the RDF4J interface introducing performance optimizations in the reading/writing procedure (not available if a generic SPARQL endpoint is used). It is important to point out that the tests performed employed the Free version of GraphDB imposing strict performance-related restrictions (only two concurrent queries simultaneously)

Tool: Converter built using the Chimera framework. RMLProcessor used for lifting, TemplateProcessor for the lowering. GraphDB Triplestore v9.0.0 Free.

Input: 1,5,10,50,100,500 -csv, -json and -xml GTFS feeds from the GTFS-Madrid-Bench

Output: A CSV GTFS feed containing the same information of the original source. A dump of the intermediate knowledge graph built during the conversion process.

Script: The script developed for this test is based on a bash script. The script sets for each configuration the environment variables and launches the Converter using the provided parametric docker-compose. If needed, the script can be launched with an option to initialize the GraphDB container before executing the Converter.

⁶⁰ <https://graphdb.ontotext.com/>

Execution TC11

The tested configurations are the ones described for TC9, plus additional configurations related to the usage of an external repository:

- **frel-6:** as *frel-4* (CR/LPC-LS/SRF), using an external repository (ER) with incremental (W-I) writes of batch size (BS) of 100k triples;
- **frel-7:** as *frel-4* (CR/LPC-LS/SRF), using an external repository (ER) with incremental and concurrent (W-IC) writes of batch size (BS) of 100k triples;
- **frel-8:** as *frel-4* (CR/LPC-LS/SRF), using an external repository (ER) with incremental (W-I) writes of batch size (BS) of 10k triples;
- **frel-9:** as *frel-4* (CR/LPC-LS/SRF), using an external repository (ER) with incremental and concurrent (W-IC) writes of batch size (BS) of 10k triples;
- **frel-10:** as *frel-4* (CR/LPC-LS/SRF), using an external repository (ER) and performing a single write at the end of the lifting process.

All the different configurations (*frel-0* to *frel-10*) were run for the 10-csv dataset using the mappings of the GTFS-Madrid-Bench to compare performances of the different configurations with a more demanding workload. Then, the configuration that obtained the best results in TC9, was tested for every dataset considering the different scales (1,5,10,50,100,500) and the different formats (CSV, JSON, XML). Finally, the impact of a very demanding workload (50-csv dataset) was tested with different configurations (*frel-4*, *frel-6*, *frel-10*) to analyse performances of the Converter and the external repository.

Each test was executed 5 times.

Analysis of the results TC11

Values reported in the section are computed as averages on the data of the 5 executions for each test. The results are compared with the ones of C-Rel presented in D5.3 (rows filled with grey). TO stands for timeout (exceeding the 24 hours timeout set), OM stands for out-of-memory (exceeding the 24GB memory limit).

Table 42 reports the tests performed with the 10-csv dataset for each possible configuration defined in TC9 considering an in-memory repository and using the GTFS-Madrid-Bench lifting mappings (3,663,380 materialized triples). The results obtained confirm the trends showcased in TC9 with *frel-4* (concurrent executor, concurrency in the lifting process on logical sources, using a single records factory to access data from the sources) as the configuration performing better. Notably, in this case, it is more evident the difference of performances in the conversion time between *frel-0* (no concurrency) and *frel-4* (3 times faster), but also the difference of resource usage (*MEM/CPU*) of the two configurations.

The difference of performances in the conversion time among the different configurations adopting the Concurrent Executor is negligible, and from the results, it emerges how not adopting additional options for concurrency (*frel-1*) helps in lowering the memory usage while obtaining an optimal

conversion time with respect other configurations. Finally, it is important to point how *frel-0* (not adopting concurrency but implementing the F-Rel version of Chimera) registered considerably lower memory consumption with respect to the same configuration in C-Rel.

Table 42 Tests for TC11 with the 10-csv dataset using an in-memory repository

	CR	LPC	SRF	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
10-csv-crel	-	-	-	544.41	536.9	7.51	9.47	140.82
10-csv-frel-0				487.86	480.31	7.56	6.9	127.77
10-csv-frel-1	X			157.95	150.38	7.57	10.56	394.05
10-csv-o-frel-2	X	TM	X	157.83	150.63	7.2	16.63	456.62
10-csv-o-frel-3	X	TM		158.2	151.03	7.17	17.17	459.95
10-csv-o-frel-4	X	LS	X	154.13	146.9	7.22	17.13	457.51
10-csv-o-frel-5		TM		232.62	225.17	7.44	17.1	288.51

Table 43 reports the tests performed with the 10-csv dataset for each possible configuration defined considering an external repository and using the GTFS-Madrid-Bench lifting mappings. It is important to point out that the performances obtained strictly depend also on the Triplestore employed (GraphDB Free v9.0.0). Indeed, the results obtained showcase how the limitation of 2 concurrent queries of the triplestore affect the performances of both lifting and lowering.

The most relevant insight is that the usage of an external repository with incremental writes (W-I) drastically reduces the memory consumption with respect to the same configuration run using an in-memory repository (*frel-4*) or writing the entire knowledge graph at the end of the lifting materialization (*frel-10*). The conversion time using incremental writes (W-I) is higher but can be acceptable in the tradeoff between the time required and resource usage. Moreover, the adoption of a triplestore without concurrency limitations would lower that difference also exploiting better the implemented option for concurrent incremental writes (W-IC). However, as discussed in the following paragraphs for the 50-csv dataset, it is important to highlight that adopting an external repository also the resource usage of the Triplestore should be considered.

In the case analysed, the configurations *frel-6* and *frel-7*, having a greater batch size, offer the best tradeoff between performances in the conversion time and resource usage. However, in our experience with the Chimera Converter, the optimal batch size depends on many factors (mappings, triplestore, etc.) and can be fine-tuned only considering a specific pipeline and environment. Similarly, in the considered case, the lowering time is higher when using an external repository (due to the concurrency limitations also for the *reading* queries in the template), but in our experience, very complex queries in templates applied to large knowledge graphs can effectively benefit of an external triplestore [2]. For example, in cases where lifting and lowering have similar execution times, it may be beneficial to increase a little bit the lifting time writing triples to an external repository, to

take advantage of a speed up in the lowering phase thanks to the reading performances of the triplestore.

Table 43 Tests for TC11 with the 10-csv dataset using an external repository

	ER	W	BS	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
10-csv-crel	-	-	-	544.41	536.9	7.51	9.47	140.82
10-csv-frel-6	X	I	100K	247.94	229.95	17.99	7.07	226.67
10-csv-frel-7	X	IC	100K	216.14	197.92	18.22	9.83	288.25
10-csv-frel-8	X	I	10K	357.8	339.88	17.92	6.11	172.99
10-csv-frel-9	X	IC	10K	283.07	264.91	18.17	7.68	219.24
10-csv-frel-10	X			186.72	168.92	17.79	16.24	345.54

Table 44 compares the tests performed with the 1,5,10,50,100 -csv, -json and -xml datasets considering the *crel* and *frel-4* configurations⁶¹ with the GTFS-Madrid-Bench. Additional detailed data are reported in Section 7.3 in Appendix B. The F-Rel version of the Converter obtained better results for every format and scale, in particular, considering XML datasets that the C-Rel version was not able to convert, even in scale 1, within the timeout. The F-Rel version was able to convert CSV, JSON and XML datasets up to 100 MB and generating 18 million triples with the available resources. The scalability limit is still set at scale 100, but for CSV and XML datasets it is now related to memory consumption and no more to the timeout. Analysing the case of the 50 scale datasets, we will discuss why this problem cannot be effectively solved using an external repository.

Table 44 Tests for TC11 with the GTFS-Madrid-Bench datasets. Average conversion times in seconds (s) are reported for each dataset.

Scale	1		5		10		50		100	
Input Size (MB)	4.9		10.42		23.64		106.1		247.5	
Triples	565,489		1,800,911		3,663,380		18,009,100		36,633,800	
Rel	C	F	C	F	C	F	C	F	C	F
CSV	22.77	10.83	164.95	55.37	544.41	154.13	11624.45	3441.67	TO	OM
JSON	50.41	35.17 (30.89)	659.11	519.64 (394.21)	2471.29	2132.18 (1467.70)	66003.36	54074.9 (34901.65)	TO	TO
XML	TO	16.26	TO	123.29	TO	434.05	TO	12648.65	TO	OM

Table 45 compares the tests performed with the 50-csv datasets considering the *crel*, *frel-4*, *frel-6* and *frel-10* configurations with the GTFS-Madrid-Bench. As discussed for the 10-csv dataset, the adoption of incremental writes to an external repository drastically improves the memory consumption of the Converter with an acceptable delay in the conversion time (also limited by the GraphDB Free version). Interestingly, in this case, *frel-10* registered a lower lifting time with respect to *frel-4*, suggesting that a not-incremental write of a great number of triples can be managed faster by the external Triplestore.

⁶¹ For the JSON datasets we reported also the *frel-0* data in parentheses, that as commented in TC9 performs better for this format.

Table 45 Tests for TC11 with 50-csv and external repository

	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
50-csv-crel	11624.45	11583.49	40.97	18.84	185.56
50-csv-frel-4	3441.67	3407.39	34.28	18.58	516.51
50-csv-frel-6	3784.34	3659.39	88.95	9.63	314.61
50-csv-frel-10	3013.78	2919.4	94.38	18	401.54

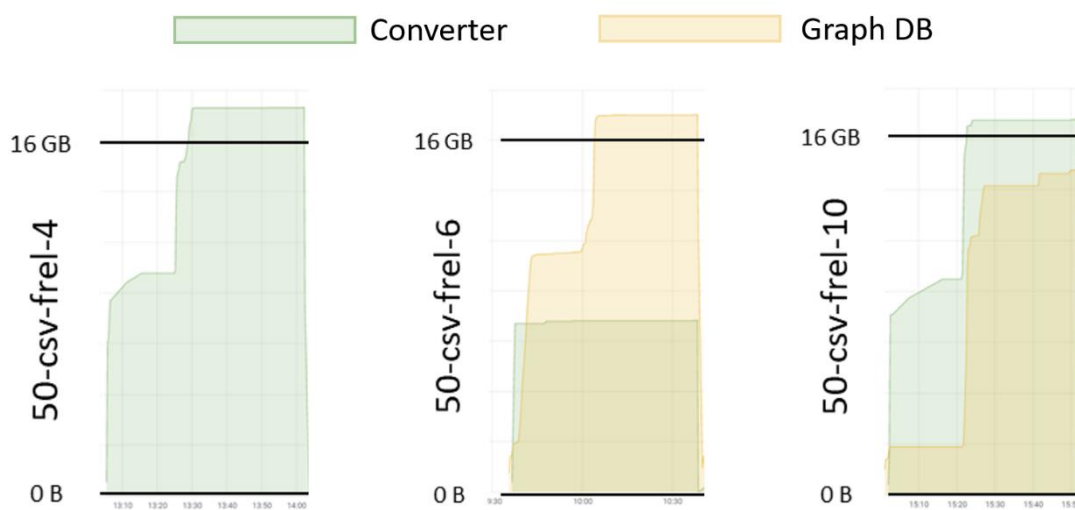


Figure 69 Converter and GraphDB memory consumption for TC-11 with 50-csv dataset.

The obtained results for 50-csv with *frel-6* are promising in terms of memory consumption but testing the same configuration with 100-csv still do not terminate the conversion procedure. The problem is pointed out by Figure 69 showcasing the memory consumption of the Converter and GraphDB containers for each configuration with the 50-csv dataset.

It is clear how *frel-6*, in particular, but also *frel-10*, shift the performance bottleneck to the triplestore that cannot handle the pace of the lifting procedure in indexing new triples added to the repository.

3.5.4 TC12 - Scalability Testing for Runtime Data/Message Conversion

The T12 scalability test evaluates the Converter adopting the annotation and declarative approaches in the runtime data/message conversion scenario and considering different increasing workloads.

Table 46 TC12 Scalability Testing for Runtime Data/Message Conversion

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.
Preparation	The Converter JARs with different pipeline configurations, the input messages (ST4RT test cases, VBB endpoint example response). Jenkins with the Performance plugin. JMeter tool.
Testing Environment (Setup)	The Converter JARs with different pipeline configurations, the input messages (ST4RT test cases, VBB endpoint example response). Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz, 4 core; 12 GB RAM; CentOS Linux release 7.8.2003 (Core), Jenkins v2.235.2 and the Performance plugin v3.18, Apache JMeter v5.3, Java (Java 8 for Annotations approach, Java 11 for the declarative approach).
Testing Scripts	For each configuration, the related Converter JAR is executed. Different JMeter test plans are defined to perform an increasing number of concurrent requests (load test) to the running Converter using the input messages provided.
Execution	The JMeter test plans are executed through Jenkins and results are gathered and visualized using the Performance plugin.

Analysing Results

The trend of response times over time, the average conversion time (with related statistics) and the number of performed/failed requests are collected considering the different test configurations.

Preparation and Setup TC12

The different converters are built as executable JARs to test the different pipelines and configurations, as described for TC10 in Section 3.5.2. Each converter is configured to expose an endpoint accepting requests for message conversion. The hardware and software specifications of the test environment are described in Table 46 (Setup).

The Apache JMeter tool is used for configuring test plans (*jmx* files) and running them for the different converters. The test plans were uploaded and set up to be executed through a Jenkins pipeline. The Performance plugin was installed in Jenkins's plugin manager to gather and visualize performance data on the tests executed.

Validating performance test TC12

As described for TC10 in Section 3.5.2, we considered the *annotation* and *declarative* approaches to building a Converter for the testing activities on the runtime data/message scenario.

Tool: Converter built using the Chimera framework.

Annotation approach: *ST4RTLiftingProcessor* and *ST4RTLoweringProcessor*

Declarative approach: *RMLProcessor* for the lifting, *TemplateProcessor* for the lowering. Configuration *frel-m-1*, performing best in TC10, was considered.

Input

Annotation approach: The test message Type 1 – FSM-to-918 defined in the ST4RT project is used as input for the conversion process.

Declarative approach: An example message obtained from the HaCon VBB endpoint, like the one shown in Figure 30, is used as input for the conversion process.

Output: The responses provided by the Converter to the requests made. JTL files containing results of the load test execution.

Script: The tests are defined as test plans using the JMeter tool and executed through Jenkins pipelines.

Execution TC12

The test plans defined using JMeter are the same for the two approaches but with different configurations for the endpoint and the input messages. The test plans were run considered an

increasing number of concurrent requests for each Converter until relevant performance degradation and errors were observed. Before the scalability tests, we performed a pre-testing session to ensure that the test plans will be started and executed via Jenkins correctly.

The test plans in JMeter define a *Thread Group* performing HTTP requests to the Converter endpoint (as for TC10 in Section 3.5.2), and several *Listeners* to collect performance data. The parameters⁶² configured for the Thread Group are:

- *Number of Threads (NoT)*: **10, 50, 100, 150, 200, 500, 1000, 2500, 5000** (number of concurrent requests).
- *Ramp-up period (R)*: **1 second**.
- *Loop Count*: **1**.

The tests based on JMeter test plans (jmx files) were executed through Jenkins pipelines. The results were captured into *jtl* files.

Analysis of results TC12

The data obtained from the JMeter tool (*jtl* files) were analysed after the execution of each configuration using the Performance Plugin of Jenkins.

Core results obtained are reported in this section. Section 7.4 in Annex B contains complete reports on the trend of response times over time, the average conversion time (with related statistics) and the number of performed/failed requests considering the different test configurations.

Annotation approach

Table 47 reports the results of the tests performed for each configuration considering the *annotation approach*. The converter implemented in the ST4RT project was designed as single-threaded. In the ST4RT testing activities, the messages were sent to the broker in series and, therefore, parallel processing was not tested. During the SPRINT project, the ST4RT converter was refactored and improved to handle multithreading and to allow the processing of multiple messages in parallel. The SPRINT Converter managed to handle up to 200 concurrent requests in the performed scalability evaluation of the *annotation approach*. As discussed in the analysis of results for TC10 in Section 3.5.2, the generality of the approach implemented to process annotations on classes does not allow to optimize the processing for a specific type of messages, especially during the lowering. This approach makes lifting and lowering demanding in terms of time and hardware resources, thus limiting the scalability that, however, can be improved considering multiple instances.

⁶² Each parameter is explained in details in Section 3.4.2

Table 47 TC12 Results of tests for the annotation approach

Number of concurrent requests (N)	Avg Time of processing N Requests [ms]	Interval between requests [ms]
10	800	100
50	7 255	20
100	11 684	10
150	15 514	6,7
200	21 280	5
500	Not completely processed	2

Declarative approach

Table 48 reports the results of the tests performed for each configuration considering the *declarative approach*. The SPRINT Converter managed to handle up to 2500 concurrent requests in the performed scalability evaluation of the *annotation approach*. Indeed, after 3000 pending requests the queue mechanism provided by Apache Camel starts dropping enqueued requests. The maximum length of the queue can be increased, however, a high number of pending requests points out that noticeable performance degradation is observed. The lower usage of resources and the optimizations made possible by the *declarative approach* result in very good scalability results for increasing workloads, even considering a single instance of the Converter.

Table 48 TC12 Results of tests for the declarative approach

Number of concurrent requests (N)	Avg Time of processing N Requests [ms]	Interval between requests [ms]
10	131	100
50	219	20
100	775	10
150	1 663	6,7
200	1 918	5
500	3 926	2
1000	7 567	1
2500	21 114	0,4
5000	Not completely processed	0,2

3.5.5 TC13 - Scalability Testing for Runtime Environment Deployment

The presented Converter test cases were performed considering one replica of the service. However, as shown in the functional validation for S9 (cf. Section 2.6), 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.

Table 49 TC13 Scalability Testing for Runtime Environment Deployment

TC13 - Scalability Testing for Runtime Environment Deployment	
Process	A Converter is deployed on a runtime environment measuring the time to complete 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.
Preparation	Docker, docker-compose, Converter JAR, Unix shell
Testing Environment (setup)	<p>The Converter JAR, the Converter Dockerfile, the docker-compose file for the Converter are loaded into a machine with the following characteristics:</p> <p>CentOS Linux 7, Intel® Xeon 8-core CPU D-1521 @ 2.40GHz and 64 GB Memory, Docker v19.01.13, docker-compose v1.18.0</p>
Related Scenarios	Scenario 9 - D5.4
Testing Scripts	Bash script building the Docker Image using the Jar and the Dockerfile, and launching a different number of replicas for the Converter service defined in the docker-compose file.
Execution	The script is deployed and executed. Each test configuration is executed 5 times.
Analysing results	The average time to deploy a replica container and to initialize the Converter endpoint.

Preparation of the test environment and Setup TC13

The Docker image of the Converter is built to run the JAR using a Java 11 capable base image (openjdk:11-jdk on Dockerhub). The JAR file of the Converter is configured to run an example pipeline that can be invoked through a REST API exposed on port 8888.

The hardware and software specifications of the test environment are described in Table 3 (Setup).

Validating scalability test TC13

The IF architecture does not define a default deployment environment, therefore, for the F-Rel we tested TC13 considering a local Docker runtime to obtain average deployment time for a Converter container. Performances can be different considering different deployment environments (e.g. a Kubernetes cluster).

For the scalability tests, we considered an increasing number of replicas to be deployed. Deployment statistics are collected using the logs of the container and the container information.

The time to deploy the container is computed comparing the timestamp of the command launching the container and the *startedAt* timestamp provided using the following command:

```
docker inspect --format '{{ .State.StartedAt }}' chimera-converter
```

The time obtained with the described method measures the creation of the container (**create**) but does not consider the time required to initialize the Converter JAR and the Camel routes (**available**). The time to initialize the Converter endpoint can be computed comparing the timestamp of the command launching the container and the timestamp referring to the endpoint initialization in the container logs. This method allows collecting a different timestamp for each replica, despite the load balancing mechanism used for the Converter endpoint. Different methods to measure the readiness of each replica can be adopted in different deployment environments (e.g. probes in Kubernetes⁶³).

Tool: Converter built using the Chimera blocks.

Script: The script developed for this test is based on a bash script. The script launches X replicas of the Converter service using the provided docker-compose file. Then, it accesses the logs of the replicas and the information of the different containers to obtain the time spent for the deployment.

Execution TC13

For the scalability tests, we considered an increasing number of replicas to be deployed in the Docker environment: 1, 3, 5 and 10. Each test was executed 5 times.

⁶³ <https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>

Analysis of the results TC13

Values reported in the section are computed as averages on the data of the 5 executions for each test.

Considering the command to launch a single replica we obtained the following times for the time required to create the container.

Create (s): 2.41, 2.67, 2.44, 2.62, 2.54

Average Create: 2.54s

Considering commands instantiating an increasing X number of replicas, we obtained the following *average* time per replica to initialize the endpoint. Table 50 reports for each execution of the test, the average among the times obtained for the X replicas deployed. The last column reports the average of the averages over the 5 executions.

Table 50 Average Available time per replica in TC13

Num. replicas	Average Available time per replica (s)					Average over the 5 executions
1	12.81	12.49	12.94	13.14	12.57	12.79s
3	14.91	15.29	14.43	13.92	15.04	14.72s
5	17.57	17.83	17.75	17.39	18.33	17.77s
10	25.82	26.7	25.69	26.25	25.91	26.07s

It is possible to notice that the creation time is much lower than the time required for the endpoint to be available, but this is highly influenced by the time required to initialize the webserver exposing the endpoint. Moreover, the time predictably increases when the number of replicas requested is higher. Times registered are, however, lower than the target values defined in D3.4. The deployment times obtained can guarantee a fast scalability process to cope with load peaks demanding an increase in the number of replicas answering the conversion requests (Scenario S9 in Section 2.6).

3.6 DISTRIBUTED SPARQL ENDPOINT

We will execute a set of SPARQL queries including preference criteria to measure how much it costs to evaluate these features on SPARQL queries. As baseline we use similar SPARQL queries without the criteria preferences. In this context, we will analyze the performance of the queries on two endpoints.

3.6.1 TC14 - Performance Testing for Distributed SPARQL endpoint

Table 51 TC14 Performance Testing for Distributed SPARQL endpoint

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
Performance Criteria	Total execution query time: Average execution time of queries
Preparation	S.O. Ubuntu, Ontario, Docker and docker-compose, Virtuoso
Testing Environment (setup)	The experiment configurations are loaded into a machine with the following characteristics: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, 40 cores, 32RAM, 50GB HDD with Ubuntu 18.04 as its operating system, Jenkins v2.222.4, Apache Maven 3.6.3, Java 8
Related Scenarios	Scenario 4 - D5.4
Testing Scripts	Bash script with a test plan from each case ⁶⁴
Execution	The generated script plan is deployed and executed with the Jenkins server
Analysing results	The maximum memory usage(KB) and the execution time for each query were obtained.

⁶⁴ https://github.com/oeg-upm/sprint/blob/main/Ontario/runQ1_wop.sh

Preparation of the test environment and Setup TC14

The hardware, software requirements for the test environment has been described in the Table of the previous section, and the steps for executing the tool are described more in detail in the GitHub repository⁶⁵.

Validating performance test TC14

Performance test execution principally involved the query execution with preferences and without preferences over two SPARQL endpoint, the proof of concept of this approach it is detailed in D5.5 section 2.2. Ontario executes the federated queries on RDF repositories using Virtuoso and then, it combines resources from these RDF repositories to process the results.

Tool : Ontario that allows us to federate the query to two SPARQL endpoints.

Input: Virtuoso that handles two RDF datasets based on Transport DCAT-AP⁶⁶ and the queries⁶⁷.

Output: Results in JSON format.

Script: The script developed for this test is based on a bash script with loops that allow us to perform several executions and measure the execution times for each query.

Execution TC14

To perform the test and can run the tool has prepared a script in bash that will be executed from the Jenkins server through the creation of a new item as you can see in Figure 70.

⁶⁵ <https://github.com/oeg-upm/sprint/tree/main/Ontario>

⁶⁶ <https://github.com/cef-oasis/DCAT-AP>

⁶⁷ <https://github.com/oeg-upm/sprint/tree/main/Ontario/queries>

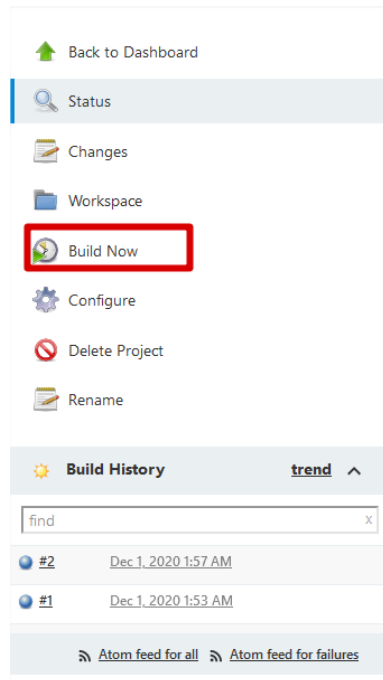


Figure 70 Item in Jenkins that executes the test script

Analysis of the results TC14

We obtain the results in a CSV file where saved the following information for each query execution:

- *Elapsed_time*: Elapsed real (wall clock) time used by the process, in seconds.
- *Kernel_mode*: Total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
- *User_mode*: Total number of CPU-seconds that the process used directly (in user mode), in seconds.
- *Memory_max(Kbytes)*: Maximum resident set size of the process during its lifetime, in Kbytes.

In Figure 71 you can see how the time of a query with preferences is widely more expensive with respect to the consultation without preference.

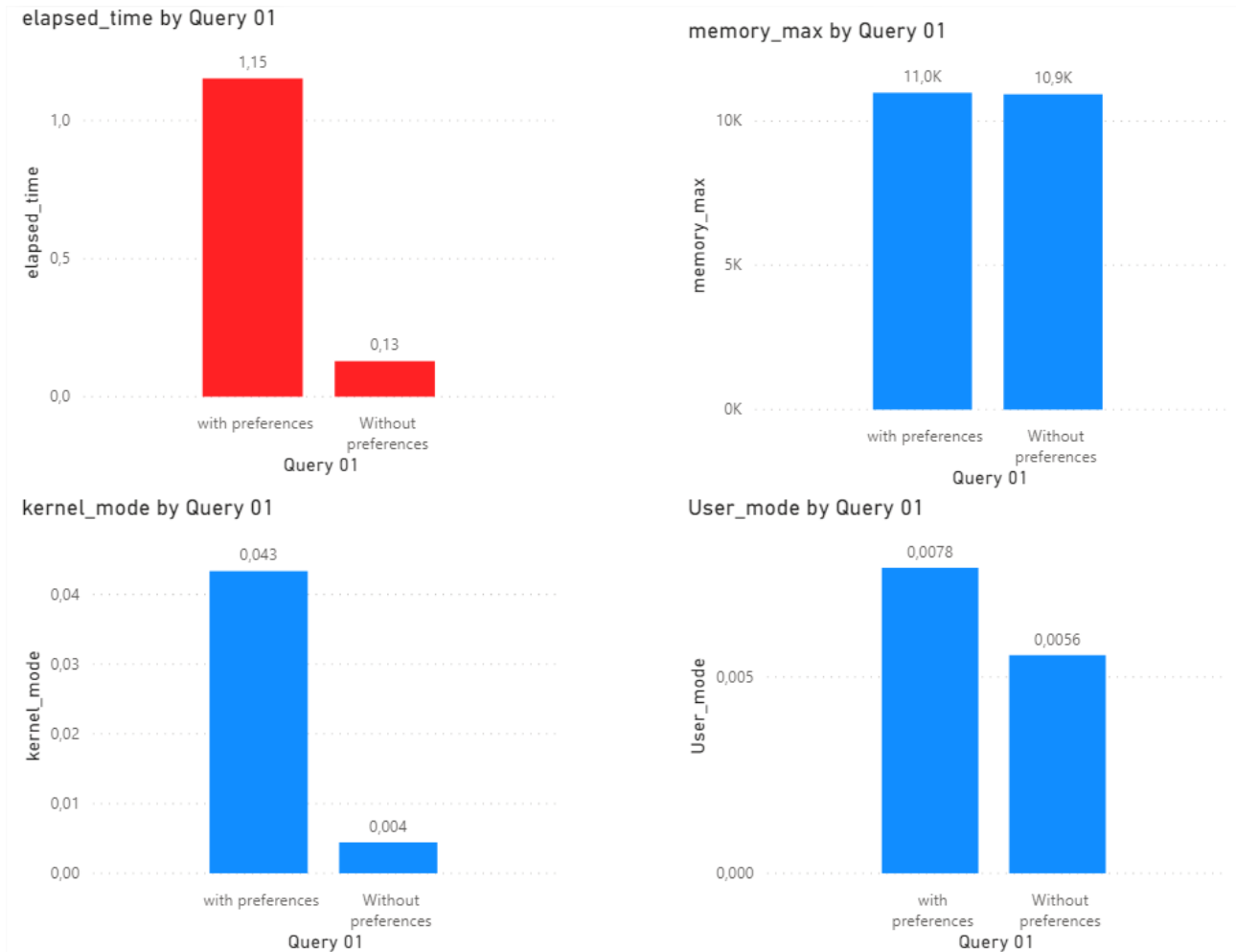


Figure 71 Performance test results

3.6.2 TC15 - Scalability Testing for Distributed SPARQL endpoint

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.

Table 52 TC15 Scalability Testing for 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
Scalability Requirements	SR-. Query/search time
Scalability Criteria	Total execution query time: Average execution time of queries
Preparation	S.O. Ubuntu, Ontario, Docker and docker-compose, Virtuoso
Testing Environment (setup)	The experiment configurations are loaded into a machine with the following characteristics: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, 40 cores, 32RAM, 50GB HDD with Ubuntu 18.04 as its operating system, Jenkins v2.222.4, Apache Maven 3.6.3, Java 8
Related Scenarios	Scenario 4 - D5.4
Testing Scripts	Bash script with a test plan from each case
Execution	The generated JMeter script plan is deployed and executed with the Jenkins server
Analysing results	The maximum memory usage(KB) and the execution time for each query were obtained.

Preparation of the test environment and Setup TC15

The hardware, software requirements for the test environment has been described in the table above (Setup) and the steps for executing the tool are described in more detail in the GitHub repository⁶⁸. In Figure 72 we show an example of the query that is used for our scalability test over twelve SPARQL endpoints.

Query without preferences

```
# Query for the frequently updated datasets (qual_freq) with the modified time (metadata_date)
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT * WHERE {
  ?d a dcat:Dataset .
  ?d dct:qualFreq ?qf .
  ?d dct:metadataDate ?md .
}
```

Query with preferences

```
# Query for the most frequently updated datasets (qual_freq)
# with the oldest last modified time (metadata_date) can be expressed by means of: SKYLINE OF qual_freq MIN, metadata_date MAX

PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT *
WHERE {
  ?d a dcat:Dataset .
  ?d dct:qualFreq ?qf .
  ?d dct:metadataDate ?md .
  OPTIONAL {
    ?d_ a dcat:Dataset .
    ?d_ dct:qualFreq ?qf_ .
    ?d_ dct:metadataDate ?md_ .
    ?d_ a dcat:Dataset .
    ?d_ dct:qualFreq ?qf_ .
    ?d_ dct:metadataDate ?md_ .
    FILTER (?qf_ <= ?qf && ?md_ >= ?md && (?qf_ < ?qf || ?md_ > ?md))
  }
  FILTER(! BOUND(?d_))
}
```

Figure 72 Example of a query with and without preferences

Validating performance test TC15

Scalability test execution principally involved the query execution with preferences and without preferences over twelve SPARQL endpoint, once again the proof of concept of this approach it is detailed in D5.5 Section 2.2. Ontario executes the federated queries on RDF repositories using Virtuoso and then, it combines resources from these RDF repositories to process the results.

Tool : Ontario that allows us to federate the query to twelve SPARQL endpoints.

⁶⁸ <https://github.com/oeg-upm/sprint/tree/main/Ontario>

Input: Virtuoso that handles twelve RDF datasets based on Transport DCAT-AP⁶⁹ and the queries⁷⁰. This is done following the steps indicated in the repository and executing the *docker-compose*⁷¹ file.

Output: Results in JSON format.

Script: The script developed for this test is based on a bash script with loops that allow us to perform several executions and measure the execution times for each query.

Execution TC15

To perform the scalability test and run the tool we prepared a script in bash that will be executed from the Jenkins server as you can see in Figure 73.

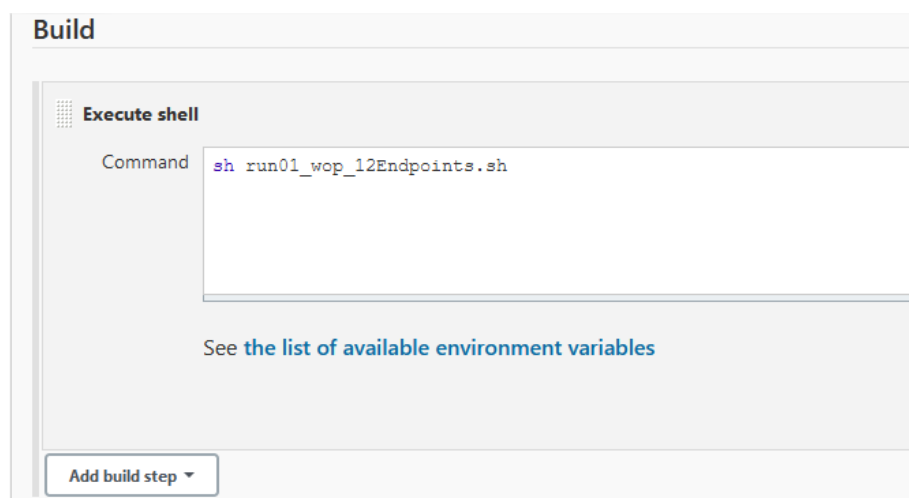


Figure 73 Running a query test script for 12 SPARQL endpoint

Analysis of the results TC15

The analysis of the results is done with the same measurement parameters as TC14. The Figure 74 shows how the average execution time of a query with preferences is very high compared to the query without preferences, this is because the query federation tool does not have optimizations designed when the data source scales horizontally to increase the number of SPARQL endpoints.

⁶⁹ <https://github.com/cef-oasis/DCAT-AP>

⁷⁰ <https://github.com/oeg-upm/sprint/tree/main/Ontario/queries>

⁷¹ <https://github.com/oeg-upm/sprint/blob/main/Ontario/docker-compose.yml>

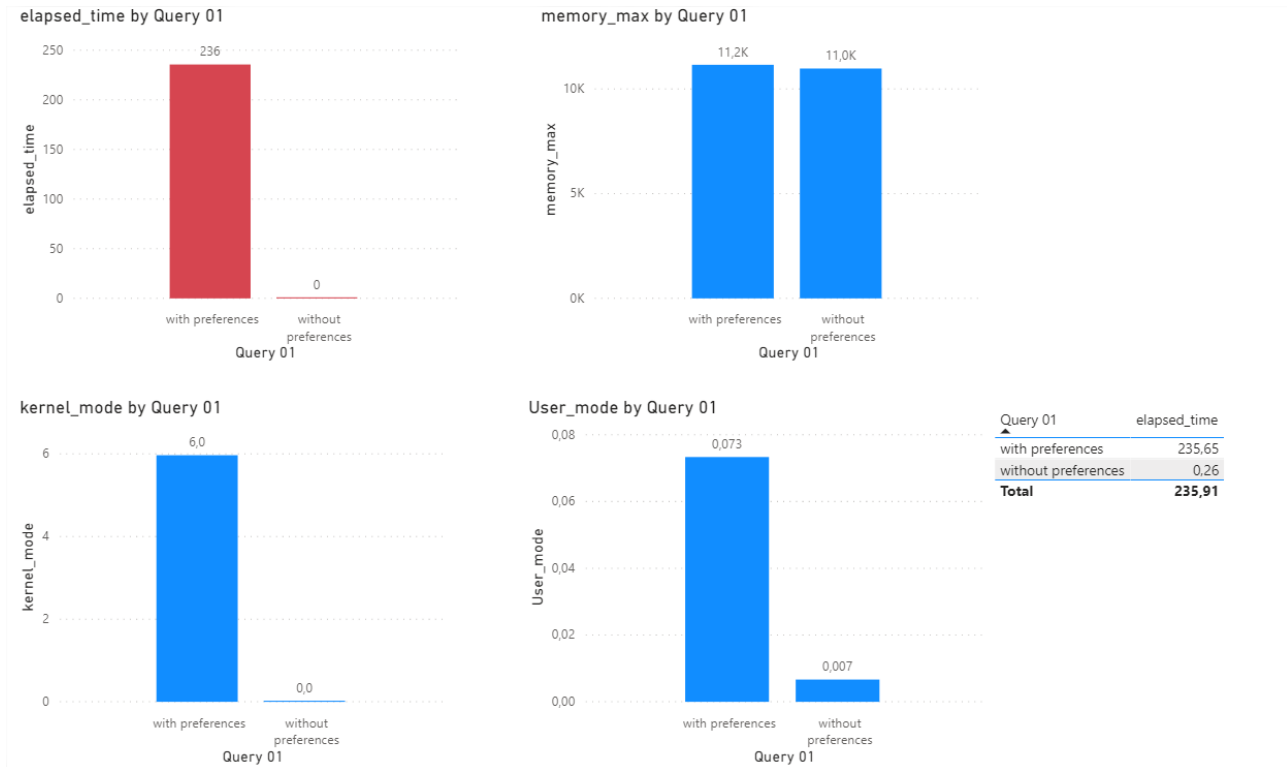


Figure 74 Scalability test results

3.7 OVERALL PERFORMANCE AND SCALABILITY EVALUATION

This section reports the F-Rel performance and scalability requirements, described in D3.4, and provides for each component the overall conclusions after running the tests.

3.7.1 Collaborative Ontology Manager

In general conclusions, the Collaborative Ontology Manager tool has an acceptable performance since the average time of documentation generation, evaluation of the tested ontologies is 26 seconds, as can be seen in the TC1, TC2 and the execution time does not have an excessive increase even when the ontology increases in several classes and properties. We conclude that the target value defined in D3.4 is satisfactorily fulfilled by not exceeding the 5 minutes the target value initially set.

Table 53 Collaborative Ontology Manager tool performance and scalability requirements

Description	Ref. the to full description of the Requirement and KPIs	F-Rel status
Ontology Documentation/Validation Time	Requirement in Deliverable D.3.4: PR-8	OK
Scalable Ontology Documentation/Validation Time	Requirement in Deliverable D.3.4: SR-7	OK

3.7.2 Automatic generation of ontologies from non-ontological sources

In TC3 and TC4 we can see the transformation times over a set of NeTEx XSD files. This test is performed with approximations over real files that are used for information exchange in the transport domain. We can observe that the execution times are in the order of magnitude of a few seconds. This leads us to conclude that it would be a good alternative to accelerate the ontology generation process. Defenitly, this process helps in the first steps of the ontology development.

Table 54 XSD2OWL tool performance and scalability requirements

Description	Ref. the to full description of the Requirement and KPIs	F-Rel status
Non-ontological conversion time	Requirement in Deliverable D.3.4: PR-9	OK
Scalable Non-ontological conversion time	Requirement in Deliverable D.3.4: SR-8	OK

3.7.3 Mapping Tool

As demonstrated in Section 3.3 (See Table 26), the target values for the Execution time has been successfully achieved for the F-Rel implementation of the tool. In specific, the overall Execution Time of the tool is **122,5953 s** and Overall Execution Time per each Term is **0,5476 s** which are satisfying the requirements specified in Table 55. Similarly, the usability of the system met the defined requirements with the overall 5 steps to generate the mappings and annotations.

Table 55 Mapping suggestion tool performance and scalability requirements

Description	Ref. the to full description of the Requirement and KPIs	F-Rel status
Execution Time	Requirement in Deliverable D.3.4: PR-6	YES
Usability	Requirement in Deliverable D.3.4: PR-7	YES

3.7.4 Asset Manager

All performance and scalability requirements expressed for F-Rel were satisfied. With respect to the performance requirement PR2, we can notice that the Asset Manager average results are around half a second, which is ten times better than the requirement value. Scalability requirement as expressed by SR2 also vastly outperformed the expectations. The requirement value was 10 parallel requests completed in 5 seconds, and the F-Rel Asset Manager was able to sustain 100 parallel requests with an average load time of 5 seconds. Moreover, we were able to answer to more than 200 requests in parallel before starting to obtain timeout errors. This means that even if the Asset Manager has been designed for a clustered deployment which components deployed on different machines, it was able to provide acceptable performances also when deployed on a single server.

Table 56 Asset Manager performance and scalability requirements

Description	Ref. the to full description of the Requirement and KPIs	F-Rel status
Performance requirements for Ad-hoc Asset creation: (exemplified scenario for Converter asset)	Requirement in Deliverable D.3.4: PR1	OK
Performance requirements for Exploration API: DCAT-AP metadata retrieval	Requirement in Deliverable D.3.4: PR2	OK
Scalability requirements for Direct Download of an Asset (exemplified scenario for Converter asset)	Requirement in Deliverable D.3.4: SR1	OK
Scalability requirements for Exploration API: DCAT-AP metadata retrieval	Requirement in Deliverable D.3.4: SR2	OK

3.7.5 Converter

Considering the performed testing activities on performance and scalability for the IF Converter developed in SPRINT, this section draws conclusions for this component.

Table 57 Converter performance and scalability requirements

Description	Ref. the to full description of the Requirement and KPIs	F-Rel status
Response Time to convert the whole data set	Requirement in Deliverable D.3.4: PR-4	OK
Response Time to convert one message	Requirement in Deliverable D.3.4: PR-5	OK
Scalability requirements for Runtime environment deployment of an Asset (exemplified scenario for Converter asset)	Requirement in Deliverable D.3.4: SR-4	OK
Response Time to convert big datasets	Requirement in Deliverable D.3.4: SR-5	OK
Response Time to convert multiple concurrent messages	Requirement in Deliverable D.3.4: SR-6	OK

The results obtained in the different test cases, satisfy the target values defined in D3.4 for the related KPIs (reported in Table 57) and prove that the reference solution proposed for the Converter can guarantee good performances and scalability in both the batch and the message/runtime data scenarios.

One of the main advantages of the SPRINT Converter is its higher configurability thanks to the modularity of the solution and the development of different blocks providing several configuration options (cf. D5.5). However, as commented in the analysis of the results, different configurations can perform better under certain circumstances and there exist some limits that should be considered.

3.7.6 Distributed SPARQL endpoint

We have performed a set of SPARQL queries with preference criteria in order to analyze and understand the pros and cons of these kind of queries, that they are usually very relevant for stakeholders to take decisions and make a proper data analysis. We can observe from the results

obtained in the TC14 and TC15 that preference queries have a low performance due to two main causes. First, the preference query is more complex since it performs major filtering of the data and the second is that the tools to make distributed queries do not integrate optimizations in their algorithms. It would be a recommendation to perform a more extensive study on optimizations that can be applied in distributed SPARQL queries with preferences criteria.

Table 58 Distributed SPARQL endpoint performance and scalability requirements

Description	Ref. the to full description of the Requirement and KPIs	F-Rel status
Asset/Service Discovery Response Time	Requirement in Deliverable D.3.4: PR-3	OK
Scalability of Query/search time	Requirement in Deliverable D.3.4: SR-3	OK

4. BUSINESS AND MARKET VALIDATION

Chapter 4 “Business and Market validation” of SPRINT Deliverable D5.3 “Validation of pilot implementation (C-REL)” describes how the SPRINT proof of concept of the IF has been designed to contribute to key evaluation indicators based on the recommendations for the IF development and deployment produced in GOF4R project’s D5.2 “Toolkit of recommendations and KPI Scoreboard” deliverable. It shows in particular that addressing all indicators *simultaneously* in an ecosystem that is *inherently* heterogeneous and unbounded cannot be achieved with a single capability implemented in a single solution:

- **Data openness** cannot be reduced to a problem of design of data structures and flows, but must be complemented by tooling that enable ecosystem participants to implement operational procedures that can leverage the data effectively in concrete business processes as exemplified in scenarios S1-S9 of the prototype. The Asset Manager IF component provides these capabilities
- **Gaining a ‘critical mass’ of IF participants** requires minimization of the need for adaptation of legacy systems and the reduction of governance provisions for participating in the ecosystem to the ability to register/publish own digitalized resources to discover those published by other partners in such a way that all participants retain full control of their digital assets. The combination of Asset Manager and Converter components provides this capability
- **Market diversity and inclusiveness** requires the minimization of development costs and governance overhead, making interoperability affordable for smaller organizations. Organizing the IF as a collection of out-of-the-box tools supports this capability.
- **Stakeholder’s management**, particularly the difficulty of interoperating across different transport mode specialists, data specifications, authorities and regulatory regimes requires the ability to separate the semantic of interoperability from the data specifications in such a way that the former can be defined explicitly in a form amenable to digital processing, i.e. described in a machine-readable ontology, and used to automate the conversion across different forms of the latter. Ontology engineering, mapping and converter framework components working in conjunction with the Asset Manager provide these capabilities.
- **Use Friendliness**, in particular addressing skills constraints in ontology and semantic interoperability, requires the provision of ‘building blocks’ developed according to open standards and widely available open source frameworks that encapsulate deeper specialist knowledge, allowing fairly standard ICT organizations to compose solutions that integrate in their existing environments.
- **Reliability and security of the ecosystem** requires an architecture that separates the provisions for reliability and security from the components that implement the interoperability mechanism, so that the latter can be deployed as units of pure functionality while reliability and security are delegated to the underlying runtime environment. This capability is absolutely essential to contribute to the objectives of rapid gaining of ‘critical mass’ and reduction of cost in the participation in the ecosystem as well, including across different mobility modes and organizations, given that security and reliability requirements and runtime environments vary greatly across them under many aspects, from identity and authentication mechanisms, to network topologies, to redundancy and back up possibilities, etc.

- **Creation of new business models** requires that the interoperability mechanisms must not incorporate application (business) level logic, so that interoperability does not induce dependencies on the range and type of applications that can be developed on a *common* interoperability mechanism. This is at the same time a critical architectural decision to support interoperability across different mobility modes and organizations.
- **Costs to join the IF** as it relates to ecosystem participants' acquisition of the necessary professional skills and adaptation of legacy systems must also be addressed by the separation of pure interoperability functionality from application level logic and the encapsulation of specialized knowledge on the semantic interoperability technologies in building block components that can be used by participant's ICT organizations in implementing specific solutions, e.g. automated conversion across different data specifications, matching local requirements.

Addressing all indicators simultaneously requires, in sum, a collection of different tools and, at the same time, an architecture that allows their flexible combination into different configurations each adapted to the particular challenges of the specific environment. Heterogeneity of environments, both technical and organizational, and different levels of ICT maturity or penetration of digitalization concepts across a large variety of ecosystem participants, and across modes of transportation, are in fact a real world situation that must be embraced and leveraged by providing participants with the tools to make it interoperable.

The preceding sections 2 and 3 of this document describe, respectively, the use case and functional validation scenarios, and the performance tests of the different specialized components, designed to meet all requirements listed in annex A and implemented according architecture principles that make them composable, of the IF.

To test the suitability of the developed components to be used in combination to support specialized environments, additional validation scenarios S10 through S13 have been added in this F-REL version of the validation of the pilot implementation.

Scenario S13 in particular has been introduced to demonstrate the ability of the IF to operate in an environment that includes National Access Points, i.e. existing external systems that support regulatory provisions and are implemented in different architectures, technologies and capabilities in different European Member States, that is a 'given' environment that cannot be 'adapted' but must, by regulation, be part of the ecosystem.

A significant result has achieved in that NAPs can be accessed by the Access Manager, importing the NAPs metadata while using IF converter technology to transform the metadata into the DCAT-AP vocabulary and consolidating them into the semantic graph in the local RDF repository. With the ability of the Asset Manager to perform 'discovery' queries over both local and remote (e.g. in NAPs) assets, the Asset Manager becomes a federator (or 'aggregator') giving ecosystem participants access to resources everywhere, in NAPs or elsewhere, across Europe, at the same time multiplying the effectiveness of NAPs themselves.

The combination of Converters and Asset Manager, i.e. the ability of Converters of importing semantic mappings stored in the Asset Manager as a specific type of asset, allows additionally the ability to collect data from NAPs identified by the Asset Manager converting them at the same time from the NAP to the specific data specification required by the ecosystem participant, and similarly

to convert a data set produced in the ecosystem participants own data specification to that accepted by the NAP. A dependency tracking mechanism in the Asset Manager specifies the relation between the converter and its mapping so that the converter can be regenerated and re-tested if a change is detected in the mapping.

Additionally, since the Asset Manager implements an asset management workflow, and since NAPs metadata in the Asset Manager are, in fact, assets, NAP's metadata can also be subjected to a workflow that can detect discrepancies and data quality issues, allowing for correction and approval before being made available as 'published' resources to ecosystem participants. Validation of the S13 scenario has indeed revealed numerous such issues with resources imported from existing official NAPs: this shows again that a 'common respository of data sets in a common format' is not, in itself, a solution for interoperability, but that it can be when supplemented by additional tooling. In this sense, the IF can be considered as an addition to the European 'interoperability toolbox' that can facilitate incorporation of NAPs in the ecosystem, thus enhancing the ability of NAPs to play a beneficial role in it.

Finally, IF components have also been combined to provide functionality for use in the CONNECTIVE project: access to the Asset Manager can be performed using the CONNECTIVE own Identity Provider. In addition, the publication of a GTFS dataset as an asset in the Asset Manager activates a converter that generates a semantic graph representation in RDF which can then be stored in the CONNECTIVE own GraphDB triple store.

5. CONCLUSIONS

This deliverable demonstrated how the SPRINT project worked with the objective of meeting the performance, scalability and flexibility requirements implied by the effort of establishing an efficient Interoperability Framework ecosystem. The tools which have been presented are able to cover different aspects connected to the IF. We demonstrated how we were able to ease the development phase of an interoperability solution using semantic technologies, showing that such technologies are ready for production. We also discovered that complex data conversion or service mediation problems can be broken down into smaller pieces arranged in pipelines, and we provided a framework which lets developers choose the approach they see best fitting their requirements, letting them combine annotations-based and declarative transformation and letting them integrate the resulting conversions easily in their production environments. We showed that governance can be automated through a publishing platform with embedded lifecycle management, and that such publishing platform can become an enabler for other kinds of automation.

We therefore believe that once the final governance structure for the IP4 Interoperability Framework will be devised, the tools we built will provide a good starting point for the establishment of a Single European Railway Area.

6. ANNEX A: FUNCTIONAL REQUIREMENTS TRACEABILITY MATRIX

This section summarizes the requirements for each of the components being developed for the SPRINT Interoperability Framework, and provides a summary which requirements are already fulfilled in C-Rel and which requirements will be further investigated during F-Rel. For each requirement we will provide a link to the corresponding scenario in D5.1 and D5.4.

Requirements are arranged in tables for each component. Below you can find the explanation of the columns:

- ID: requirement identifier
- Description: requirement description
- Scenario: where the requirement can be verified. It can be either:
 - a Scenario as described in D5.1 (D5.1 Sx)
 - “CONN” when the requirement comes from the collaboration scenarios between SPRINT and CONNECTIVE
 - “Dx.x” if the requirement is described in that deliverable.
- F-Rel status:
 - OK: the requirement is fully fulfilled
 - NO: the requirement is not fulfilled

6.1 AUTOMATIC GENERATION OF ONTOLOGIES FROM NON-ONTOLOGICAL SOURCES

Table 59 Functional requirements for SPRINT xsd2owl tool

ID	Description	Scenario	F-Rel status
OE-R1	Selection of XSD input reference data to be transformed	D5.4 S12	OK
OE-R2	XSD2OWL must be able to allow generating an ontological file as output	D5.4 S12	OK
OE-R3	XSD2OWL must be able to allow extract complex and simple types from XSD files	D5.4 S12	OK

6.2 COLLABORATIVE ONTOLOGY MANAGER

Table 60 Functional requirements for SPRINT Collaborative Ontology Manager tool

ID	Description	Scenario	F-Rel status
OE-R4	Collaborative ontology engineering must be able to documentation Generation of the selected ontologies	D5.4 S11	OK
OE-R5	Collaborative ontology engineering tool must be able to evaluation the Generation of the selected ontologies	D5.4 S11	OK
OE-R6	Collaborative ontology engineering tool must be able to access a public GitHub repository	D5.4 S11	OK
OE-R7	Collaborative ontology engineering tool must be able to preview generated documentation	D5.4 S11	OK
OE-R8	Collaborative ontology engineering tool must be able to generate documentation from a select step	D5.4 S11	OK
OE-R9	Collaborative ontology engineering tool must be able to generate version control(using Git)	D5.4 S11	OK

6.3 MAPPING SUGGESTER

Table 61 Functional requirements for the SPRINT Mapping Suggester

ID	Description	Scenario	F-Rel status
M-R1	It must allow users to upload to the system standard/specification vocabularies in various formats including XML and XSD for source standard, and OWL and TTL (for the target ontology)	D5.3, S7	OK
M-R2	It must build mapping of the concepts presented in the two input standards employing machine learning approaches.	D5.3, S7	OK
M-R3	It must build mapping of the concepts presented in the two input standards employing both structural mapping and machine learning approaches.	D5.3, S7	OK
M-R4	It must allow users to view the suggested mappings of the concepts.	D5.3, S7	OK
M-R5	It must allow users to interactively review and choose between various suggestions through dedicated GUI.	D5.3, S7	OK
M-R6	It must build Java and RML annotations based on the approved suggestions.	D5.3, S7	OK

6.4 DISTRIBUTED SPARQL ENDPOINT

Table 62 Functional requirements for the SPRINT Distributed SPARQL Endpoint

ID	Description	Scenario	F-Rel status
DSE-R1	Allows queries on metadata catalog	-	OK
DSE-R2	Distributed SPARQL Endpoint that can receive and processing SPARQL Protocol requests	-	OK
DSE-R3	Allows you to obtain results from a query over different sources (metadata catalog)	D5.1 S4	OK
DSE-R4	Distributed SPARQL Endpoint you to add more data sources	D5.1 S4	OK
DSE-R5	Allows to process queries with preferences about the metadata catalog	D5.3 S4	OK

6.5 ASSET MANAGER

Table 63 Functional requirements for the SPRINT Asset Manager

ID	Description	Scenario	F-Rel status
AM-R1	The Asset Manager must be able to host an arbitrary number of asset types.	-	OK
AM-R2	The Asset Manager must be able to host an arbitrary number of assets of a given asset type.	-	OK
AM-R3	The Asset Manager must allow describing remote resources/assets.	D5.1 S6	OK
AM-R4	The Asset Manager must allow attaching files to assets.	D5.1 S5	OK
AM-R5	The Asset Manager must allow users to download attachments of an asset.	D5.1 S5	OK
AM-R6	The Asset Manager must be able to visualise metadata of an asset	D5.1 S3	OK
AM-R7	The Asset Manager must be able to link an asset type to a governance process.	-	OK
AM-R8	The Asset Manager must be able to host an unspecified number of governance processes.	-	OK
AM-R9	The Asset Manager must allow both human tasks and automatic tasks to be specified in a governance processes.	-	OK
AM-R10	The Asset Manager must allow browsing the assets belonging to a specific asset type.	D5.1 S3	OK
AM-R11	The Asset Manager must allow searching for a specific asset.	D5.1 S3	OK
AM-R12	The Asset Manager must separate the Consumer's user interface and the Contributor's user interface.	D5.1 S1, S2	OK
AM-R13	The Asset Manager must implement a "Request for access" process allowing consumers to ask for the permission to access and use an asset owned by a different user.	D5.1 S3	OK

AM-R14	The Asset Manager must hide the details and attachments of an asset if not owned by the user.	D5.1 S3	OK
AM-R15	The Asset Manager must implement a notification system to inform users about the results of a request.	D5.1 S3	OK
AM-R16	The Asset Manager should notify users via email.	D5.1 S3	OK
AM-R17	The Asset Manager must allow publishing “Converter” assets.	-	OK
AM-R18	The Asset Manager must allow publishing “Ontology” assets.	-	OK
AM-R19	The Asset Manager must allow publishing “Mapping” asset.	-	OK
AM-R20	The Asset Manager must allow publishing “RDF Dataset” assets.	-	OK
AM-R21	The Asset Manager must allow publishing “Journey planning service/dataset” assets.	CONN	OK
AM-R22	The Asset Manager must allow publishing “Booking service” assets.	CONN	OK
AM-R23	The Asset Manager must allow publishing “Trip tracking service” assets.	CONN	OK
AM-R24	The Asset Manager must allow publishing “Issuing service” assets.	CONN	OK
AM-R25	The Asset Manager must allow specifying source and target specification/standard when publishing Converters.	D5.1 S8	OK
AM-R26	The Asset Manager must allow both describing a Converter as a remote service and as a downloadable artifact.	D5.1 S8	OK
AM-R27	The Asset Manager must allow reusing Ontologies, RDF Datasets and Mappings while describing a Converter.	D5.1 S8	OK
AM-R28	The Asset Manager must support the execution of arbitrary post-processing scripts to support Continuous Integration and Delivery	D4.2, D5.1 S8	OK
AM-R29	The Asset Manager must be able to create deployable artifacts for Converters, reusing the specified Ontologies, RDF Datasets and	D5.1 S8	OK

	Mappings and assembling a conversion pipeline using the Converter Framework (Chimera)		
AM-R30	When creating Converter deployable artifacts, the Asset Manager must support deployment of a Converter as a single JAR.	D5.1 S8	OK
AM-R31	When creating Converter deployable artifacts, the Asset Manager must support deployment of a Converter as a container.	D5.1 S8	OK
AM-R32	When creating Converter deployable artifacts, the Asset Manager must support deployment of a Converter as a containerised application composed by a load balancer and a replicable/scalable conversion service.	D5.1 S9	OK
AM-R33	The Asset Manager must support the publication of parametric SPARQL queries ("Exploration API")	D4.2	OK
AM-R34	The Asset Manager must allow accessing parametric queries as APIs with parameters	D4.2	OK
AM-35	The Asset Manager must be able to access National Access Points, provided that a mapping exists between the NAP metadata schema and DCAT-AP 2.0.1	D5.4	OK
AM-36	The Asset Manager must allow users to search through local and external assets	D5.4	OK
AM-37	The Asset Manager must be able to warn the owner of an asset if an asset dependency changes	D5.4	OK

6.6 USER MANAGER

Table 64 Functional requirements for the SPRINT User Manager

ID	Description	Scenario	F-Rel status
UM-R1	The User Manager must provide a login mechanism	S1, S2	OK
UM-R2	The User Manager must be able to assign different roles to users	S1, S2	OK
UM-R3	The User Manager must be able to assign permissions to users and roles	S1, S2	OK
UM-R4	The User Manager must be able to provide JWT tokens for API	-	OK
UM-R5	The User Manager must provide a registration form which lets the user state the desired role	S1, S2	OK
UM-R6	The user must be notified when he's successfully registered	S1, S2	OK

6.7 CONVERTER

Functional requirements for the IF Converter have been gathered considering the outcomes of the ST4RT project and the state of the art discussed in D4.1, the scenarios defined in D5.1 and refined in D5.4 as a result of the work carried out in WP3, and the automation workflows described in D4.2 and D4.3.

As discussed in D5.3, the main functional requirements for the Converter were already satisfied in C-Rel. In F-Rel, we mainly focused on improving the performance and scalability of the solution. New requirements defined and fulfilled for F-Rel are CF-R14, CF-R15, CF-R16.

Table 65 Functional requirements for the SPRINT Converter framework

ID	Description	Scenario	F-Rel status
CF-R1	The Converter framework must support creating batch converters.	D5.4 S5	OK
CF-R2	The Converter framework must support creating service mediators.	D5.4 S6	OK
CF-R3	The Converter framework must be able to access, process and produce different data formats.	D5.4 S5, S6	OK
CF-R4	The Converter framework must support the definition of conversion pipelines through a configuration file.	D5.4 S8	OK
CF-R5	The Converter framework must integrate the ST4RT annotations for lifting and lowering.	D4.1	OK
CF-R6	The Converter framework must support lifting based on declarative mappings.	D4.1	OK
CF-R7	The Converter framework must support lowering based on declarative mappings.	D4.1	OK
CF-R8	The Converter framework must support the definition of preprocessing/custom steps in the conversion pipeline.	D4.1	OK
CF-R9	The Converter framework must support data enrichment considering external data sources during the conversion.	D4.1	OK
CF-R10	The Converter framework must support inference enrichment considering a set of ontologies.	D4.1	OK

CF-R11	The Converter framework must be configurable to run as a standalone jar.	D5.4 S5, S8	OK
CF-R12	The Converter framework must support the definition of endpoints to request a runtime conversion.	D5.4 S6	OK
CF-R13	The Converter framework must be deployable as a Docker image.	D4.2, D5.4 S8, S9	OK
CF-R14	The Converter framework must be deployable as a scalable service load-balancing multiple instances.	D4.3, D5.4 S9	OK
CF-R15	The Converter framework must be configurable through an external configuration and to access external resources provided by a Resolver.	D5.4 S10	OK
CF-R16	The Converter framework must be configurable to use a remote repository for the RDF graph.	D5.3	OK

7. ANNEX B: Additional Data Performance and Scalability Evaluation

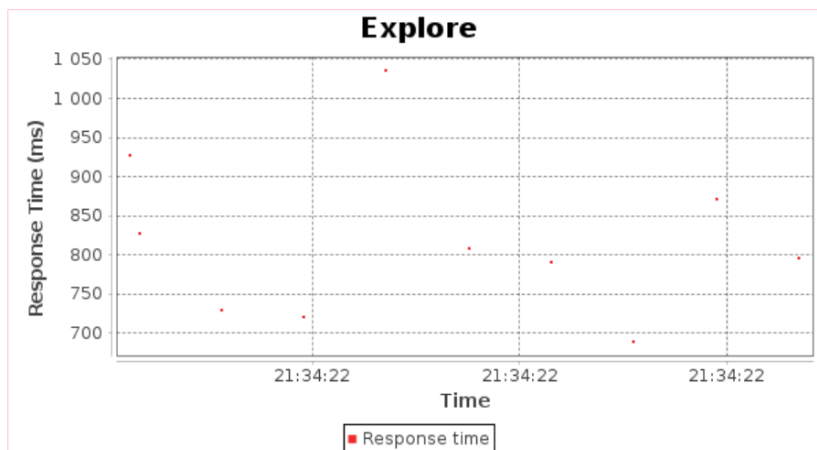
This annex contains additional data collected in the performance and scalability evaluation discussed in Section 3.

7.1 TC8 – SCALABILITY TESTING FOR ASSET MANAGER

Detailed data obtained in the performed testing activities for TC8.

Table 66 TC8 Summary of results obtained for Asset Manager

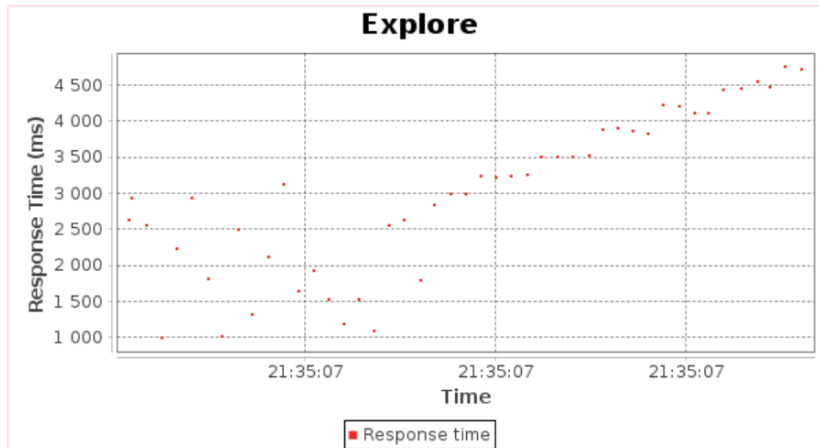
Number of HTTP requests (<i>N</i>)	Avg Time of processing <i>N</i> HTTP Requests [ms]
10	819
50	2 874
100	5 685
150	8 999
200	11 297
500	Not completely processed (24,4% of Errors 502)



URI: Explore

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Explore	10 +0	819 0	689 0	796 0	927 0	1036 0	200	0.0 % 0.0 %	90.11 0.0	901.12 0.0

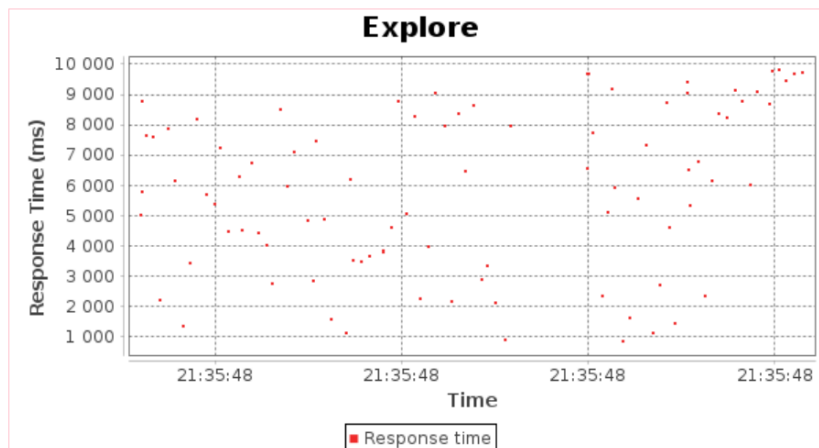
Figure 75 Asset Manager scalability test TC8: results with 10 parallel requests



URI: Explore

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Explore	50 +0	2874 0	998 0	2940 0	4439 0	4761 0	200	0.0 % 0.0 %	90.11 0.0	4505.62 0.0

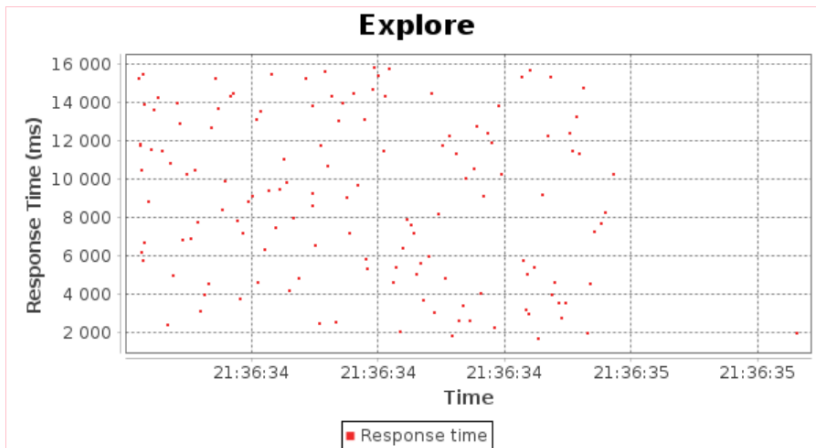
Figure 76 Asset Manager scalability test TC8: results with 50 parallel requests



URI: Explore

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Explore	100 +0	5685 0	855 0	5809 0	9112 0	9821 0	200	0.0 % 0.0 %	90.11 0.0	9011.23 0.0

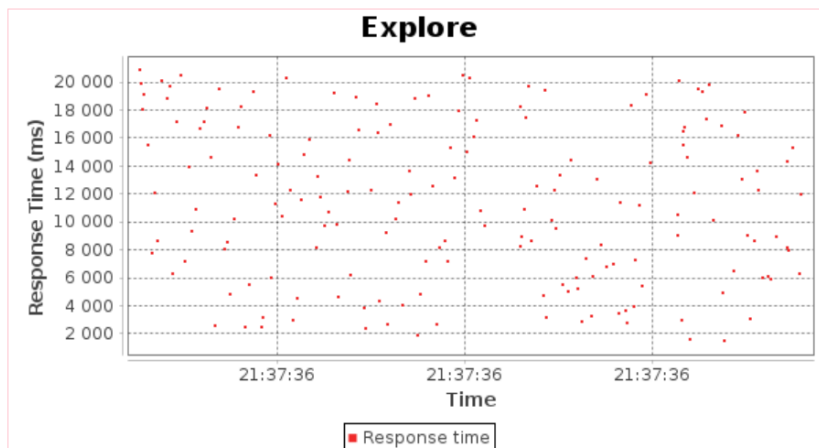
Figure 77 Asset Manager scalability test TC8: results with 100 parallel requests



URI: Explore

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Explore	150 +0	8999 0	1696 0	9083 0	14470 0	15828 0	200	0.0 % 0.0 %	90.11 0.0	13516.85 0.0

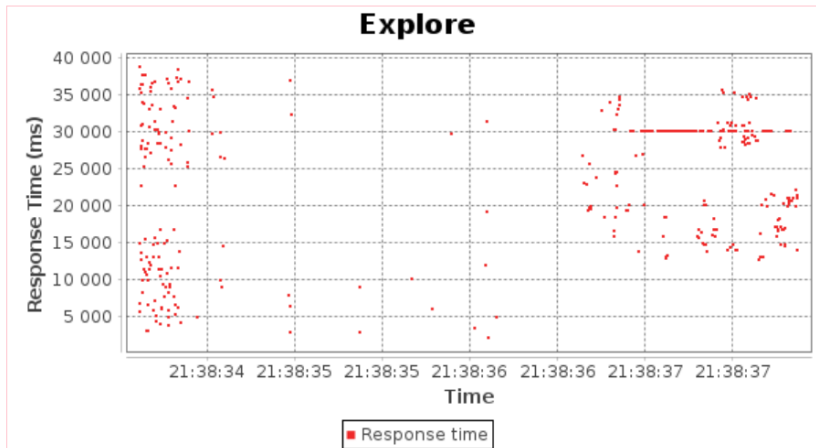
Figure 78 Asset Manager scalability test TC8: results with 150 parallel requests



URI: Explore

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Explore	200 +0	11297 0	1496 0	11349 0	18902 0	20905 0	200	0.0 % 0.0 %	90.11 0.0	18022.46 0.0

Figure 79 Asset Manager scalability test TC8: results with 200 parallel requests



URI: Explore

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Explore	500 ⁺⁰	22142 ⁰	2133 ⁰	25552 ⁰	32418 ⁰	38829 ⁰	200,502	24.2 % ^{0.0} %	68.35 ^{0.0}	34175.01 ^{0.0}

Figure 80 Asset Manager scalability test TC8: results with 500 parallel requests

7.2 TC10 - PERFORMANCE TESTING FOR RUNTIME DATA/MESSAGE CONVERSION

Detailed data obtained in the performed testing activities for TC10.

7.2.1 Annotation approach

Type 1 – FSM-to-918 request

Table 67 TC10 Complete results for Type 1 request

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	16	356	372
2	19	109	128
3	14	124	138
4	13	145	158
5	14	118	132
6	13	143	156
7	13	107	120
8	13	168	181
9	13	102	115
10	15	99	114
Average	14,3	147,1	161,4

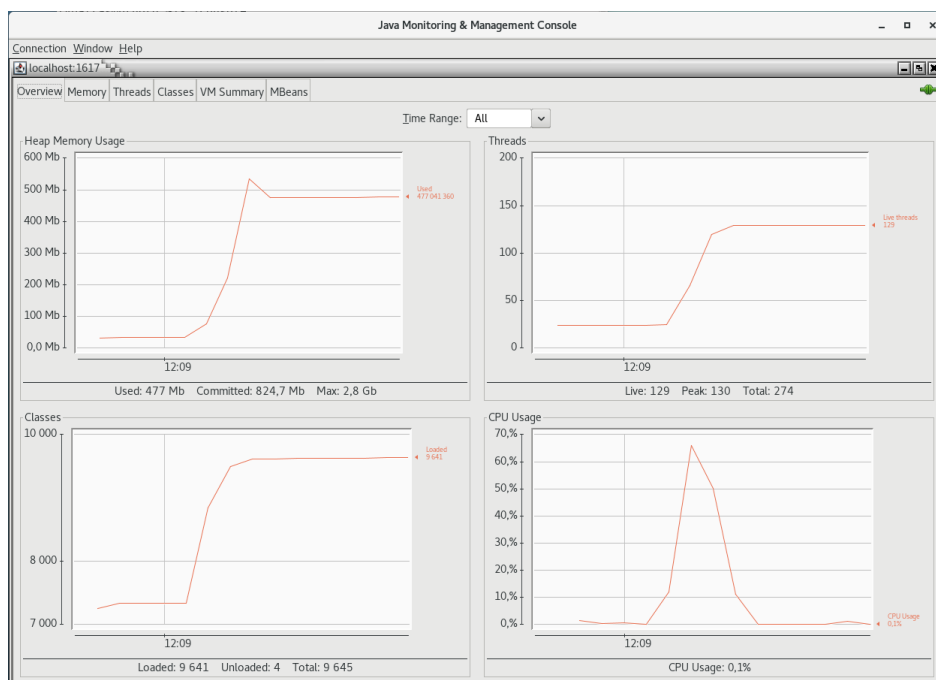


Figure 81 TC10 JConsole monitoring for Type 1 request

Type 1 – 918-to-FSM response

Table 68 TC10 Complete results for Type 1 response

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	5	1236	1241
2	6	1184	1190
3	5	1253	1258
4	6	1194	1200
5	4	1249	1253
6	4	1306	1310
7	5	1220	1225
8	4	1169	1173
9	5	1228	1233
10	4	1776	1780
Average	4,8	1281,5	1286,3

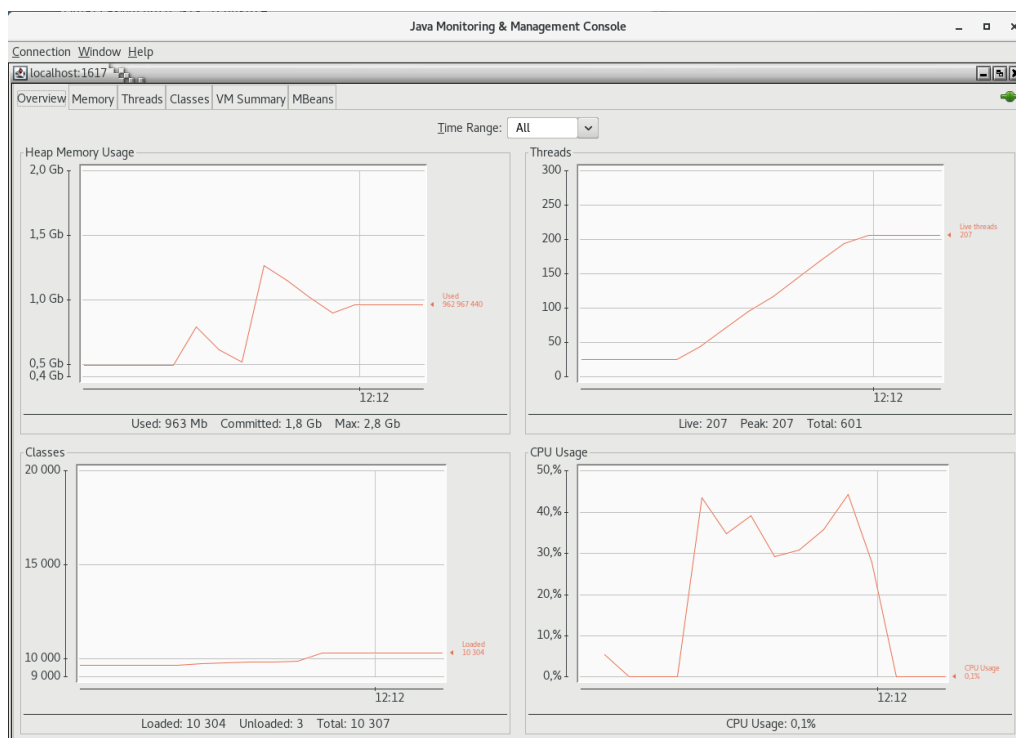


Figure 82 TC10 JConsole monitoring for Type 1 response

Type 2 – FSM-to-918 request

Table 69 TC10 Complete results for Type 2 request

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	7	78	85
2	9	79	88
3	8	78	86
4	7	79	86
5	8	78	86
6	8	77	85
7	9	111	120
8	10	80	90
9	8	115	123
10	9	83	92
Average	8,3	85,8	94,1

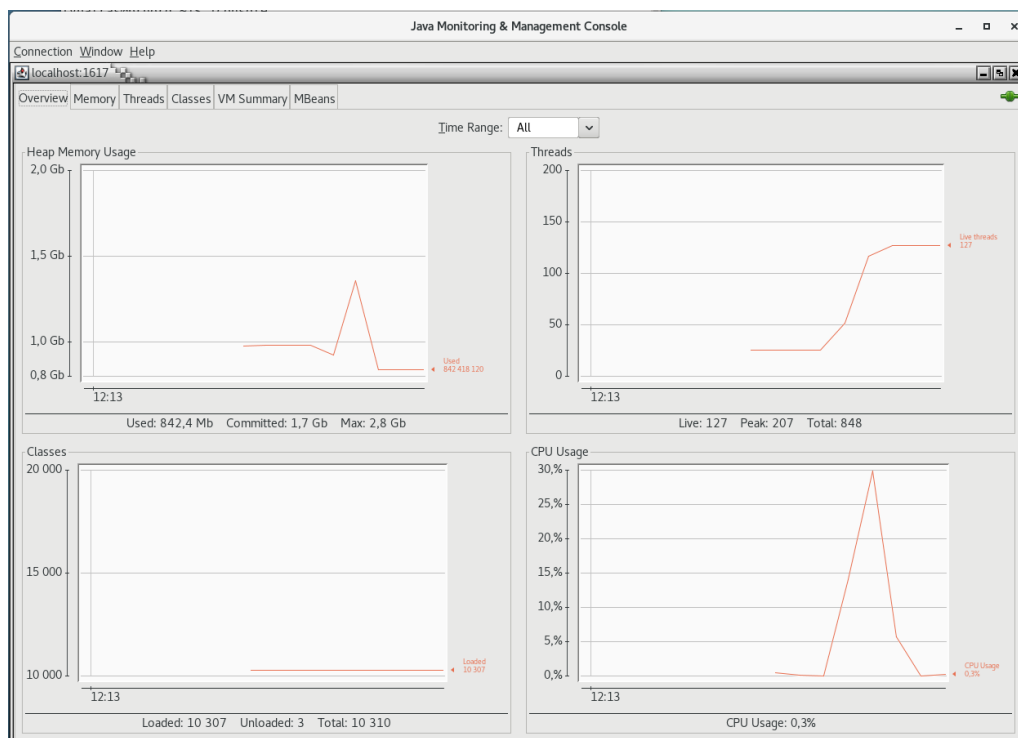


Figure 83 TC10 JConsole monitoring for Type 2 request

Type 2 - 918-to-FSM response

Table 70 TC10 Complete results for Type 2 response

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	3	1191	1194
2	4	1153	1157
3	4	1206	1210
4	4	1170	1174
5	5	1184	1189
6	5	1224	1229
7	4	1184	1188
8	5	1156	1161
9	4	1104	1108
10	4	1259	1263
Average	4,2	1183,1	1187,3

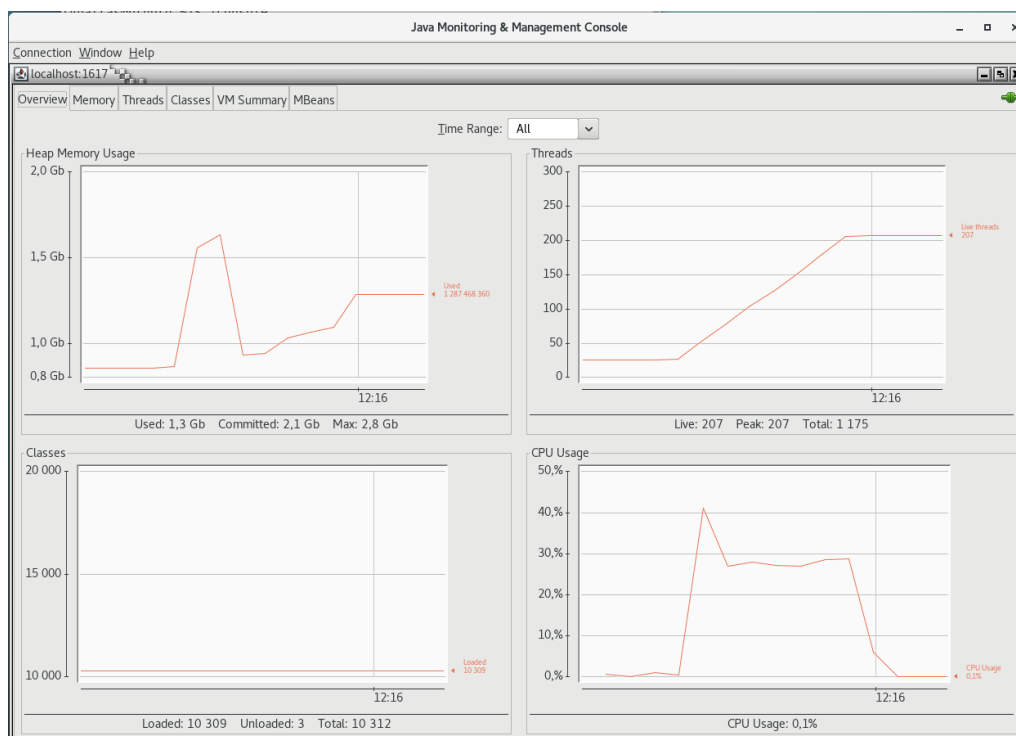


Figure 84 TC10 JConsole monitoring for Type 2 response

Type 3 - FSM-to-918 request

Table 71 TC10 Complete results for Type 3 request

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	7	78	85
2	7	106	113
3	8	77	85
4	9	85	94
5	7	89	96
6	8	77	85
7	8	96	104
8	8	82	90
9	7	78	85
10	9	78	87
Average	7,8	84,6	92,4

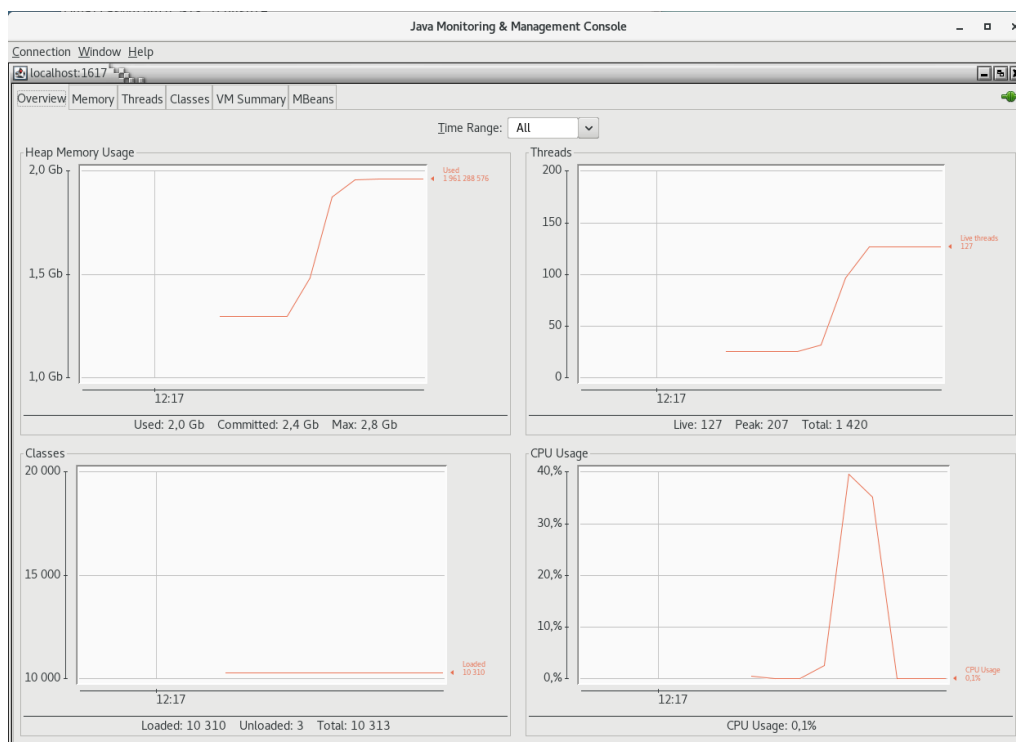


Figure 85 TC10 JConsole monitoring for Type 3 request

Type 3 - 918-to-FSM response

Table 72 TC10 Complete results for Type 3 response

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	4	1244	1248
2	3	1165	1168
3	3	1158	1161
4	5	1196	1201
5	4	1142	1146
6	4	1137	1141
7	4	1197	1201
8	3	1156	1159
9	3	1149	1152
10	3	2489	2492
Average	3,6	1303,3	1306,9

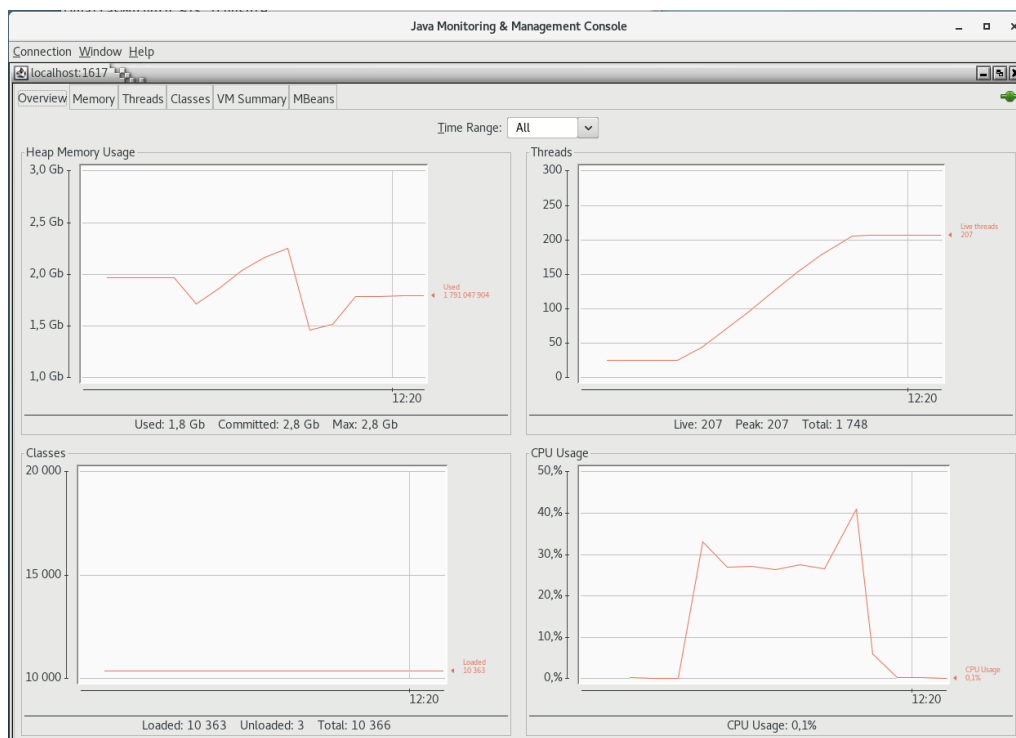


Figure 86 TC10 JConsole monitoring for Type 3 response

Type 4 - FSM-to-918 request

Table 73 TC10 Complete results for Type 4 request

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	12	77	89
2	7	76	83
3	7	76	83
4	7	112	119
5	8	141	149
6	6	75	81
7	7	75	82
8	8	75	83
9	8	100	108
10	11	122	133
Average	8,1	92,9	101

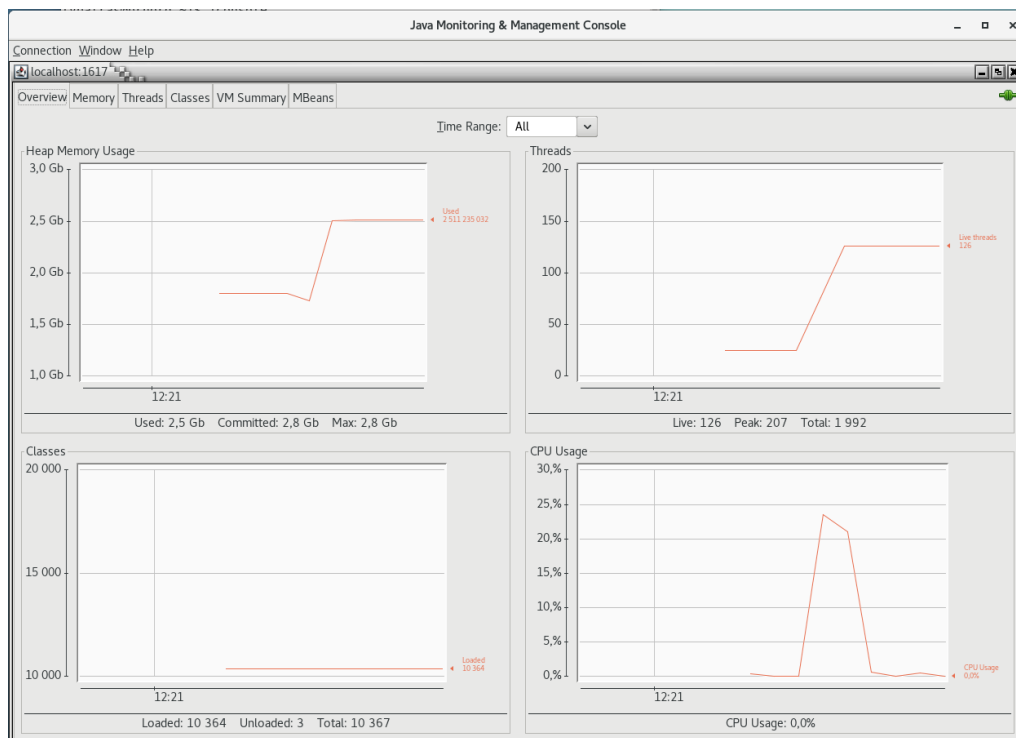


Figure 87 TC10 JConsole monitoring for Type 4 request

Type 4 – 918-to-FSM response

Table 74 TC10 Complete results for Type 4 response

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	3	1131	1134
2	3	1168	1171
3	3	1153	1156
4	4	1191	1195
5	4	1188	1192
6	4	1194	1198
7	3	1216	1219
8	3	1255	1258
9	6	1139	1145
10	4	1203	1207
Average	3,7	1183,8	1187,5

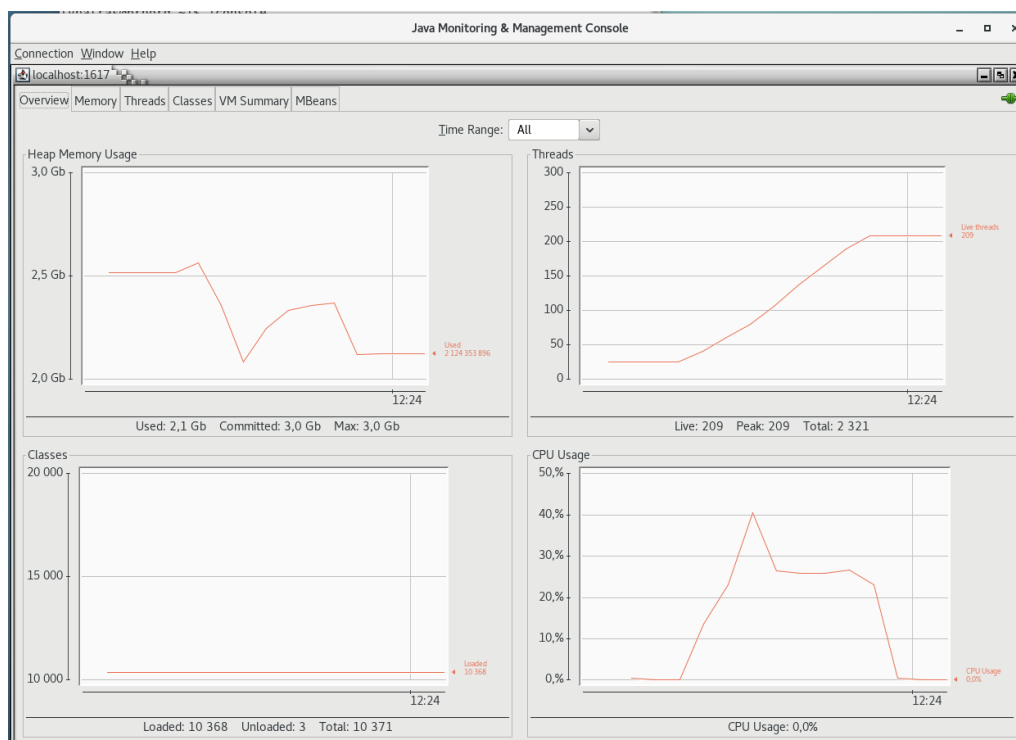


Figure 88 TC10 JConsole monitoring for Type 4 response

Type 5 – FSM-to-918 request

Table 75 TC10 Complete results for Type 5 request

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	7	81	88
2	8	85	93
3	8	77	85
4	8	78	86
5	36	77	113
6	8	103	111
7	10	80	90
8	9	105	114
9	8	76	84
10	11	78	89
Average	11,3	84	95,3

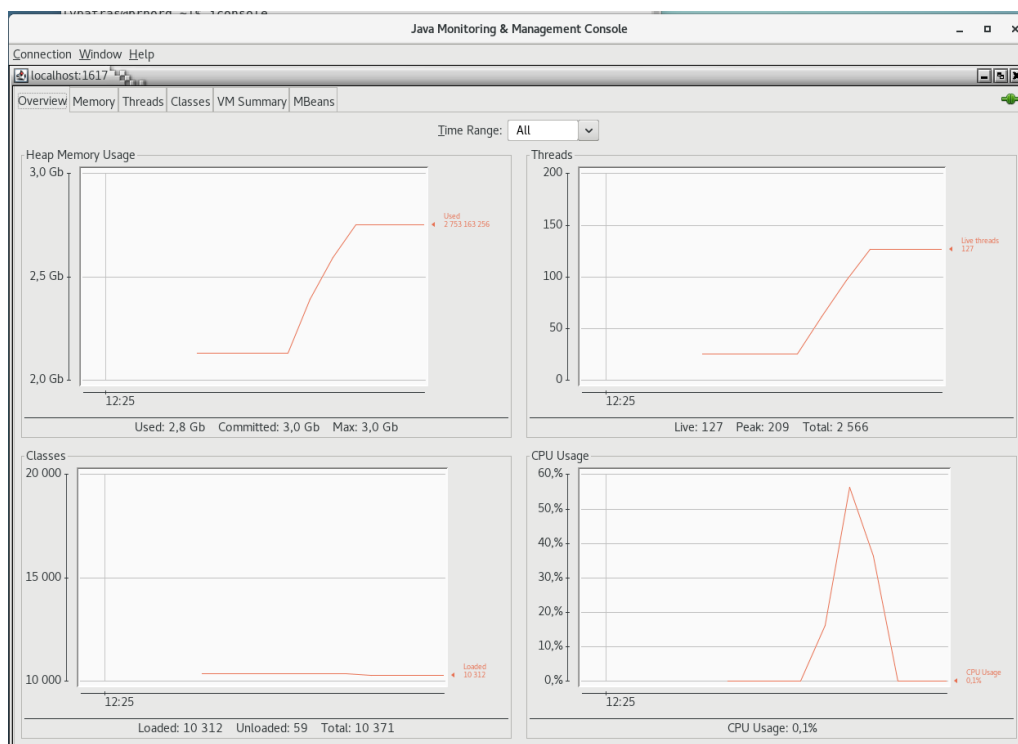


Figure 89 TC10 JConsole monitoring for Type 5 request

Type 5 – 918-to-FSM response

Table 76 TC10 Complete results for Type 5 response

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	4	2255	2259
2	6	1961	1967
3	4	1986	1990
4	4	2170	2174
5	5	1986	1991
6	4	1998	2002
7	4	1953	1957
8	5	1105	1110
9	3	2024	2027
10	4	1938	1942
Average	4,3	1937,6	1941,9

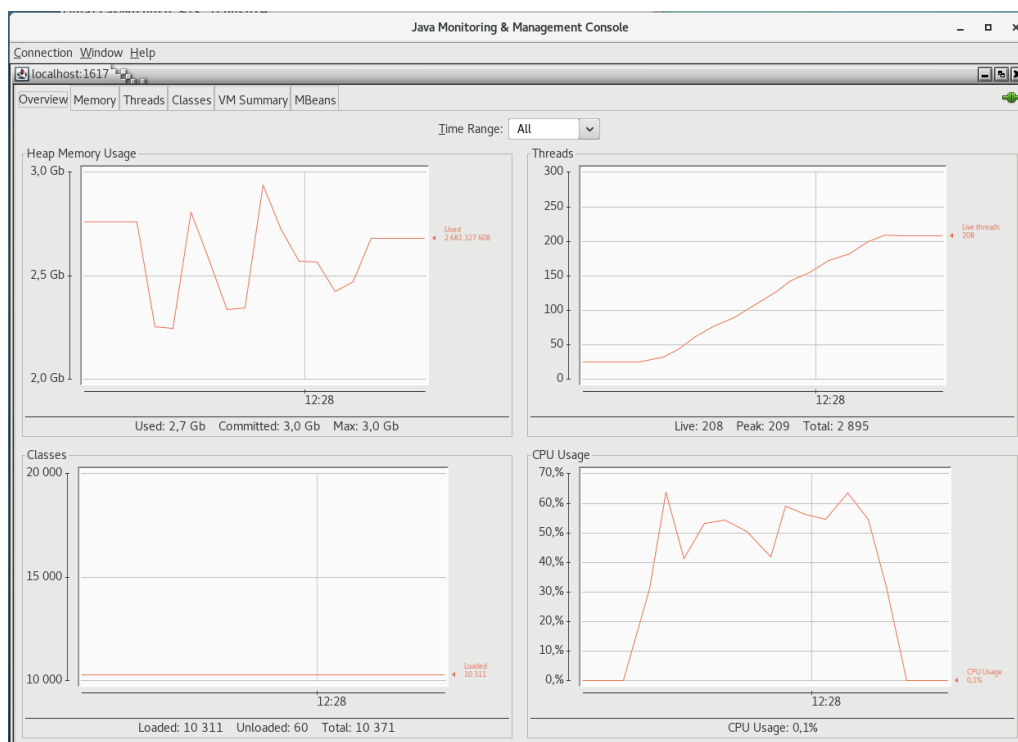


Figure 90 TC10 JConsole monitoring for Type 5 response

7.2.2 Declarative approach

CREL-M

Table 77 TC10 Complete results for VBB request with *crel-m* configuration

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
Cold start	1979	326	2305
Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	712	33	745
2	811	29	840
3	737	28	765
4	745	28	773
5	682	29	711
6	708	28	736
7	666	25	691
8	689	26	715
9	671	26	697
10	693	25	718
Average 1-10	711	28	739

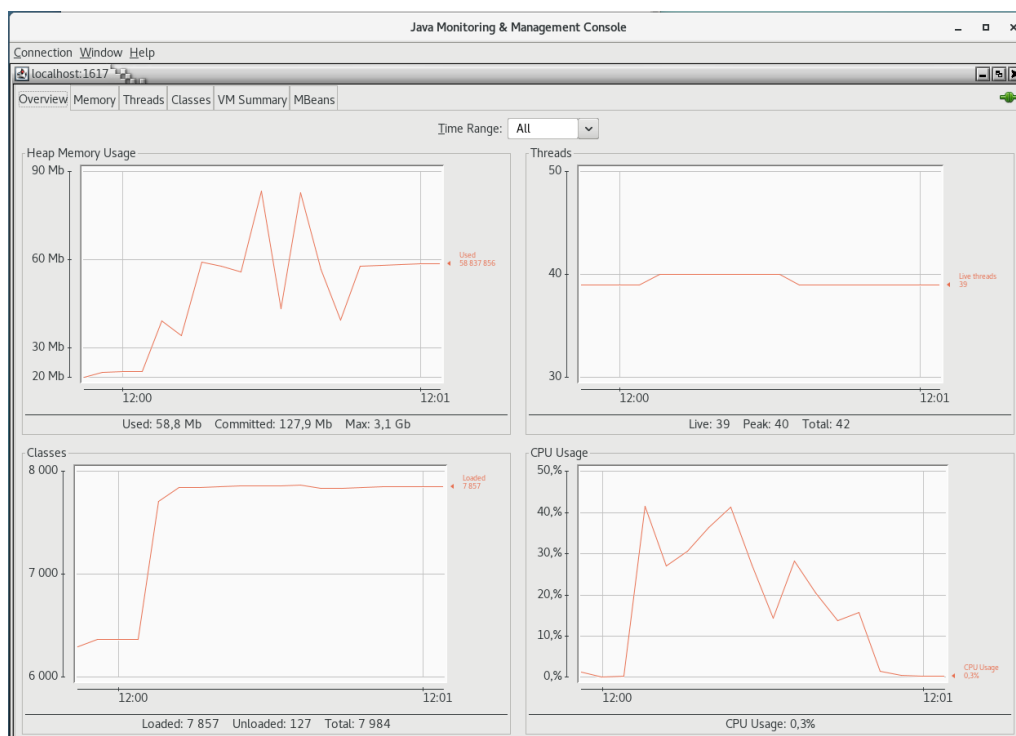


Figure 91 TC10 JConsole monitoring for VBB request with *crel-m* configuration

FREL-M-0

Table 78 TC10 Complete results for VBB request with *frel-m-0* configuration

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
Cold start	1884	344	2228
Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	111	31	142
2	141	31	172
3	141	29	170
4	192	29	221
5	138	28	166
6	137	28	165
7	135	32	167
8	136	28	164
9	135	35	170
10	101	26	127
Average 1-10	137	30	166

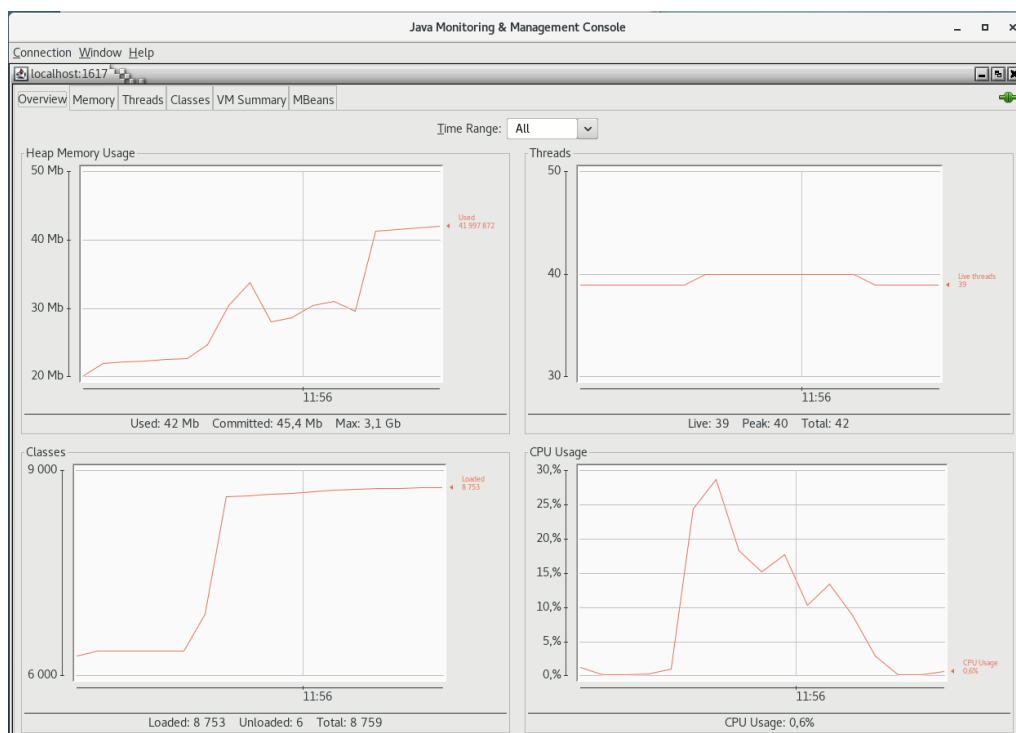


Figure 92 TC10 JConsole monitoring for VBB request with *frel-m-0* configuration

FREL-M-1

Table 79 TC10 Complete results for VBB request with *frel-m-1* configuration

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
Cold start	1881	351	2232
Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	111	32	143
2	119	34	153
3	108	33	141
4	109	30	139
5	113	32	145
6	79	29	108
7	110	30	140
8	108	30	138
9	107	30	137
10	109	30	139
Average 1-10	107	31	138

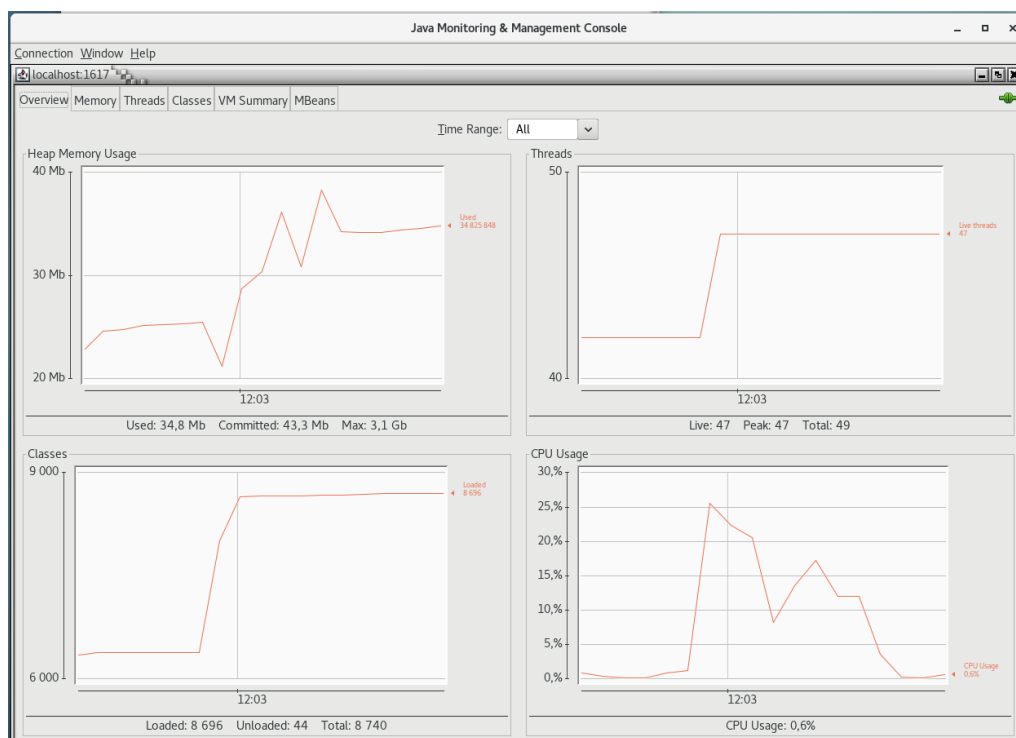


Figure 93 TC10 JConsole monitoring for VBB request with *frel-m-1* configuration

FREL-M-2

Table 80 TC10 Complete results for VBB request with *frel-m-2* configuration

Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
Cold start	1491	392	1883
Msg. order	Lifting time[ms]	Lowering time [ms]	Total time [ms]
1	133	44	177
2	125	46	171
3	128	43	171
4	143	47	190
5	135	42	177
6	128	42	170
7	124	39	163
8	118	39	157
9	121	40	161
10	96	30	126
Average 1-10	125	41	166

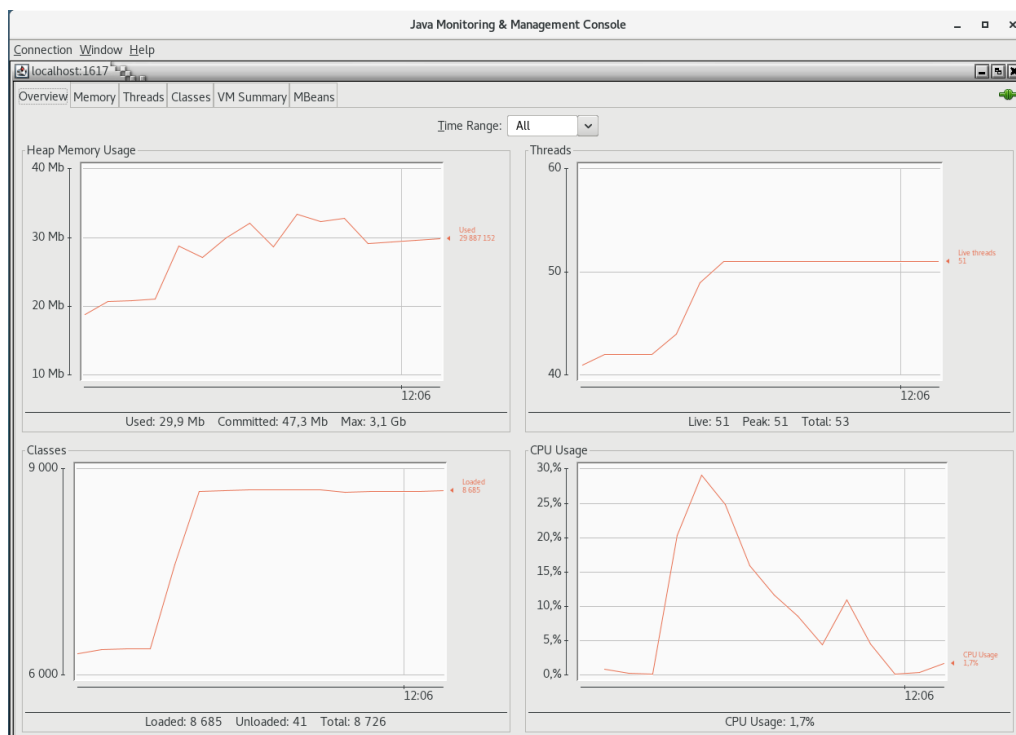


Figure 94 TC10 JConsole monitoring for VBB request with *frel-m-2* configuration

7.3 TC11 - SCALABILITY TESTING FOR BATCH DATA CONVERSION

Table 81 compares the tests performed with the 5-csv, 5-json and 5-xml datasets considering the *crel* and *frel-4* configurations with the GTFS-Madrid-Bench.

Table 81 Tests for TC11 with 5-csv, 5-json and 5-xml dataset

	CR	LPC	SRF	Conversion time	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
5-csv-crel	-	-	-	164.95	160.74	4.21	7.86	158.21
5-csv-frel-4	X	LS	X	55.37	49.89	5.48	15.35	443.88
5-json-crel	-	-	-	659.11	654.49	4.62	7.24	117.87
5-json-frel-0				394.21	389.85	4.36	5.30	122.26
5-json-frel-4	X	LS	X	519.64	515.58	4.06	11.48	262.55
5-xml-crel	-	-	-	TO	-	-	-	-
5-xml-frel-4	X	LS	X	123.29	119.23	4.05	16.72	439.48

Table 82 compares the tests performed with the 10-csv, 10-json and 10-xml datasets considering the *crel* and *frel-4* configurations with the GTFS-Madrid-Bench.

Table 82 Tests for TC11 with 10-csv, 10-json and 10-xml dataset

	CR	LPC	SRF	Conversion time	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
10-csv-crel	-	-	-	544.41	536.9	7.51	9.47	140.82
10-csv-frel-4	X	LS	X	154.13	146.9	7.22	17.13	457.51
10-json-crel	-	-	-	2471.29	2463.67	7.63	9.94	110.52
10-json-frel-0				1467.70	1460.03	7.67	7.79	110.83
10-json-frel-4	X	LS	X	2132.18	2124.95	7.23	15.09	250.97
10-xml-crel	-	-	-	TO	-	-	-	-
10-xml-frel-4	X	LS	X	434.05	426.97	7.08	16.36	426.63

Table 83 compares the tests performed with the 50-csv, 50-json and 50-xml datasets considering the *crel* and *frel-4* configurations with the GTFS-Madrid-Bench.

Table 83 Tests for TC11 with 50-csv, 50-json and 50-xml dataset

	CR	LPC	SRF	Conversion time	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
50-csv-crel	-	-	-	11624.45	11583.49	40.97	18.84	185.56
50-csv-frel-4	X	LS	X	3441.67	3407.39	34.28	18.58	516.51
50-json-crel	-	-	-	66003.36	65959.54	43.82	18.6	176.15
50-json-frel-0				34901.65	34861.97	39.68	18.54	153.97
50-json-frel-4	X	LS	X	54074.9	54040.82	34.08	18.44	328.94
50-xml-crel	-	-	-	TO	-	-	-	-
50-xml-frel-4	X	LS	X	12648.65	12614.62	34.03	18.44	540.01

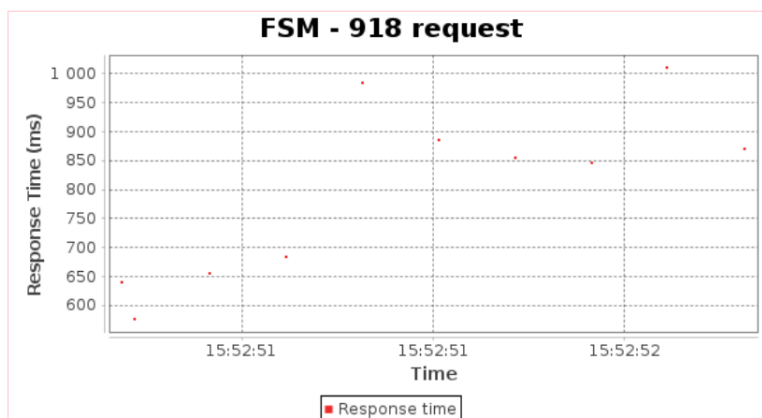
7.4 TC12 - SCALABILITY TESTING FOR RUNTIME DATA/MESSAGE DATA CONVERSION

7.4.1 Annotation approach

The test message Type 1 – FSM-to-918 request defined in the ST4RT project was executed for the *annotation approach* evaluation.

Table 84 TC12 Average response times for annotation approach

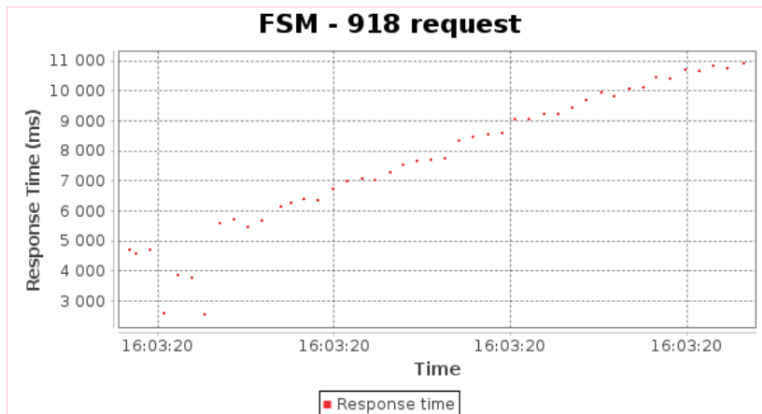
Number of requests (N)	Avg Time of processing N Requests [ms]
10	800
50	7 255
100	11 684
150	15 514
200	21 280
500	Not completely processed



URI: FSM - 918 request

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
FSM - 918 request	10 +0	800 0	576 0	846 0	985 0	1011 0	200	0.0 % 0.0 %	1.34 0.0	13.42 0.0

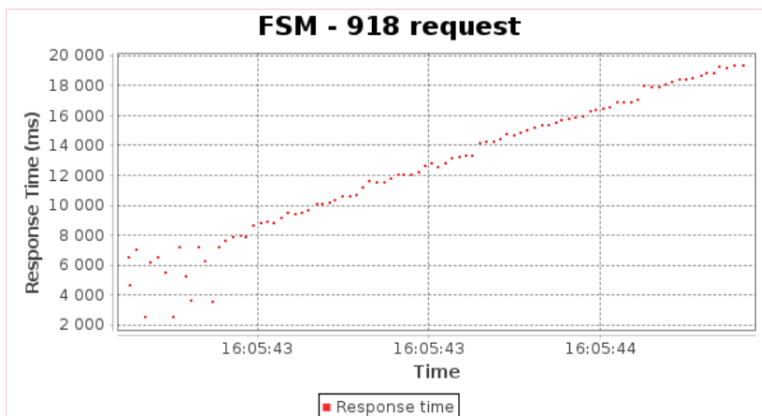
Figure 95 TC12 Response time 10 requests for annotation approach



URI: FSM - 918 request

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
FSM - 918 request	50 ⁺⁰	7255 ⁰	2554 ⁰	7313 ⁰	10459 ⁰	10932 ⁰	200	0.0 % ^{0.0} %	1.34 ^{0.0}	67.05 ^{0.0}

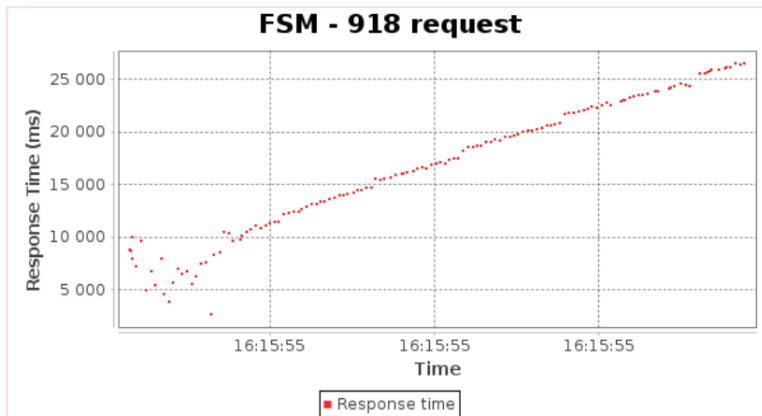
Figure 96 TC12 Response time 50 requests for annotation approach



URI: FSM - 918 request

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
FSM - 918 request	100 ⁺⁰	11684 ⁰	2523 ⁰	12035 ⁰	18251 ⁰	19355 ⁰	200	0.0 % ^{0.0} %	1.34 ^{0.0}	134.13 ^{0.0}

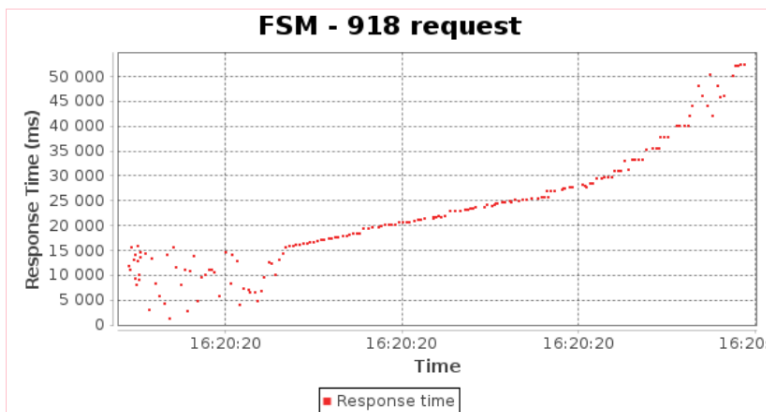
Figure 97 TC12 Response time 100 requests for annotation approach



URI: FSM - 918 request

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
FSM - 918 request	150 ⁺⁰	15514 ⁰	1035 ⁰	16008 ⁰	24305 ⁰	26511 ⁰	200	0.0 % 0.0 %	1.37 ^{0.0}	206.15 ^{0.0}

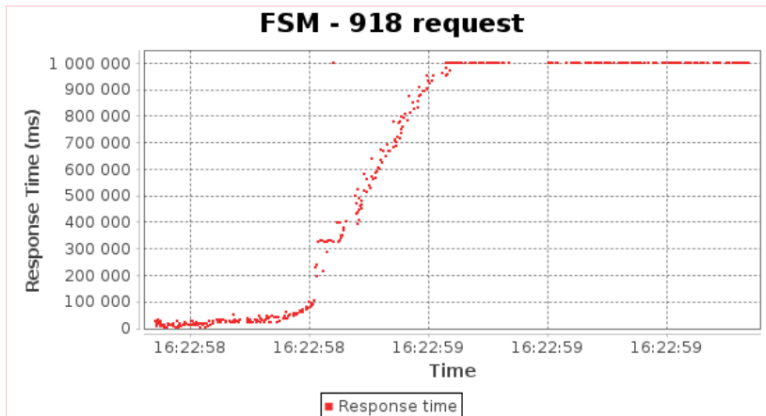
Figure 98 TC12 Response time 150 requests for annotation approach



URI: FSM - 918 request

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
FSM - 918 request	200 ⁺⁰	21280 ⁰	1171 ⁰	20107 ⁰	37903 ⁰	52337 ⁰	200	0.0 % 0.0 %	1.35 ^{0.0}	270.82 ^{0.0}

Figure 99 TC12 Response time 200 requests for annotation approach



URI: FSM - 918 request

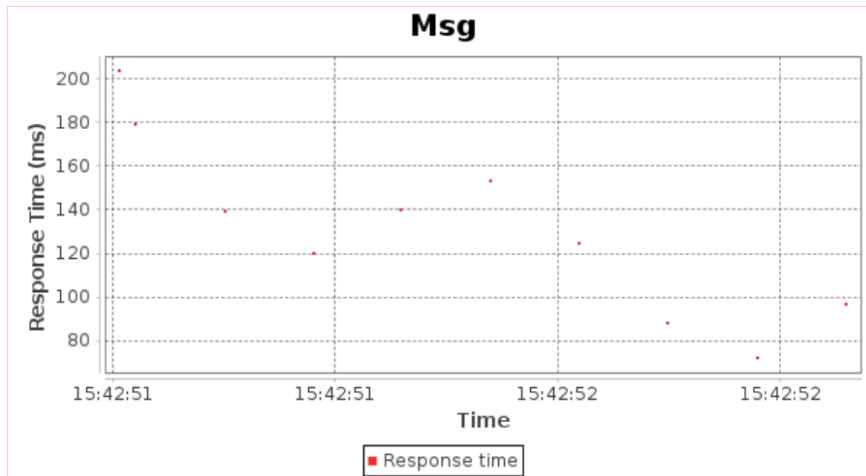
URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
FSM - 918 request	500 ⁺⁰	472341 ⁰	3134 ⁰	382413 ⁰	1000101 ⁰	1000579 ⁰	0,500,200	32.2 % ^{0.0} %	2.27 ^{0.0}	1137.48 ^{0.0}

Figure 100 TC12 Response time 500 requests for annotation approach

7.4.2 Declarative approach

Table 85 TC12 Average response times for declarative approach

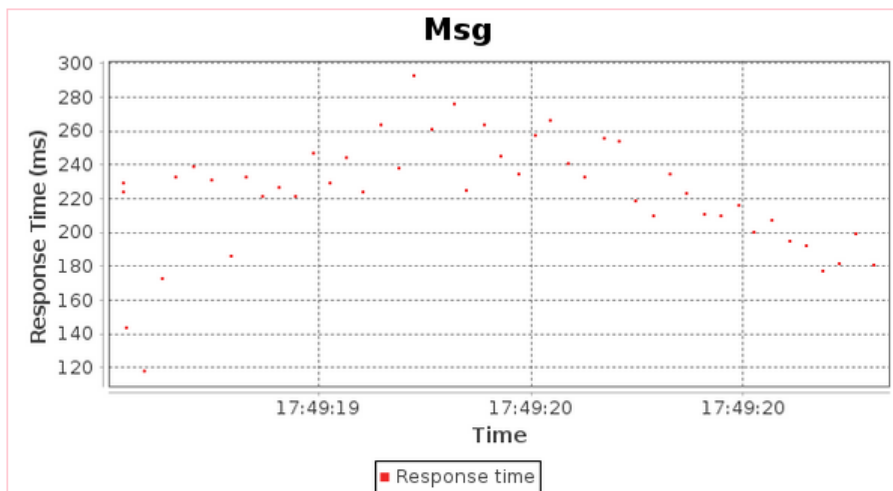
Number of requests (N)	Avg Time of processing of N Requests [ms]
10	131
50	219
100	775
150	1 663
200	1 918
500	3 926
1000	7 567
2500	21 114
5000	Not completely processed



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	10 ⁺⁰	131 ⁰	72 ⁰	125 ⁰	179 ⁰	204 ⁰	200	0.0 % ^{0.0 %}	29.76 ^{0.0}	297.61 ^{0.0}

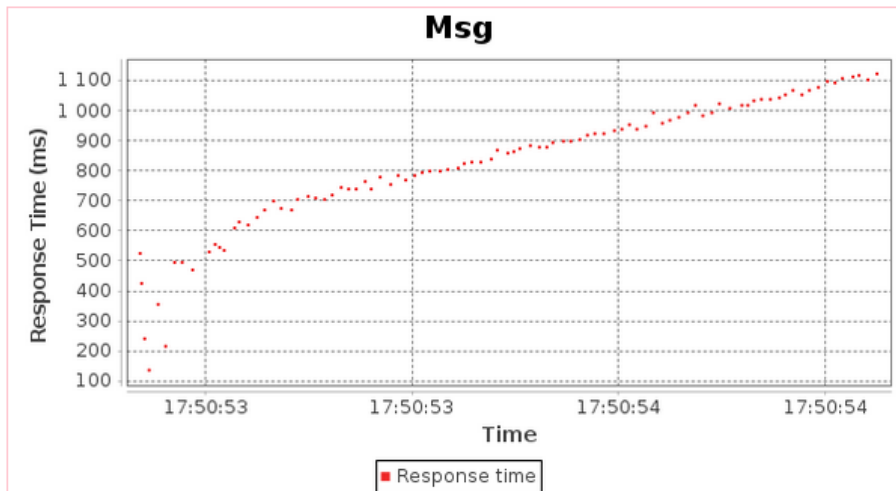
Figure 101 TC12 Response time 10 messages for declarative approach



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	50 ⁺⁰	219 ⁰	118 ⁰	224 ⁰	261 ⁰	293 ⁰	200	0.0 % ^{0.0 %}	29.76 ^{0.0}	1488.04 ^{0.0}

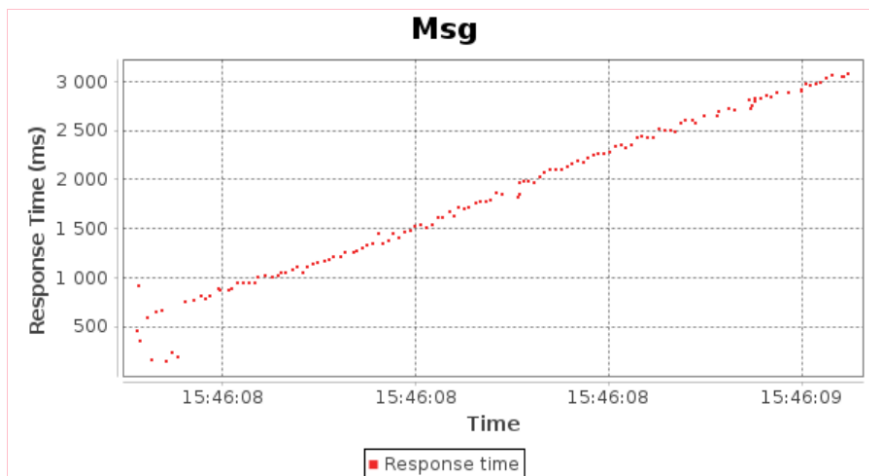
Figure 102 TC12 Response time 50 messages for declarative approach



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	100 +0	775 0	136 0	823 0	1054 0	1122 0	200	0.0 % 0.0 %	29.76 0.0	2976.07 0.0

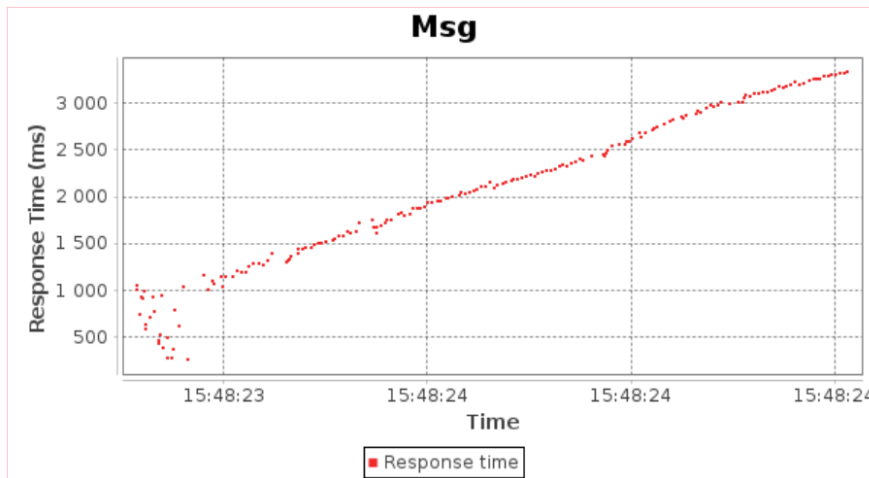
Figure 103 TC12 Response time 100 messages for declarative approach



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	150 +0	1663 0	147 0	1619 0	2850 0	3081 0	200	0.0 % 0.0 %	29.76 0.0	4464.11 0.0

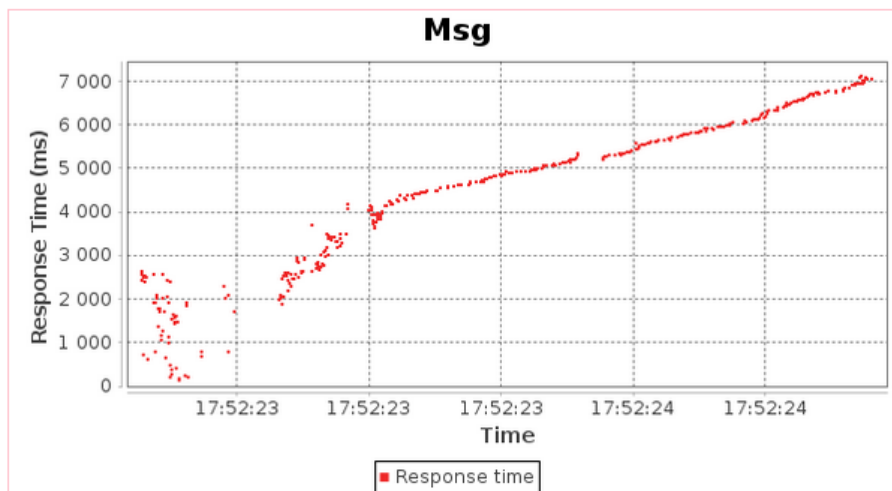
Figure 104 TC12 Response time 150 messages for declarative approach



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	200 ⁺⁰	1918 ⁰	132 ⁰	1962 ⁰	3120 ⁰	3340 ⁰	200	0.0 % ^{0.0} %	29.76 ^{0.0}	5952.15 ^{0.0}

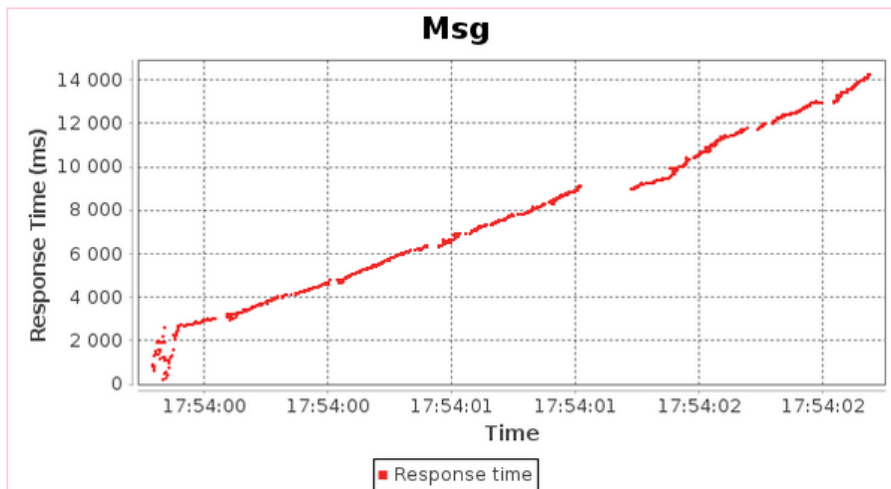
Figure 105 TC12 Response time 200 messages for declarative approach



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	500 ⁺⁰	3926 ⁰	137 ⁰	4108 ⁰	6475 ⁰	7108 ⁰	200	0.0 % ^{0.0} %	29.76 ^{0.0}	14880.37 ^{0.0}

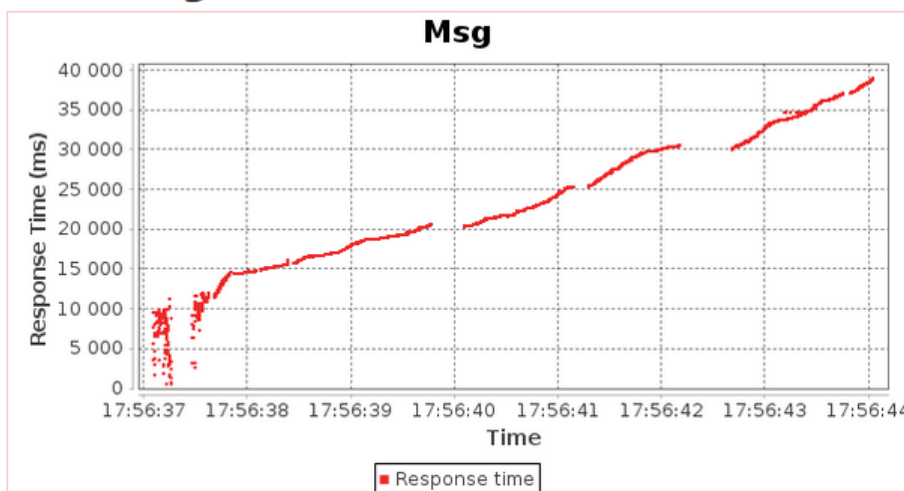
Figure 106 TC12 Response time 500 messages for declarative approach



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	1000 +0	7567 0	123 0	7595 0	12875 0	14234 0	200	0.0 % 0.0 %	29.76 0.0	29760.74 0.0

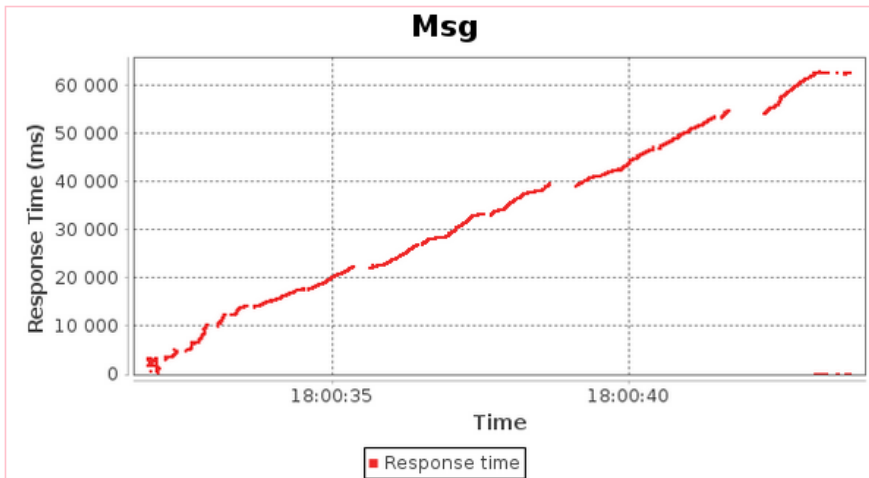
Figure 107 TC12 Response time 1 000 messages for declarative approach



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	2500 +0	21114 0	406 0	21257 0	35286 0	38927 0	200	0.0 % 0.0 %	29.76 0.0	74401.86 0.0

Figure 108 TC12 Response time 2 500 messages for declarative approach



URI: Msg

URI	Samples	Average (ms)	Min(ms)	Median(ms)	Line 90.0(ms)	Max(ms)	Http Code	Errors (%)	Average (KB)	Total (KB)
Msg	4618 ⁺⁰	32336 ⁰	0 ⁰	32816 ⁰	56703 ⁰	62876 ⁰	0,500,200	3.097 % ^{0.0} %	28.91 ^{0.0}	133488.95 ^{0.0}

Figure 109 TC12 Response time 5 000 messages for declarative approach

8. BIBLIOGRAPHY

- [1] ST4RT, “D5.1 - Requirements for the Demo,” 2018. [Online]. Available:
<http://www.st4rt.eu/Page.aspx?CAT=DELIVERABLES&IdPage=3cd536b1-4a34-4432-8987-115810d01dd3>.
- [2] M. Scrocca, M. Comerio, A. Carenini and I. Celino, “Turning Transport Data to Comply with EU Standards while Enabling a Multimodal Transport Knowledge Graph,” in *International Semantic Web Conference (ISWC 2020)*, Online, 2020.