



## D3.3 Technical Specification

Authors: UVA(lead), TUDA, ASC, INESC, TIE, UVI, ISOFT

**Delivery Date:**

2012-09-24

**Due Date:**

2012-08-31

**Dissemination Level:**

PUBLIC

This deliverable presents the technical specification of the ADVENTURE platform. It defines the ADVENTURE platform from a technical point of view and shows how the components cooperate with each other in order to provide the desired functionality to the end user. It also reaches a consensus on technical decision of choosing suitable technologies/ tools for the components based on ADVENTURE vision.



D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>1</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Document History	
<b>Draft Version</b>	<p>V0.1, UVA, November 9<sup>th</sup>, 2011</p> <p>V0.2, UVA, January 4<sup>th</sup>, 2012</p> <p>V0.3, UVA, April 23<sup>rd</sup>, 2012</p> <p>V0.4, UVA, May 10<sup>th</sup>, 2012</p> <p>V0.5, UVA, May 16<sup>th</sup>, 2012</p> <p>V1.1, UVA, ASC, TIE, TUDA, June 12<sup>th</sup>, 2012</p> <p>V1.2, UVA, ASC, TIE, TUDA, ISOFT, INESC, June 24<sup>th</sup>, 2012</p> <p>V1.3, UVA, ASC, TIE, TUDA, ISOFT, INESC, June 25<sup>th</sup>, 2012</p> <p>V1.4, UVA, ASC, TIE, TUDA, ISOFT, INESC, July 26<sup>th</sup>, 2012</p> <p>V1.5, UVA, ASC, TIE, TUDA, ISOFT, INESC, July 30<sup>th</sup>, 2012</p> <p>V1.6, UVA, ASC, TIE, TUDA, ISOFT, INESC, July 31<sup>st</sup>, 2012</p> <p>V1.7, UVA, ASC, TIE, TUDA, ISOFT, INESC, UVI, August 1<sup>st</sup>, 2012</p> <p>V1.8, UVA, ASC, TIE, TUDA, ISOFT, INESC, UVI, August 5<sup>th</sup>, 2012</p> <p>V1.9, TIE, August 21<sup>st</sup>, 2012</p> <p>V2.0, UVA, ASC, TIE, TUDA, ISOFT, INESC, UVI, September 7<sup>th</sup>, 2012</p> <p>V2.1, ISOFT, September 13<sup>th</sup>, 2012</p> <p>V2.2, UVA, ASC, TIE, TUDA, ISOFT, INESC, UVI, September 21<sup>st</sup>, 2012</p>
<b>Contributions</b>	<p>UVA</p> <ul style="list-style-type: none"> <li>- Ahm Shamsuzzoha</li> <li>- Yuqiuge Hao</li> </ul> <p>TUDA</p> <ul style="list-style-type: none"> <li>- Dieter Schuller</li> <li>- Sebastian Zöller</li> <li>- Ronny Hans</li> </ul> <p>ASC</p> <ul style="list-style-type: none"> <li>- Sven Abels</li> <li>- Tim Dellas</li> <li>- Rafael Karbowski</li> </ul> <p>INESC</p> <ul style="list-style-type: none"> <li>- Americo Azevedo</li> <li>- Filipe Ferreira</li> <li>- Jose Faria</li> </ul> <p>TIE</p> <ul style="list-style-type: none"> <li>- Juanvi Vidagany</li> <li>- Stuart Campbell</li> <li>- Vadim Petrenko</li> </ul>

	<p>UVI</p> <ul style="list-style-type: none"><li>- Juergen Mangler</li><li>- Tobias Hildebrandt</li></ul> <p>ISOFT</p> <ul style="list-style-type: none"><li>- Georgi Pavlov</li><li>- Irena Pavlova</li><li>- Velimir Manafov</li><li>- Atanas Manafov</li></ul>
<b>Internal Review 1</b>	Stuart Campbell, TIE, 7 <sup>th</sup> August, 2012
<b>Internal Review 2</b>	Irena Pavlova, Georgi Pavlov, ISOFT, September 13 <sup>th</sup> , 2012
<b>Final Version</b>	September 21 <sup>st</sup> , 2012

## Table of Contents

Executive Summary.....	13
1 Introduction.....	14
1.1 ADVENTURE Project Aims .....	14
1.2 Deliverable Purpose, Scope and Context.....	15
1.3 Document Status.....	16
1.4 Target Audience .....	16
1.5 Abbreviations and General Terms.....	17
1.6 Document Structure .....	17
2 General Description of Technical Specification .....	18
3 Technical Specification.....	23
3.1 Dashboard.....	24
3.1.1 Major Design Decisions.....	24
3.1.2 Technology Comparison and Selection.....	29
3.1.3 Technical Component Specification .....	37
3.1.4 Specification of Interfaces, Protocols and Formats .....	43
3.1.5 Summary.....	46
3.2 Cloud Storage .....	48
3.2.1 Major Design Decisions.....	48
3.2.2 Technology Comparison and Selection.....	49
3.2.3 Technical Component Specification .....	63
3.2.4 Specification of Interfaces, Protocols and Formats .....	70
3.2.5 Summary.....	81
3.3 Data Provisioning and Discovery (DPD).....	84
3.3.1 Major Design Decisions.....	84
3.3.2 Technology Comparison and Selection.....	88
3.3.3 Technical Component Specification .....	104
3.3.4 Specification of Interfaces, Protocols and Formats .....	115
3.3.5 Summary.....	127
3.4 Message Routing .....	129



3.4.1	Major Design Decisions.....	129
3.4.2	Technology Comparison and Selection.....	129
3.4.3	Technical Component Specification .....	139
3.4.4	Specification of Interfaces, Protocols and Formats .....	142
3.4.5	Summary.....	157
3.5	Transformation Service .....	159
3.5.1	Major Design Decisions.....	159
3.5.2	Technology Comparison and Selection.....	161
3.5.3	Technical Component Specification .....	164
3.5.4	Specification of Interfaces, Protocols and Formats .....	167
3.5.5	Data Extraction Service.....	176
3.5.6	Summary.....	190
3.6	Gateways .....	193
3.6.1	Major Design Decisions.....	193
3.6.2	Technology Comparison and Selection.....	195
3.6.3	Technical Component Specification .....	202
3.6.4	Specification of Interfaces, Protocols and Formats .....	207
3.6.5	Summary.....	221
3.7	Process Designer .....	223
3.7.1	Major Design Decisions.....	223
3.7.2	Technology Comparison and Selection.....	225
3.7.3	Technical Component Specification .....	235
3.7.4	Specification of Interfaces, Protocols and Formats .....	251
3.7.5	Summary.....	258
3.8	Smart Process Execution .....	260
3.8.1	Major Design Decisions.....	260
3.8.2	Technology Comparison and Selection.....	266
3.8.3	Technical Component Specification .....	282
3.8.4	Specification of Interfaces, Protocols and Formats .....	288
3.8.5	Summary.....	310

3.9	Process Monitoring.....	312
3.9.1	Major Design Decisions.....	312
3.9.2	Technology Comparison and Selection.....	313
3.9.3	Technical Component Specification .....	320
3.9.4	Specification of Interfaces, Protocols and Formats .....	328
3.9.5	Summary.....	331
3.10	Process Optimization .....	333
3.10.1	Major Design Decisions.....	333
3.10.2	Technology Comparison and Selection.....	334
3.10.3	Technical Component Specification .....	342
3.10.4	Specification of Interfaces, Protocols and Formats .....	345
3.10.5	Summary.....	352
3.11	Smart Object Integration .....	354
3.11.1	Major Design Decisions.....	354
3.11.2	Technology Comparison and Selection.....	355
3.11.3	Technical Component Specification .....	358
3.11.4	Specification of Interfaces, Protocols and Formats .....	360
3.11.5	Summary.....	363
4	Conclusion.....	364
Annex A	.....	365

## List of Figures

Figure 1 – Context of the Deliverable .....	16
Figure 2 – Architecture of the ADVENTURE Platform .....	18
Figure 3 – Overview of the Dashboard Homepage.....	25
Figure 4 – Process Design .....	26
Figure 5 – Process Management.....	27
Figure 6 – Partner Profile Tab .....	28
Figure 7 – Application Setup Tab .....	29
Figure 8 – Dashboard Component Structure .....	38
Figure 9 – Login and Session Management Sequence Diagram.....	41
Figure 10 – Create Components as Portlet .....	42
Figure 11 – Add Portlet in Portal Page .....	43
Figure 12 – Cloud Storage component structure .....	63
Figure 13 – Sequence diagram for bucket based access control .....	65
Figure 14 – Sequence diagram for bucket creation .....	66
Figure 15 – Sequence diagram for bucket deletion .....	67
Figure 16 – Sequence diagram for access rights request .....	68
Figure 17 – Sequence diagram for adding and updating access rights .....	69
Figure 18 – Sequence diagram for bucket querying .....	70
Figure 19 – Cloud Storage Message Interface .....	71
Figure 20 – DPD component structure .....	105
Figure 21 – Sequence diagram for Provide new member profile.....	112
Figure 22 – Sequence diagram for Search member profile .....	113
Figure 23 – Sequence diagram for Find services for activity .....	114
Figure 24 – DPD conceptual data model .....	126
Figure 25 – Message Routing component structure .....	139
Figure 26 – Activity Diagram Message Routing with Buffering .....	141
Figure 27 – Sequence diagram for Two Message Exchanges using Message Routing.....	142
Figure 28 – Transformation Service component structure .....	164

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>7</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Figure 29 – Transformation internal sequence diagram .....	166
Figure 30 – Transformation Service interfaces (provided and needed) .....	167
Figure 31 – Transformation request XML schema.....	170
Figure 32 – Transformation response schema .....	171
Figure 33 – Translator Interface request schema .....	172
Figure 34 – Translator Interface response schema .....	173
Figure 35 – Transformation operation description object model .....	174
Figure 36 - An extraction service call is defined as part of the process model .....	178
Figure 37 – Sequence Diagram - Data Extraction Service called from the execution environment after a gateway operation (as part of a designed process model).....	178
Figure 38 – Sequence Diagram - Data Extraction Service called from within a gateway operation .....	179
Figure 39 – Data extraction service component structure.....	181
Figure 40 – Data Extraction internal sequence diagram .....	182
Figure 41 – Data extraction Service interfaces (provided and needed) .....	183
Figure 42 – Data Extraction request schema.....	185
Figure 43 – Graphical representation of the XML schema .....	187
Figure 44 – Extraction operation description object model .....	189
Figure 45 – Gateway component structure.....	202
Figure 46 – Gateways, call/ response scenario .....	205
Figure 47 – Gateways, pushing message scenario .....	206
Figure 48 – Sequence Diagram - Data Extraction Service called from within a gateway operation .....	207
Figure 49 – Gateway interfaces (required and provided).....	207
Figure 50 – Gateway request schema .....	209
Figure 51 – Gateway response schema. ....	211
Figure 52 – Gateway services description data model .....	212
Figure 53 – TSB request schema for document sending.....	214
Figure 54 – TSB Response Schema .....	215
Figure 55 – Process Designer structure .....	235

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>8</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Figure 56 – Show Process Model Manager, list Process Models and filter sequence diagram.....	238
Figure 57 – Mock-up of the PMM main page.....	239
Figure 58 – New process model sequence diagram.....	239
Figure 59 – Show Process Template Manager and related actions sequence diagram.....	240
Figure 60 – New process model from template sequence diagram.....	241
Figure 61 – Edit process model sequence diagram.....	242
Figure 62 – Save process model sequence diagram.....	243
Figure 63 – Share process model sequence diagram .....	244
Figure 64 – New activity & activity configuration sequence diagram .....	247
Figure 65 – Assign profile classification sequence diagram .....	248
Figure 66 – New transformation activity sequence diagram .....	249
Figure 67 – New sub-process from existing process sequence diagram.....	251
Figure 68 – Process Model Designer interfaces (required and provided).....	252
Figure 69 – BPMN 2.0 diagram .....	254
Figure 70 – Replaceable Process Engine at the Core.....	261
Figure 71 – BPMN New Instance Spawning.....	264
Figure 72 – ADVENTURE Explicit Instance Spawning.....	265
Figure 73 – ADVENTURE SPE Component Structure .....	283
Figure 74 – Adaptation .....	286
Figure 75 – Forecasting & Simulation.....	287
Figure 76 – Process Instances – Structure of Interface.....	290
Figure 77 – SPE Event Model .....	301
Figure 78 – PM component structure .....	321
Figure 79 – Monitoring Conceptual Data Model .....	323
Figure 80 – Real Time Monitoring Mock-up.....	324
Figure 81 – Process Analytics Mock-up.....	326
Figure 82 – Rules Engine MockUp .....	327
Figure 83 – Sequence diagram for update order status .....	328
Figure 84 – Optimization Component Structure.....	342

Figure 85 – Sequence Diagram for Computing an optimized Invocation Plan.....	344
Figure 86 – ComputeInvocationPlan Example Request .....	348
Figure 87 – ComputeInvocationPlan Schema (part 1) .....	349
Figure 88 – ComputeInvocationPlan Schema (part 2) .....	350
Figure 89 – InvocationPlan Example Response .....	351
Figure 90 – InvocationPlan Schema.....	351
Figure 91 – Smart Object Integration component structure .....	358
Figure 92 – Sequence diagram for receiving and forwarding a query message .....	360
Figure 93 – Sequence diagram for forwarding a Smart Object message .....	360

## List of Tables

Table 1 – Main Enterprise Portal Platforms Summary .....	32
Table 2 – Comparison of technologies for Dashboard.....	34
Table 3 – Technology selection criteria and comparison of technologies for Structured Data Storage .....	52
Table 4 – Technology selection criteria and comparison of technologies for Semi- Structured Data Storage .....	54
Table 5 – Technology selection criteria and comparison of technologies for Semantic Data Storage .....	56
Table 6 – Technology selection criteria and comparison of technology for Binary Data Storage .....	59
Table 7 – Technology selection criteria for Data Provisioning and Discovery .....	88
Table 8 – Technology selection criteria and comparison of technologies for Data Provisioning and Discovery.....	93
Table 9 – Applicability of the data model employed by the reviewed technologies to the DPD domain model.....	95
Table 10 – Technology selection criteria and comparison of technologies for Message Routing.....	133
Table 11 – Comparison of different technologies/strategies for Gateways .....	198
Table 12 – Technology selection criteria and comparison of technologies for Process Designer .....	230
Table 13 – Technology selection criterion and comparison of technologies for Smart Process Execution Engine .....	272
Table 14 – Management & Execution Shaping Interface.....	291
Table 15 – Adaptation Interface.....	303
Table 16 – Forecasting & Simulation Interface .....	305
Table 17 – Specific parameters of technology selection criteria for Real Time Monitoring.....	313
Table 18 – Technology selection criterion and comparison of different technologies for Real Time Monitoring.....	316

Table 19 – Technology selection criterion and comparison of different technologies for Structured Data Storage .....	337
Table 20 – Technology selection criteria for Smart Object Integration .....	356



## Executive Summary

This deliverable (D3.3) takes as a basis the Global Architecture (D3.1) and the Functional Specification (D3.2) to highlight and to further elaborate the design of the ADVENTURE platform into a concrete implementable solution. As a result, the concrete operations, standard protocols and data exchange formats were specified, to ensure effective execution, communication and global interoperability among the thirteen ADVENTURE components.

The method used to achieve the purpose outlined above, comprised the following related activities performed for each of the components:

The major design decisions were specified in order to define the context for the next steps.

Technology comparison and selection was performed in order to select the best technology for designing the component. The technology selection process was based on predefined criteria which were divided into generic and specific ones. Based on that an analysis of the relevant technologies was made and the best fits were elicited. Finally, the missing elements and implementation needs were identified.

Each of the components was specified with respect to its static structure (sub-components design) and dynamic behavior (sub-components collaborations design). The necessary communication protocols, interfaces and formats between the components were specified.

The detailed descriptions provided by this technical specification will underpin the further developments of the project and will eventually end up with the final ADVENTURE software, supporting the forming and the execution of plug and play virtual factories.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>13</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

# 1 Introduction

ADVENTURE – ADaptive Virtual ENterprise manufacTURING Environment – is a project funded in the Seventh Framework Programme by the European Commission. ADVENTURE creates a framework that enhances the collaboration between suppliers, manufacturers and customers for industrial products and services.

## 1.1 ADVENTURE Project Aims

The framework proposed by ADVENTURE provides mechanisms and tools that facilitate the creation and operation of manufacturing processes in a modular way. ADVENTURE combines the power of individual factories to achieve complex manufacturing processes. It provides tools for partner-finding, process creation, process optimization, information exchange as well as real-time monitoring combined with the tracking of goods and linking them to cloud services.

There have already been several research projects that address the combination of different independent manufacturers to so-called virtual factories. Most of these research projects focus primarily on the business-side and on aspects like partner-finding and factory-building processes. However no proven tools or technologies exist in the market that provide for the creation of virtual factories applying end-to-end integrated Information and Communication Technology (ICT). ADVENTURE is aiming to provide such tools and processes that will help to facilitate information exchange between factories and move beyond the boundaries of the individual enterprises involved. The collaborative manufacturing process will be optimized by enabling the integration of factory selection, forecasting, monitoring, and collaboration during runtime.

ADVENTURE builds on concepts and methods of Service-oriented Computing and benefits from the advancements in this field. The monitoring and governance of the collaborative processes will be supported by technologies from the Internet of Things such as wireless sensors. Existing tools and services that can be integrated will be considered during the development of the platform for ADVENTURE.

The increased degree of flexibility provided through ADVENTURE will benefit SMEs especially as it helps them to react quickly to changes and to participate in larger, cross-organizational manufacturing processes. Furthermore, ADVENTURE will help manufacturers in assessing the environmental friendliness of actual manufacturing

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>14</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

processes and resulting products and services. Other objectives of ADVENTURE include research in areas such as service-based manufacturing processes, adaptive process management, process compliance, and end-to-end-integration of ICT solutions.

## 1.2 Deliverable Purpose, Scope and Context

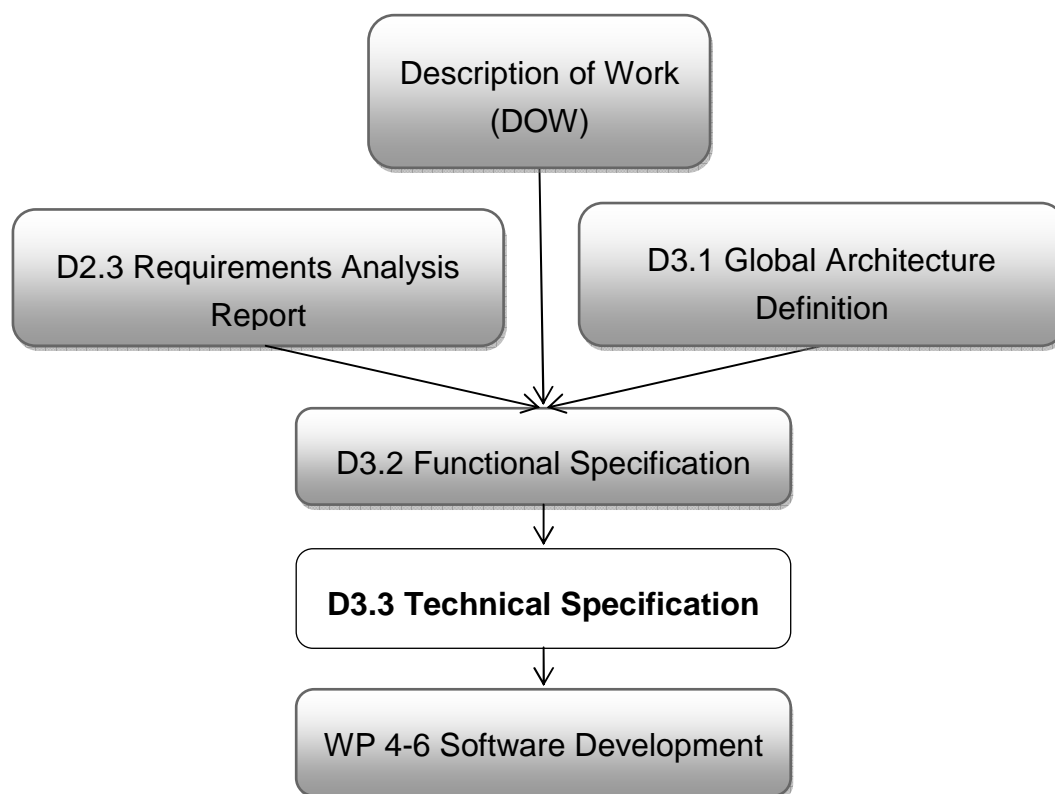
The goal of this deliverable is to present the technical specification of the ADVENTURE platform. It defines and describes the technical view of ADVENTURE architecture that will eventually end up with the final ADVENTURE software.

As shown in Figure 1, this deliverable is closely related with previous deliverables of the project, namely *D2.3 – Requirements Analysis Report* and *D3.1 – Global Architecture Definition* and directly related with *D3.2 – Functional Specification* and WP 4-6 (Software Development).

This deliverable specifies a design that supports the functionality defined in D3.2, consequently satisfying ADVENTURE user requirements as elicited in D2.3. The design is also constrained and put into technical context by D3.1. It describes how the ADVENTURE components are structured and operate technically, in order to provide the necessary functionality to its users.

The technical specification of the components will contain the necessary information for the technical design of the platform. Therefore, this deliverable serves as a basis for WP4, WP5, and WP6, where the software components that implement these functionalities will be developed.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>15</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



*Figure 1 - Context of the Deliverable*

### 1.3 Document Status

This document is listed in the DOW as ‘public’ since the technical specification may be useful for external parties as a reference platform design or as a base for understanding the technical elements of the project. It may also be used as a starting point when integrating external technical systems or software tools into ADVENTURE.

### 1.4 Target Audience

This deliverable is to be used by all project members, especially by the technical partners, to develop the intended prototypes at the early stage of the project and to integrate them into an end-to-end ADVENTURE solution. As a public deliverable it could also be useful for Future Internet Enterprise Systems (FINES) Cluster participants and other similar projects.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>16</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

## 1.5 Abbreviations and General Terms

A definition of general, common terms and roles related to the realization of ADVENTURE as well as a list of abbreviations is available in the supplementary document “Supplement: Abbreviations and General Terms” which is provided in addition to this deliverable.

Further information can be found at: <http://www.fp7-ADVENTURE.eu/glossary>.

Abbreviations, specific for this deliverable, are also listed in the *Annex A*.

## 1.6 Document Structure

This deliverable is broken down into the following sections:

**Section 1** provides an overall introduction to the ADVENTURE project and the general information related to the structure of this document. It also highlights the purpose, as well as relationships to other deliverables.

**Section 2** provides an overview of the system and presents functional analysis that includes the overall architecture design, functional recapitulation and translation from functional to technical specification. The section also contains a general description of the technical specification and explains the generic and the specific parameters used to select the right technology for each of the ADVENTURE components.

**Section 3** specifies the concrete technical details of ADVENTURE individual components. This includes the major design decision, technology comparison and selection, technical component specification and specification of interfaces, protocols and formats.

**Section 4** concludes the document and highlights the final outcomes from this deliverable.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>17</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

## 2 General Description of Technical Specification

This deliverable is closely aligned with previous deliverables of the project, particularly specifying the user requirements, system functionalities and the global architecture. As specified in deliverable *D3.1 Global Architecture Definition and shown in Figure 2* the functionalities of ADVENTURE will be realized by 13 components, organized in four architectural layers.

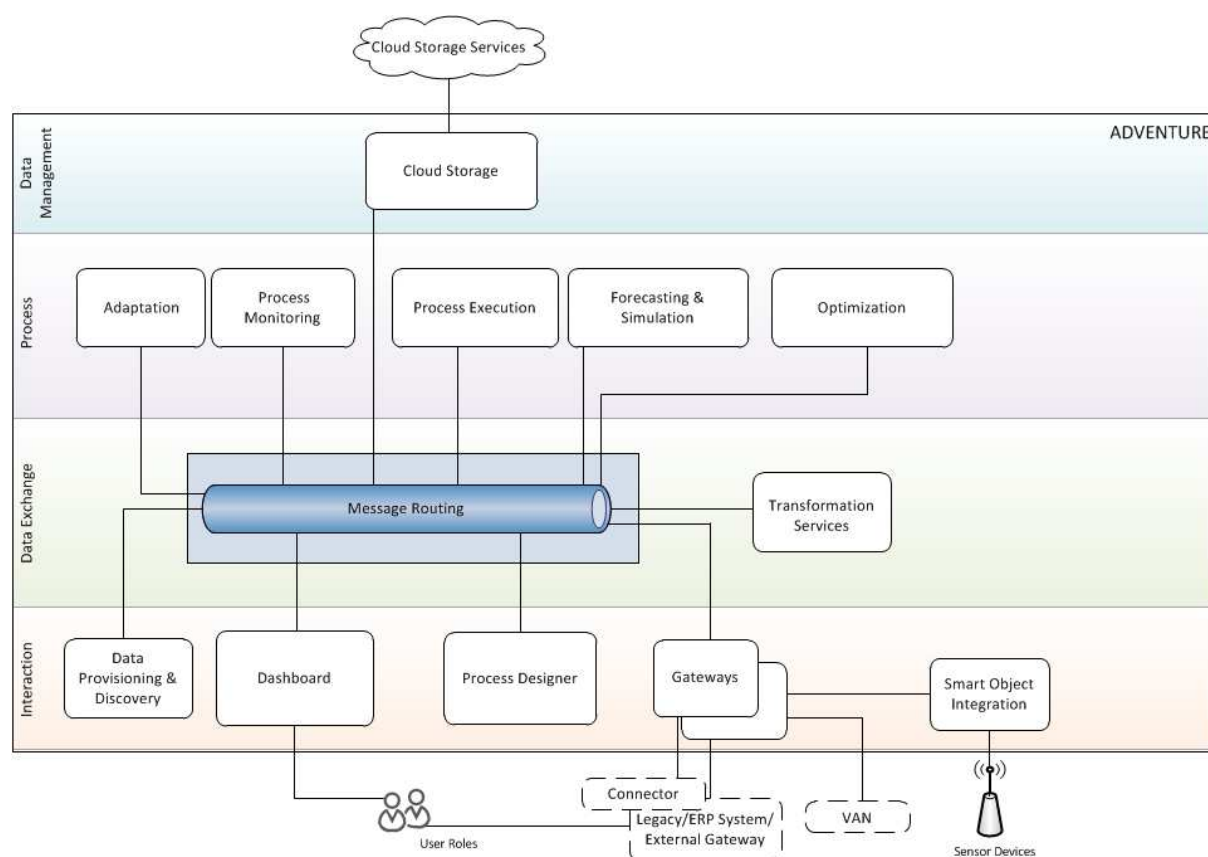


Figure 2 – Architecture of the ADVENTURE Platform

Based on this architecture the deliverable *D3.2 Functional Specification* presented a system level analysis of the ADVENTURE platform from a functional point of view. An overall functional characterization of each component explains the main ideas on what the component will do and how. Also, for each component, the corresponding use cases were identified and described. The interactions between the components were analyzed in detail and the services that each component will provide and consume were specified.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>18</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The Technical Specification moves the focus from the functional to the technical requirements to provide a technical understanding of the ADVENTUREs vision with regard to each individual component. It provides a pre-selection of technologies and compares their suitability with respect to users' requirements. The deliverable defines the related data structures, data models, choice of programming languages and tools, etc. The document also identifies the standard protocols and data exchange formats of each of the ADVENTURE components. These protocols and data exchange formats enable the communication among all the specific components and are considered as the backbone of interoperability between different technologies providers.

This technical specification addresses imperative pattern of data communication and provides an implementation scope on top of the defined standards. It accumulates technologies descriptions, specifies implementation options and highlights the interpretation of the standards. This approach helps developers to understand and define different interfaces and standards to ensure interoperability among various components of the ADVENTURE platform and also the wider/outside world.

In this deliverable, each of the thirteen components of the ADVENTURE platform is elaborated in a common form that addresses the following concerns:

**Abstract:** It is the summary of the functionality.

**Major Design Decisions:** It outlines common design patterns, practices, and concepts that relate to the software architecture.

**Technology Comparison and Selection:** This is divided into four parts. 1) Firstly, the selection criteria used to guide the technologies selection are defined. The best fit technologies are further to be selected based on these criteria. The table is defined in D3.2 Functional Specification and contains both generic criteria and specific parameters. The generic criteria are explained in D3.2, and the Specific Parameters are specific for each component and are presented in this document. The generic criteria that will be considered are the following:

- **Maturity & Stability:** Selecting a stable and mature solution is one of the key criteria for many components of ADVENTURE for ensuring a usable implementation of the ADVENTURE. However, for other components using cutting edge technologies may be more relevant
- **Regularly Updated:** The ICT market is developing significantly faster than any other market in the world and technologies which haven't been updated

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>19</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

within three years or more may be considered dead. As such, the consortium considers it important to choose as a base technologies that are updated regularly to reflect new market considerations

- **Technical Up-to-Datedness / Appeal:** Similar to the update policy, the technical up-to-datedness is considered as being important criteria. This will allow the consortium to use cutting-edge solutions and modern approaches for their development, hence creating more innovative solutions which will attract more attention during the project dissemination activities
- **Open Source:** Even if an existing technology may be used as a base for a component, it is very likely that the technology needs to be extended or changes in order to meet all project requirements. As such, Open Source solutions are likely to be preferred but in certain circumstances where this is not the focal RTD, or that this is not the market reality, off-the-shelf solutions can be of preference
- **Non-Infecting:** As specified in the CA, solutions that come with an infecting license such as GPL should be avoided and Apache 2 (like) licenses should be preferred
- **Code Quality:** A good code quality is required for any technology that is selected as a base for components
- **Extensibility:** As a potential alternative to a good quality Open Source solution, a technology with well-defined extendibility may be considered, e.g. in terms of plug-in mechanisms
- **Community:** For clarifying questions and discussing possible problems of a technology, a strong community should be available. This community could also be considered to be represented by the members of the developing organization for providing free or commercial support;
- **Performance:** Performance is an important criterion for all core components of ADVENTURE although the meaning of “performance” might differ from component to component;
- **Reuse of existing developments:** Reusing existing solutions of the partners should be preferred as those solutions that have been developed by the consortium partners will most likely be easier to adopt
- **EU project origin:** Bearing in mind the origin of ADVENTURE in the EU FP programs its leading-edge nature, solutions by other EU RTD projects can be quite useful in the context of a project although the juxtaposition is that most

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>20</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



likely these technologies may not be mature or market accepted. However, in general this criterion does not have a high priority compared to others;

- **Platform (Portability):** If possible, technologies should be preferred that allow a deployment into different platforms. For server based components such as the storage component, this will basically mean support of Linux and Windows systems. For components such as the Dashboard this will mean a support of all major browsers and for protocols this will mean the existence of libraries for different mobile, Web or desktop programming languages;
- **Open Standards Compliance:** Whenever an open standard exists, solutions using them should be preferred as long as those standards are openly accessible and applicable and as long as they are real world standards (defacto or dejure) that are actually used in the market and that do not hinder the technical up-to-datedness as described above;
- **Interoperability:** If possible, technical solutions that provide a good base for achieving interoperability should be preferred. For example, instead of a closed binary data format for querying or saving data in the cloud, an open and XML based solution should be preferred as it will potentially ease the interoperability with other components and thirds party systems. This criteria also reflects the integration capability with third party systems - e.g., ERPs

2) Secondly, the existing technologies for the component are listed with description and comparison. Because there are many technologies that exist, only the most relevant ones are listed based on pre-selection in the comparison process. However, many of the additional alternative technologies are tested and evaluated. 3) Thirdly, the most suitable technologies for ADVENTURE are selected that fulfill Generic Criteria and Specific Parameters. Critical reasoning is given for the selection. Different technologies/tools might be selected for each component. These technologies will be used as a base for the implementation. 4) Fourthly, the selected technologies are missing some elements that are needed in order to satisfy all the ADVENTURE requirements; therefore extra implementation might be necessary. Hence, supplementary technologies and possible changes are adapted to best fit ADVENTURE project.

**Technical Component Specification (component structure, internal sequence diagram):** This item depicts the overall static structure of the component as well as its internal interactions through a sequence diagram. The sequence diagram shows

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>21</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

how the sub-components operate and in what order. It is adapted in the context of the component breakdown as discussed in the component structure chapter (Chapter 3).

**Specification of Interfaces, Protocols and Formats (API specification, content formats and protocol definition):** In order to support the communication with other components in ADVENTURE platform, a list of API methods is provided. This section defines an API (which is formalized with Java language or (REST) service APIs which are formalized in terms of http requests/responses. It also defines the interaction protocols and the format of the exchanged data (scheme) over the protocols.

In this document the three components Adaptation, Simulation & Forecasting and Process Execution are integrated into a single one named Smart Process Execution (SPE). That is why the components as presented within D3.3 (11) differ in number compared to D3.1 (13). The sequence of these 11 components within the document is presented as follows:

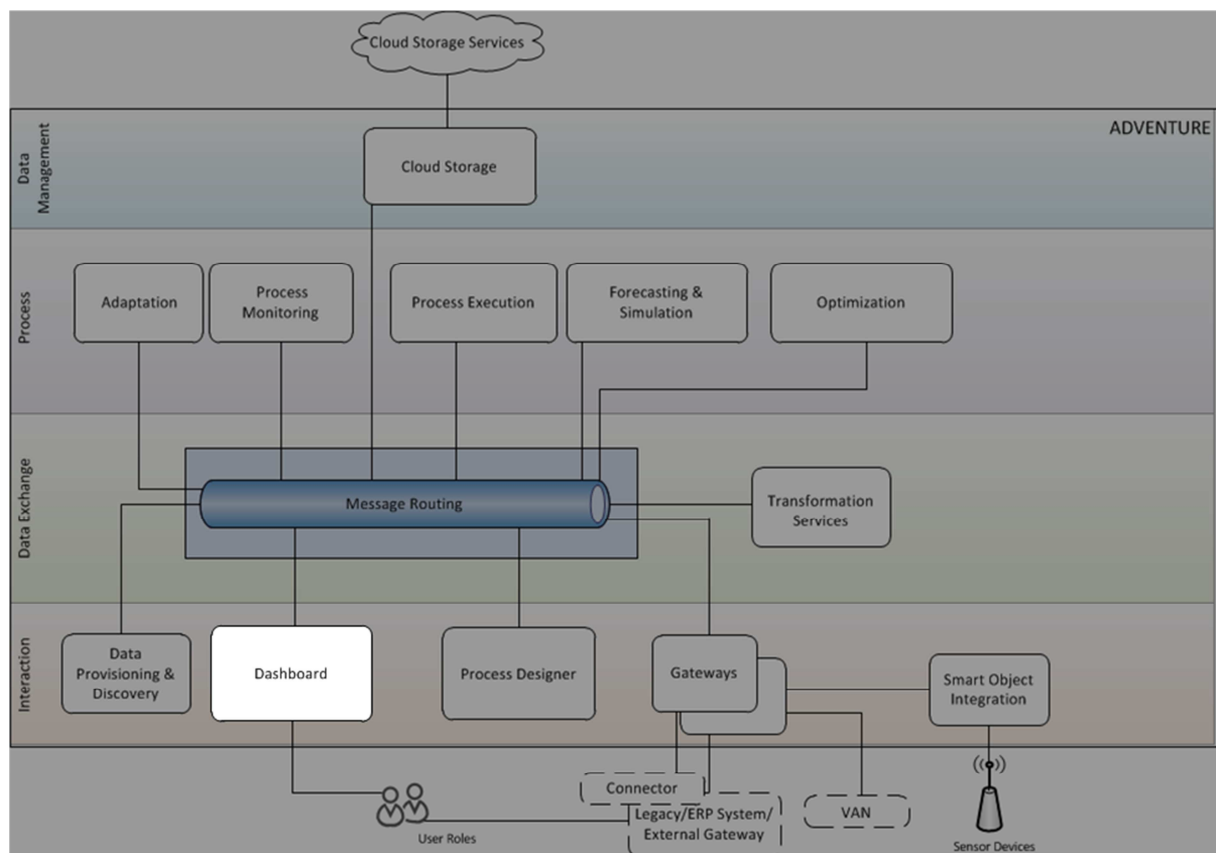
- Dashboard
- Cloud Storage
- Data Provisioning and Discovery
- Message Routing
- Transformation Service
- Gateways
- Process Designer
- Smart Process Execution
- Process Monitoring
- Process Optimization
- Smart Object Integration

This deliverable as a successor of the Functional Specification identifies the technical decisions for the future work within ADEVENTURE. The document will be used by the technology partners as a basis for further ADVENTURE developments and their integration into a complete solution.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>22</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### 3 Technical Specification

# Dashboard



## 3.1 Dashboard

Abstract:

Dashboard is designed to present a visual interface to ADVENTURE users.

Dashboard is supported by other components, such as Data Provisioning and Discovery, Process Designer and Process Monitoring.

All the functionalities are visualized by ADVENTURE Dashboard in four functional groups: Process Design, Process Management, Partner Management and Application Configuration.

### 3.1.1 Major Design Decisions

The ADVENTURE Dashboard serves as a graphical user interface (GUI) to all ADVENTURE functionalities. By using the Dashboard the user can create process templates and share the best practices with other business partners to improve the Virtual Factory Management process. The Dashboard provides convenience to users through a Web browser based interface, thus enabling users' access on any device that supports HTML and JavaScript. The Dashboard components are configurable for users with different roles allowing different views of the ADVENTURE components.

This Dashboard is designed as a basic UI environment supporting typical functionalities of a group based collaboration tool such as login or user registration.

The Dashboard will act as a “cockpit” for ADVENTURE users to see important system information in one single place, right after logging in. According to *D3.2 Functional Specification*, the ADVENTURE Dashboard is classified into four functional groups:

Virtual Factory Process Design

Virtual Factory Process Management

Virtual Factory Partner Management

Application Configuration and Setup

Each group will have 3 levels (From Level 0- Level 2). Level 0 is the Homepage and menu structure that enables users to navigate to all functionalities. It provides also customized overview monitoring information to users. Level 1 is the entry screen for each of the functional groups. Level 2 displays detailed information for each functionality.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>24</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The Dashboard features will cover the first 2 levels of the ADVENTURE UI, i.e. the start page that the user sees (level 0) and a more detailed overview page (level 1). As soon as more detailed information is needed (e.g. the visualization of a process), the UI of the corresponding component will be invoked.

After logging in, the user will see a welcome page with an overview of all their processes and their status information. Figure 3 presents an 'Overview of ADVENTURE Home Page' where the user can observe the overall high level status information of each of the processes that allows to monitor and control the collaborative activities, as needed to run a virtual enterprise successfully.

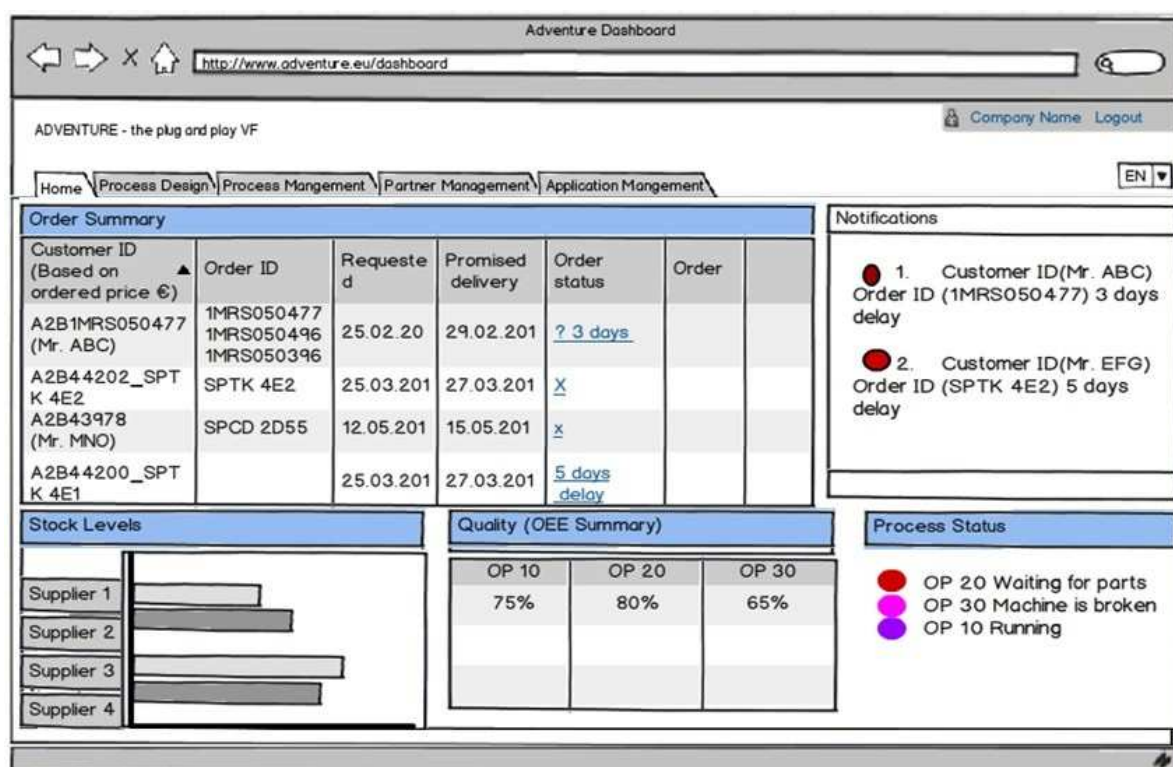


Figure 3 - Overview of the Dashboard Homepage

Clicking on the tabs on menu bar will lead to Level 1, which is a more detailed view of current processes of a user – see Figure 3.

Each group includes different functionalities to realize Virtual Factory Management:

The Virtual Factory Process Design contains necessary supportive information related to design a virtual factory. It also provides template suggestions to brokers, in order to improve the process design action. In this group, Level 1 includes

functionalities as Process Model Management (PMM), Process Model Editor (PME), Process Simulation and Process Optimization.

Figure 4 shows the Process Design Level 1 example, which offers a core component which is the graphical process designer, but it also includes tools allowing partner search and assignment, as well as process simulation and optimization.

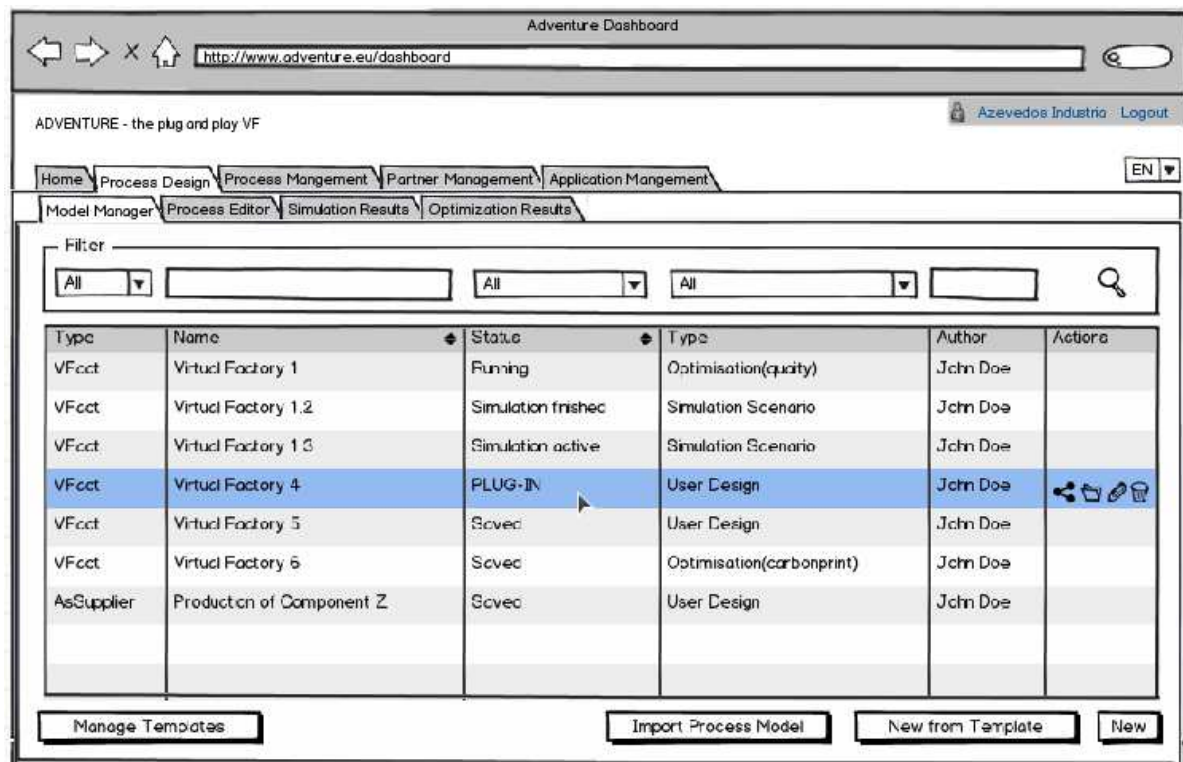


Figure 4 - Process Design

The Virtual Factory Process Management deals with the activity that supports the execution, monitoring and assessment of the business processes. This functionality group Level 1 includes Real-time Monitoring, Process Log, Process Analytics and Process Adaptation. When there is any potential risk during the Process Execution, an alert message or notification presents the detailed information to brokers. This alert message includes forecasting of delay or content related information such as “low stock level”. If any process fails, process adaptation should be readily available to overcome it. Figure 5 shows the main functionalities that a user is able to access within the Process Management functionality group.

Instance Status	Instance Type	IID	Process Status	Involved Partners	Operations	Monitoring
running (waiting)	Main	RO	Waiting for Orders	Azevedos, Control2K, ABB,	<a href="#">Stop</a>	<a href="#">Details</a>
running (waiting)	Main	ME	Periodically Querying Environmental Sensors	Azevedos, Control2K, ABB	<a href="#">Stop</a>	<a href="#">Details</a>
running	Subprocess	HM1-3	Details for Portugese Machine Building	Azevedos	<a href="#">Stop</a>	<a href="#">Details</a>
running	Subprocess	HM1-2	Details for Portugese Machine Building	Azevedos	<a href="#">Stop</a>	<a href="#">Details</a>
stopping	Subprocess	HM1-4	Fire in Portugese Factory 1	Azevedos		<a href="#">Details</a>
stopping	Subprocess	HM2-2	Water in eTMCO Facility 2	eTMCO	<a href="#">Edit</a>	<a href="#">Details</a>
stopped	Subprocess	HM2-1	Manual Cleanup in eTMCO Facility 1	eTMCO	<a href="#">Edit</a> <a href="#">Start</a>	<a href="#">Details</a>
finished	Subprocess	HM3-1	Space Station finished	ABB		<a href="#">Details</a>

Figure 5 - Process Management

The Virtual Factory Partner Management provides basic functionality to maintain the profiles of the Virtual Factory Partners. This profile is populated with both the description of the attributes of the partners related to a specific business service such as capacity and capabilities, production lead-time, price level, KPIs, performance level, etc., and the level of process interoperability that allows them to join in a Virtual Factory successfully. Level 1 functionality includes the useful information according to the user needs. Figure 6 shows the main functionalities of the Partner Management along with its associated sub-functionalities such as Profile Editor, Partner Finding, and Partner Analytics.

Adventure Dashboard

http://www.adventure.eu/dashboard

ADVENTURE - the plug and play VF

Azevedos Industria Logout

Home Process Design Process Mangement Partner Management Application Mangement

EN

Profile Editor Partner Finding Partner Analytics

Legal Contacts Description Services Processes

Name: Feelgood Inc.

Local identity code: 1234567890

VAT code: BG1234567890

Legal form: Limited Liability Company

HQ Location: Sofia 1000, Bulgaria

Legal address: Sofia 1000, Bulgaria

Telephone: +350 2 123 456

Fax: +350 2 123 789

Web site: www.feelgood.com

email: feelgood@cloud.com

Save Cancel Delete

Figure 6 - Partner Profile Tab

The Application Configuration and setup group (Figure 7) provides the ability to customize the application, which includes Configurable User Interface, Gateways Configuration, Alarms Configuration, Message Transformation and Routing Configuration. This allows brokers to customize the process visualization according to their own needs.



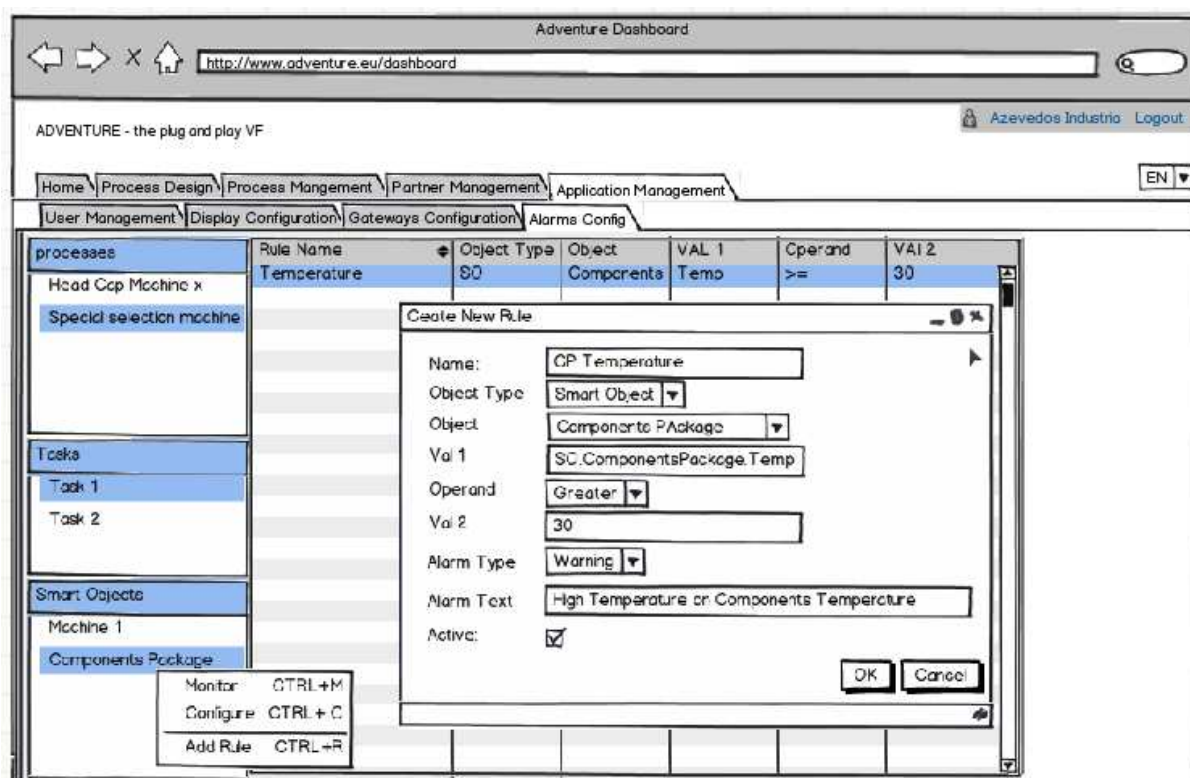


Figure 7 - Application Setup Tab

As an UI component, the Dashboard should visualize the four functional groups namely, Process Design, Process Management, Partner Management and Application Configuration. The different groups are further supported by different ADVENTURE components (i.e. Process Designer, Data Provisioning and Discovery, Process Forecasting and Simulation, Process Monitoring, Process Optimization, etc.). Hence, other components will be interfaced with the Dashboard component, in order to visualize the relevant valuable information on it.

### 3.1.2 Technology Comparison and Selection

#### 3.1.2.1 Selection Criteria

The selection criteria for Dashboard technologies were defined in the *D3.2 Functional Specification* in section 5.1.5 on page 48. The description of the generic parameters can be found in the *D3.2 Functional Specification* in section 5 on page 28. Now follows the description of the Dashboard specific parameters:

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>29</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**User friendly:** The UI should be easy to use and the information should be presented quickly and accurately.

**Visualization quality:** Because Dashboard conveys all the information to users, up-to-date and relevant visualization is a key driver to successful delivery value.

**Browser Support:** There are several widely used browsers, so the UI should support the most popularly used browsers.

**Community Management:** The Dashboard should provide an interface to manage the virtual factory partners' community.

**Interface Protocol:** The protocol is for communicating UI events and updates on the Web

**Authentication:** The Authentication Service UI provides a mean for gathering the credentials needed by the Authentication Service to validate a user.

Based on the above specific parameters, existing platforms will be compared for the Dashboard component in the next sub-section. These platforms will be used as a basis for the development.

When considering Dashboard design, it is important to make sure that this UI supports dynamic variation of input fields, flexible changes of access rights and personal preferences, as well as consistent data integrity. In general there are two strategic choices of implementing Dashboard: (i) to develop from scratch (manual creation), and (ii) to develop based on an existing Enterprise Portal. Usually, developers make this decision based on their specific requirements. For the concrete case it is necessary to analyze the implementation requirements of ADVENTURE.

If considering developing UI from scratch, the advantage is that it does not limit or burden the developers. In this approach, it is very important to consider usability and consistency. Developers should define the overall theme of the application and make sure the interfaces of Dashboard between one level and another are consistent.

When considering developing Dashboard from scratch, the UI should include three components: content presentation, user navigation and data manipulation.

**Content presentation:** Developers need to consider the presentation layer of the Dashboard, for instance appropriate chart types, table format, and the content presentation. The display's parameters such as captions, type of chart or graph, minimum/maximum values, data source, color codes and others should be defined properly.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>30</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**User navigation:** The way the user navigates to other pages and how the pages are linked to each other should be considered. The user navigation affects the usability of Dashboard.

**Data manipulation:** The Dashboard is used to update the Dashboard's internal database. It is important to ensure the content of Dashboard is up to date, hence the queries must run regularly to deliver information and refresh the data on Dashboard.

However, the problem of developing UI from scratch is time consuming. The purpose of Dashboard is to visualize all the functionalities, so it is good to spend the core of implementation time on other design aspects. In ADVENTURE, the effort for developing the UI can be minimized by exploiting a pre-built UI kit. The second strategy is to develop a UI based on an existing Enterprise Portal. The Enterprise Portal through providing a design methodology, allows developers to quickly implement UI, without the need of recreating all the components. This design standard can insure the consistency in developing process. Developers can reuse architectural frameworks, component libraries, and methods developed by the Enterprise Portal Vendors. Of course, the limitation of choosing this method is the loss of governance. In this sense, the selection of an Enterprise Portal is particularly important. In ADVENTURE project, the approach of designing based on an Enterprise Portal is preferred. One of the greatest benefits of Enterprise Portal is that it provides an easily accessible interface to users and a stable and extensible development environment to developers. Although developing a UI from scratch may not be difficult, it consumes lots of time and resources. An Enterprise Portal is designed to aggregate information through different Portlets. In this way, it can effectively avoid developing all components from scratch and ADVENTURE development team can concentrate more on the core components development. Portlets are pluggable UI components that are managed and displayed in a web portal. Portlets will produce fragments of markup code, and these fragments are aggregated into the portal. Practically, several non-overlapping portlets are displayed on one portal page.

### 3.1.2.2 Possible Technologies

An Enterprise Portal is a framework for integrating information, people and processes across organizational boundaries. Its solution promises significant decrease of time

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>31</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

and costs of implementation. There are many vendors that provide Enterprise Portals and according to Gartner's Magic Quadrant for Horizontal Portal Products in 2011<sup>1</sup>, Microsoft, IBM, Oracle, SAP, Liferay are named leaders in this marketplace. They have a full range of capabilities to support a variety of portal deployment scenarios and have demonstrated consistent product delivery in meeting customer needs for a substantial period of time. RedHat (JBoss) is a challenger with a significant ability to execute, but lacks the degree of portal-specific vision, demonstrated by market leaders. In this section, these Enterprise Portal platforms are compared and presented in Table 1:

*Table 1 - Main Enterprise Portal Platforms Summary*

Vendor	Product Name	Technology	Portlet API	Introduction
Liferay	Liferay Portal	Java EE	JSR-168, JSR-286, WSRP	<a href="http://www.liferay.com/">http://www.liferay.com/</a>
RedHat	JBoss Enterprise Portal Platform	Java EE	JSR-286, WSRP	<a href="http://www.redhat.com/products/jbossenterprisemiddlewa re/">http://www.redhat.com/products/jbossenterprisemiddlewa re/</a>
Microsoft	Office Sharepoint Server 2010	ASP.NET	WSRP	<a href="http://sharepoint.microsoft.com">http://sharepoint.microsoft.com</a>
Oracle	Oracle WebCenter Suite	Java EE	JSR-168, JSR-286, WSRP	<a href="http://www.oracle.com/technetwork/middleware/Webcenter/suite/overview/index.html">http://www.oracle.com/technetwork/middleware/Webcenter/suite/overview/index.html</a>
IBM	WebSphere Portal	Java EE	JSR-286, WSRP	<a href="http://www-01.ibm.com/software/WebSphere/portal/">http://www-01.ibm.com/software/WebSphere/portal/</a>
SAP	SAP NetWeaver	Java EE/ ABAP	JSR-168	<a href="http://help.sap.com/nw70/">http://help.sap.com/nw70/</a>

## Liferay Portal

Liferay is an enterprise Web platform for building business solutions. It is primarily used to power corporate intranets and extranets. Liferay offers integrated Web publishing and content management, an Enterprise Service Bus and Service-

<sup>1</sup> <http://www.gartner.com/technology/reprints.do?id=1-17RTIFE&ct=111025&st=sg>

Oriented Architecture, and compatibility with all major IT infrastructures. It supports for plugins extensibility into multiple programming languages. It can support various application servers and is ready for third-party Java assets to gain scalability and enterprise compatibility.

### **RedhatJBoss Enterprise Portal Platform**

The JBoss Enterprise Application Platform is an Open-Source platform used for building, deploying, and hosting highly-transactional Java applications and services. The JBoss Enterprise Application Platform is part of the JBoss Enterprise Middleware portfolio of software. Because it is Java-based and it operates cross-platform, it is usable on any operating system that supports Java.

### **Microsoft Office SharePoint**

Microsoft SharePoint is a Web application platform. It builds based on .NET platform. SharePoint provides portal to leverage Web tools and functions without having to understand the underlying technical platform. It facilitates intranet portals, document and file management, collaboration, social networks, extranets, Web sites, enterprise search, and business intelligence. It also has capabilities around system integration, process integration, and workflow automation. However, it often requires support from third-party products for filling its gaps.

### **Oracle WebCenter Suite**

Oracle WebCenter is Oracles Application Development Framework. It contains a set of components for building Web applications and team collaboration/social sites. Oracle WebCenter is targeted at both the development community and business users, delivering a development environment that includes WebCenter Framework and WebCenter Services along with an out-of-the-box application for team collaboration and enterprise social networking.

### **IBM WebSphere Portal**

IBM WebSphere Portal is designed as a framework and a collection of services, often implemented as Portlets, widgets or other components. It provides a single access point to Web content and applications. The basic package includes a Web server, database, development tools, Web site templates and other tools. Although IBM continues to simplify and improve overall usability, WebSphere still need sophisticated IT departments and advanced skills to configure properly, and the user

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>33</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

experience depends largely on IT and design skills, rather than out-of-the-box characteristics.

### SAP NetWeaver

SAP NetWeaver is SAP's integrated technology computing platform and is the technical foundation for many SAP applications. SAP NetWeaver is marketed as a service-oriented application and integration platform. SAP NetWeaver provides the development and runtime environment for SAP applications and can be used for custom development and integration with other applications and systems.

The following table shows all potential Enterprise Portals as well as their rating:

*Table 2 - Comparison of technologies for Dashboard*

Parameter	Importance (- - - +/- +++)	Liferay Portal	Redhat (JBoss)	Share point	WebCenter Suite	WebSphere Portal	NetWeaver
<b>Generic Parameters</b>							
Maturity & Stability	+++	+++	+	+++	++	+++	++
Regularly Updated	+	++	-	-/+	+	+	-/+
Technical Up-to-Dateness / Appeal	++	++	-/+	+	++	++	+
Open Source	+/-	YES	YES	NO	NO	NO	NO
Non-Infecting	++	NO	NO	N/A	N/A	N/A	N/A
Code-Quality	++	+++	++	N/A	N/A	N/A	N/A
Extensibility	++	++	+	+	++	+	++
Community	++	YES	YES	YES	YES	YES	YES
Performance	+++	+++	++	+++	++	+	+
Reuse of existing developments	++	+++	N/A	N/A	N/A	N/A	N/A

EU project origin	+/-	YES	NO	NO	NO	NO	NO
Platform (Portability)	+++	+++	+++	+	++	+	++
Open Standards Compliance	++	+++	++	+	++	++	+
Interoperability	+++	+++	+	+	+	+	+
<b>Specific Parameters</b>							
User friendly	+++	+++	+	+++	++	+	++
Visualization quality	+++	+++	+	+	++	+	++
Browser Support	+++	+++	++	+++	++	++	+
Community Management	+++	+++	-	+	++	+	++
Interface Protocol	+++	+++	+	+	++	+	++
Authentication	+++	-/+	++	++	+	++	++

### 3.1.2.2.1 Conclusion

This section includes a comparison of various properties of different Enterprise Portals. Six alternative solutions are introduced and evaluated based on the criteria (Generic Parameters and Specific Parameters). This technology comparisons table tends to select Liferay as a more valuable and best fit Enterprise Portal for ADVENTURE. In the following sub section, more reasons are presented to support the decision.

### 3.1.2.3 Technology Selection

When selecting the Enterprise Portal, there are many key factors to be considered. For instance, this technology can be open or close source, dynamic, easy to use, secure, etc. Besides, this solution needs extension and customization.

Based on the comparison of different Enterprise Portals, Liferay Portal is selected for ADVENTURE Platform and it enables the communication between different Portlets.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>35</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Liferay Portal allows easier control over the tasks that users could perform. It provides easy access to the user and reduces the burden on developers. Liferay does not impose specific requirements on the use of any development frameworks, so that portal developers can choose the best tools for their projects. Hence, it provides more flexibility to developers. The Liferay advantages are summarized as follows:

Supports all existing API specifications.

Supports configurable Dashboard based on AJAX technology.

Supports single-sign-in to reduce time consume and improve the security. In ADVENTURE, a single-sign-in solution is one of the requirements from user perspective.

Supports the community setup and management. In this way, the communications between different ADVENTURE users can be setup easily.

Supports Web Services, JMS, SOAP, XML, etc.

Provides login analysis. This helps to implement the user login and session management.

Supports different databases.

#### 3.1.2.4 Missing Elements and Implementation Needs

Liferay provides Web-based UI development architecture. Liferay support aggregate content and applications, access control, and also a consistent user experience. Users can start Dashboard with a single centralized place. However, the major shortcoming of most Enterprise Portals is lower rate of adoption. For instance, the lack of standard support for different browsers, the security risks of rich UI approaches. The main issue is to customize the overall look and feel of Liferay.

From this aspect, supplementary technology is needed. Liferay portal includes Vaadin as a pre-packaged framework for developing attractive, easy-to-use applications. The collaboration between Liferay and Vaadin will simplify and reduce the effort of developing rich and secure UIs for ADVENTURE platform functionalities. Developers can take advantage of using Vaadin library when creating new Liferay Portlets.

Vaadin (<https://vaadin.com/home>) is an Open Source Web application framework for rich Internet applications. While providing generic support for all portals implementing the standards, Liferay portal could supports Vaadin library and the needed portal-

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>36</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



specific configuration is given based on Liferay. Vaadin features a server-side architecture, which means that the majority of the logic runs on the servers. AJAX technology is used at the browser-side to ensure a rich and interactive user experience. On client-side Vaadin is built on top of and can be extended with Google Web Toolkit (GWT).

Google Web Toolkit (<https://developers.google.com/Web-toolkit/>) is an Open Source set of tools. It allows developers to create and maintain complex JavaScript front-end applications in Java. The advantage is that everything is Java source, and it can be built on any supported platform with the included GWT Ant <sup>2</sup>build files.

### 3.1.3 Technical Component Specification

#### 3.1.3.1 Component Structure

The Dashboard Component Structure (Figure 8) gives a basic illustration of the Dashboard architecture. The core of the ADVENTURE Dashboard is the application server.

---

<sup>2</sup> <http://code.google.com/p/ant-gwt/>

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>37</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

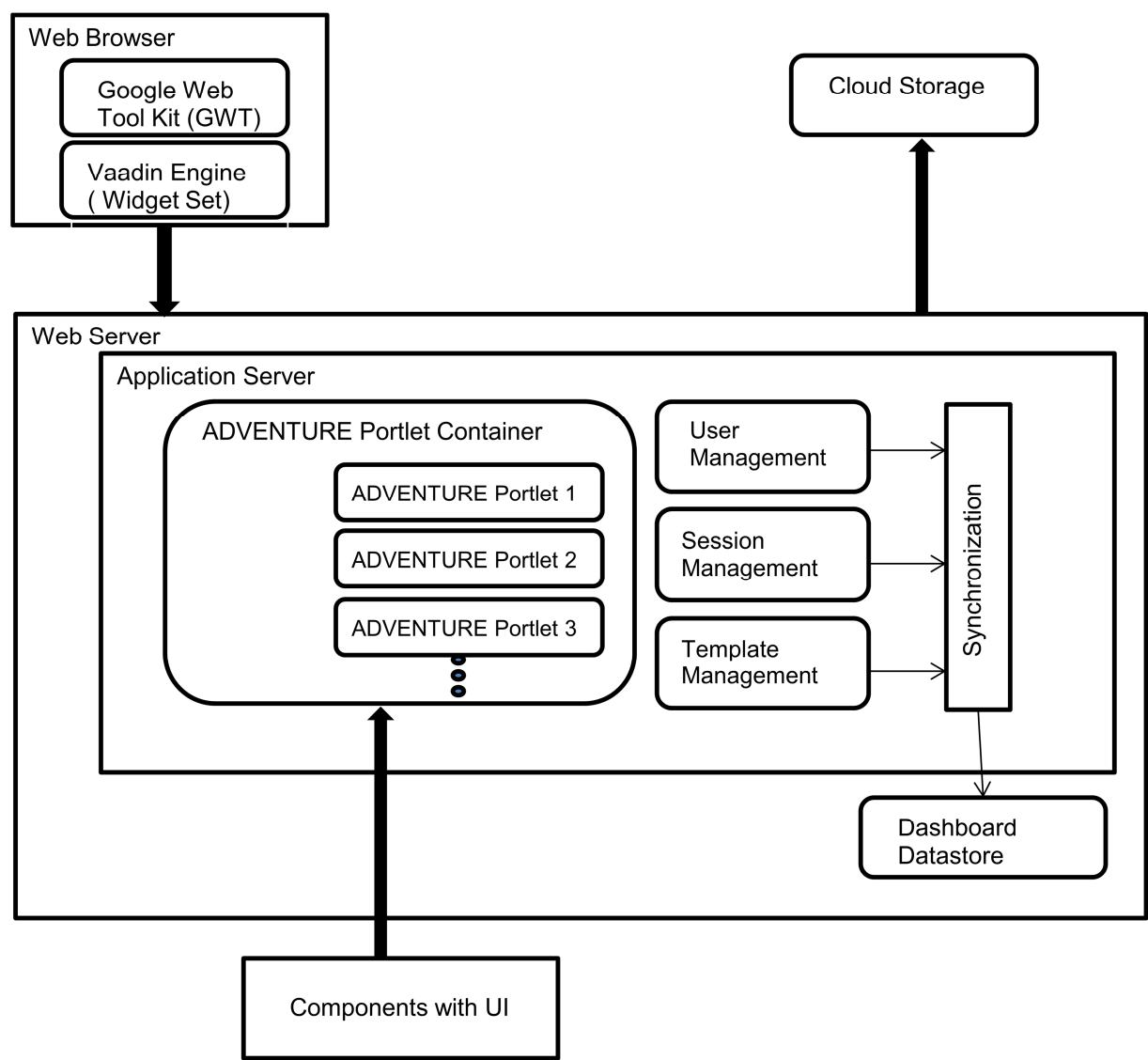


Figure 8 - Dashboard Component Structure

**Web Browser:**

The Client-Side Engine of Vaadin manages the rendering in the Web browser using Google Web Toolkit (GWT). It communicates user interaction and UI changes with the server-side using the User Interface Definition Language (UIDL), a JSON-based language. The communications are made using asynchronous HTTP or HTTPS requests.

**ADVENTURE Portlet:**

The ADVENTURE Portlets are reusable and pluggable UI modules that are managed and displayed in the ADVENTURE Dashboard. Portlets provide access to other ADVENTURE Components. A portal page is displayed as a collection of non-overlapping Portlet windows, where each window displays a Portlet. Hence a Portlet (or collection of Portlets) resembles a Web-based application that is hosted in a portal. Some examples of Portlet applications in ADVENTURE are Order Summary, Stock Levels, Notification, Process Status, etc. The Portlet can be collapsed, removed and edited. The Portlet can be placed in different position by using drag and drop technique.

**ADVENTURE Portlet Container:**

The ADVENTURE Portlet Container is responsible for aggregating a set of Portlets that are to appear on any particular page. That means, a page can contain multiple Portlets and when the user interacts with one Portlet, it may need the other Portlets to react to the change immediately. The ADVENTURE Portlet Container executes Portlets and manages their life cycle. It is the runtime environment for Portlets using the JSR 168/286 or WSRP specifications, in which Portlets are instantiated, used, and finally destroyed. ADVENTURE Portlet Container is responsible for aggregating the set of Portlets that are to appear on any particular page.

**Component with UI:**

Several components in the ADVENTURE need a UI. For instance, Process Designer, Process Monitoring, Process Optimization, Process Forecasting and Simulation, Process Execution and Process Adaption. For each such component the UI can be designed and decomposed into several Portlets. Although these Portlets are responsible for different functionalities, they cooperate with each other. Therefore, a portal page contains a collection of non-overlapping Portlets.

**User Management:**

User accounts within the ADVENTURE platform will be managed. This Dashboard provides a UI to create, edit and delete users. Roles with different privileges are created to manage the users.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>39</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Session Management:**

User sessions within the ADVENTURE platform will be managed. User login information, user actions and running Portlets will be tracked. Once the client requests are sent to the Web server, they are interpreted into user events for a particular session. Sessions are tracked using cookies.

**Template Management:**

The configurable ADVENTURE Dashboard will have a flexible UI to support the needs of different users, which means a portal page may display different Portlets to different users. Therefore, Template Management will allow users to create a role-or user-based view on the ADVENTURE Platform.

**Synchronization:**

The User Management, Session Management and Template Management need to be synchronized and managed together. The data will be directly loaded from and saved into Dashboard Datastore, which could be integrated with Cloud Storage.

**Dashboard Datastore:**

Liferay provides a default database called "hypersonic." In ADVENTURE platform, it needs to switch to a real database to use Liferay. The Dashboard Datastore will be used to store all the Liferay related data, for instance user information, login information and application setting information, etc.

**3.1.3.2 Internal Sequence Diagrams**

In the ADVENTURE case, there are four main portal pages (Process Design, Process Management, Partner Management and Application Setup), each page displays a collection of Portlet windows produced by other components (Process Designer, Process Monitoring, Process Optimization, etc.). As mentioned previously, the Level-0 and Level-1 will be implemented by Dashboard. The sequence diagrams of Level 0 and how to create Level 1 are introduced below.

**3.1.3.2.1 Login ADVENTURE Platform**

The user should login the ADVENTURE Platform in order to access all the functionalities of Dashboard. After user login, all the actions and events within Dashboard are tracked and recorded by Session Management. The system needs to

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>40</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

be logged out to ensure the session is ended and the data transmission ends securely (Figure 9).

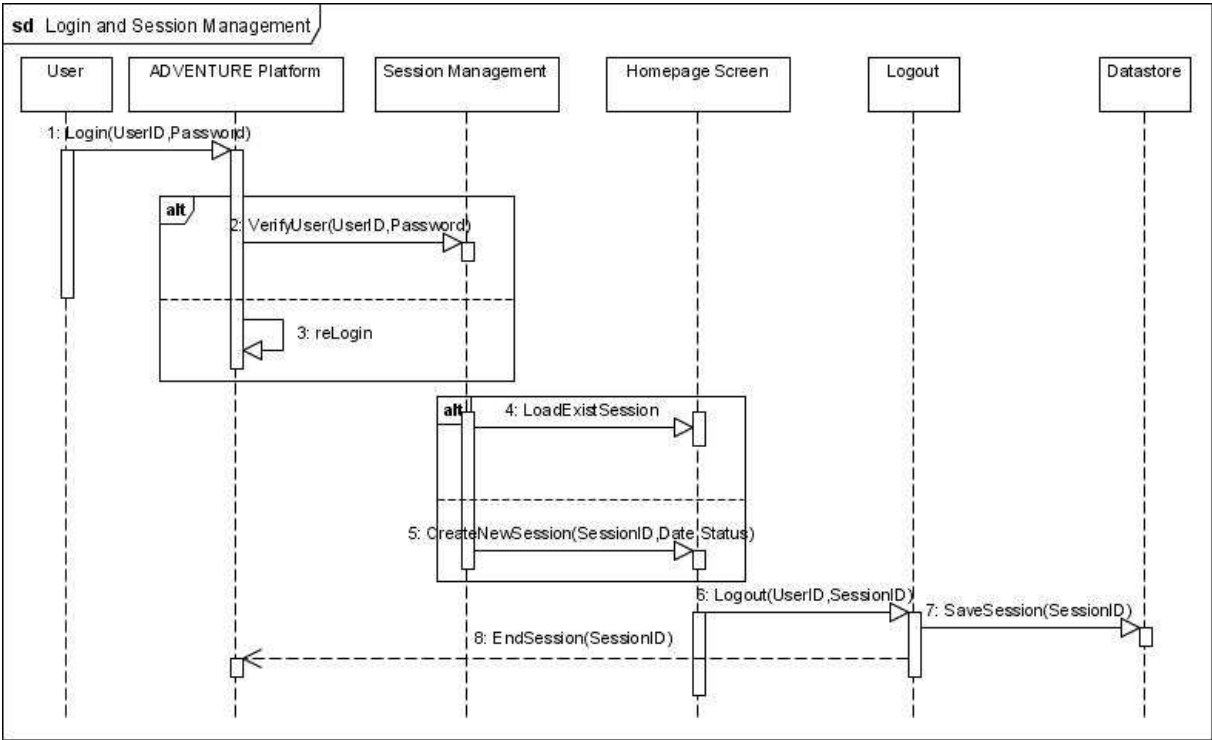


Figure 9 - Login and Session Management Sequence Diagram

Users can login to the system with a correct username and password and then the Dashboard Homepage screen should be displayed. The session management checks whether a previous open session exists for the given user. If not, then it creates a new session. Relevant session information will be created by session management, i.e. session identifier (session ID) and other session data (user name, account number, etc.) and tracking information (which applications /components are open and using, which documents each application has opened). The User sign out of the platform and the session ends. This can create a session history for future use.

3.1.3.2.2    Create Portlet

Other components in the ADVENTURE need consistent UI as well. For instance, Process Designer, Process Monitoring, Process Optimization, Process Forecasting and Simulation, Process Execution and Process Adaption. Developers send request to create a new page. This page can include several Portlets. Once container is

approved, developers can create new Portlet based on the standards. All the information related to each Portlet should store in the Dashboard Datastore. The Portlet can be further re-used (Figure 10).

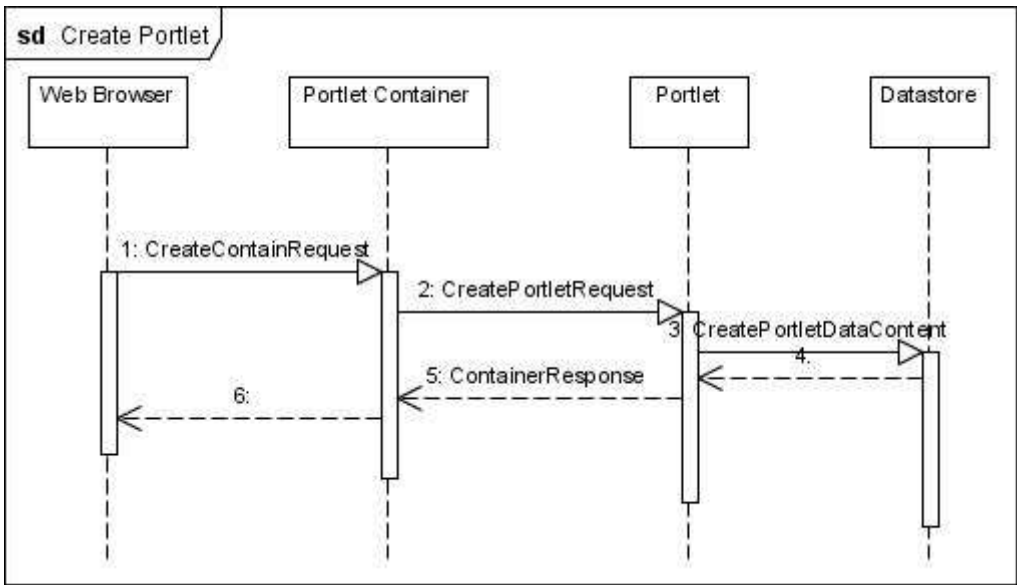


Figure 10 - Create Components as Portlet

3.1.3.2.3 Add Portlet

Portlets are the heart of portal, because they contain the actual functionalities. These Portlets are deployed as plugins into a Liferay instance. A single plug-in can contain multiple Portlets.

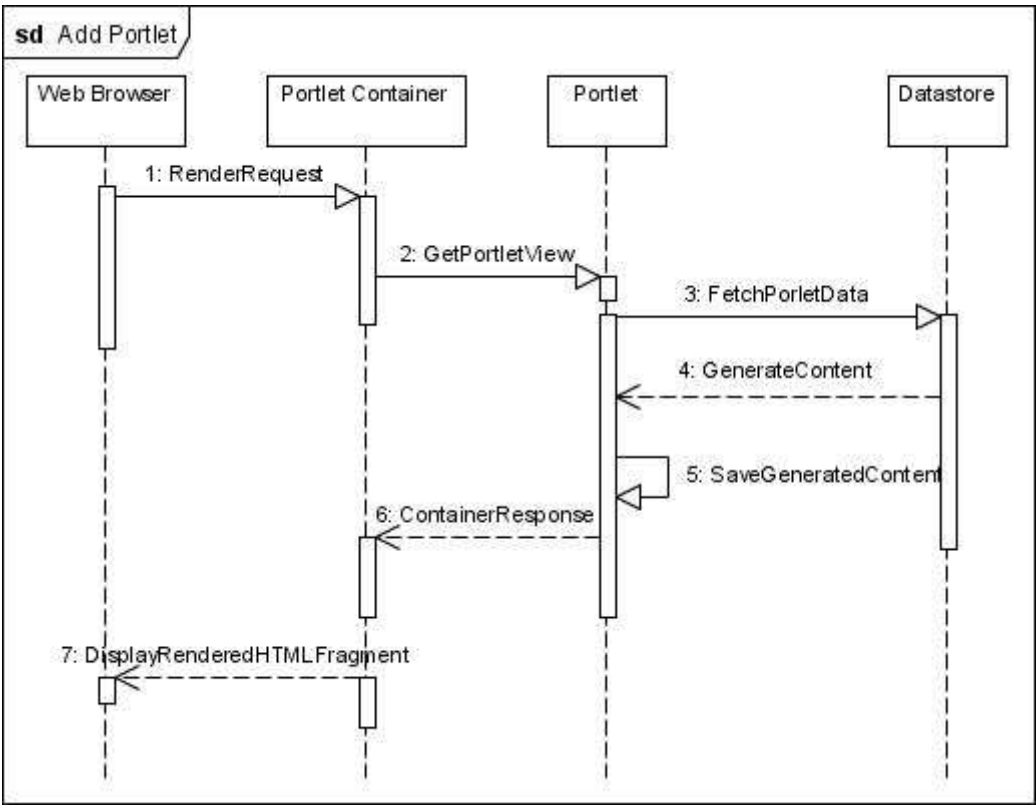


Figure 11 - Add Portlet in Portal Page

After creating a component as a Portlet, in order to use this Portlet, it should be added to the portal. All the Portlets should be displayed in Liferay platform. For instance, when creating the Process Monitoring component as a Portlet, it should be added to a portal page, such as Process Management Page. In order to add ADVENTURE components as Portlets, a user has to sign in as privileged user (Administrator). After creating a new Portlet, developers can navigate to the right Portlet container, and render request of adding Portlet. Once Portlet is deployed in the background, the Portlet can be displayed on the Web page as HTML files (Figure 11).

3.1.4 Specification of Interfaces, Protocols and Formats

Liferay is configured by a combination of settings which are stored in the database (configured by the use of the Control Panel) and settings which are stored in properties (text) files. These files can be modified to change Liferay's behavior in certain ways. For instance, specify where to get the overridden properties, how to

release the portal, etc. In order to change the value of its properties, all the modifications are done in the configuration file: portal-ext.properties. It is stored in the Liferay Home directory.

#### 3.1.4.1 API specification

Portlets are Web-based components managed by Portlet containers that supply dynamic content. Portals employ Portlets as pluggable UI components, a presentation layer for the system. Because there are many different Enterprise Portals, various vendors have created different APIs for Portlets. This variety of incompatible interfaces generates problems for application providers, portal customers, and portal server vendors. To enable interoperability between Portlets and Portals, this specification will define a set of APIs for Portal computing addressing the requirements of aggregation, personalization, presentation, and security for Portlets running.

JSR 168 is the Version 1.0 of the Java Portlet Specification. The purpose of the specification is to solve the problem of Portlet compatibility between portal servers offered by different vendors. JSR 286 is Version 2.0 of the Java Portlet Specification that extended the capabilities to include coordination between Portlets, resource serving, and other advanced features. The Web Services for Remote Portlets (WSRP) specification defines a Web service interface for accessing and interacting with interactive presentation-oriented Web services.

These specifications are not competing technologies. JSR 168 may be used to define a Portlet, and WSRP may be used to define a Portlet's operations to remote containers. JSR 168 Portlets and WSRP may be used together to define a Portlet and to provide remote operations.

#### Database configuration

Liferay comes with a default database called HSQL or "hypersonic." When adapt to ADVENTURE cloud based storage, the default database system should be changed accordingly. Liferay supports different external databases, including: IBM DB2, MySQL, Oracle, PostgreSQL, SQL Server and Sybase which can also be used.

#### Cloud configuration

Liferay Portal is deployable to the cloud and virtualized environments, including EC2, Elastic Beanstalk, and VMWare.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>44</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



### 3.1.4.2 Content Formats and Protocol Definitions

Deploying a Portlet application based on Liferay is the same as deploying a regular application. Liferay provides a demo package WAR (available: <http://www.liferay.com/downloads/liferay-portal/additional-files>) Liferay provides a standard to enable developers to create portlets that can be plugged into Liferay.

The Java Portlet Specification allows a Portlet to be packed as a .war file for deployment to a J2EE application server. Just like a .war file used to deploy a typical J2EE Web application, it contains a WEB-INF/Web.xml file to configure the application context. However, with a Portlet application, the WEB-INF folder must also contain a `portlet.xml` file. The `portlet.xml` file is a descriptor file, containing configuration details about all bundled Portlets in the .war file.

In this Liferay based ADVENTURE, the file `liferay-portlet.xml` is used to define Liferay Portal specific features. This file has been placed in the `WEB-INF` directory of any Portlet application. Following is an example of what this file may look like:

The following listing shows a simple example of a `portlet.xml` file.

```
<?xml version="1.0"?>
<liferay-portlet-app>
  <portlet>
    <portlet-name>Sample</portlet-name>
    <icon>/html/icons/search.png</icon>
    <struts-path>search</struts-path>
    <configuration-action-
class>com.liferay.portal.kernel.portlet.DefaultConfigurationAction</configuration-action-class>
    <preferences-owned-by-group>true</preferences-owned-by-group>
    <use-default-template>false</use-default-template>
    <restore-current-view>false</restore-current-view>
    <private-request-attributes>false</private-request-attributes>
    <private-session-attributes>false</private-session-attributes>
    <header-portlet-css>/html/portlet/search/css/main.css</header-portlet-css>
    <css-class-wrapper>portlet-search</css-class-wrapper>
    <add-default-resource>true</add-default-resource>
  </portlet>
</liferay-portlet-app>
```

### Portlet Execution Method

One of the characteristics of Portlet development is that, two phases are needed: Action phase and Render phase. As mentioned before, a Portlet does not own a whole Portlet container. It only generates a fragment of it. The portal that holds the Portlet is the one responsible for generating the page by invoking one or several Portlets and adding some additional HTML around them. Usually, when a user

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>45</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

interacts with the page, the interaction is within a specific Portlet. The portal must forward the action performed by the user to that Portlet and after that it must render the whole page, showing the content of that Portlet, which may have changed, and also the content of the other Portlets. For the other Portlets in the page which have not been invoked by the user, what the portal does to get their content is to repeat the last invocation again.

It is important to be aware to pass information from the action phase to the render phase. It could be accomplished through render parameters. Invoke the `setRenderParameter` within the implementation in the `processAction`, to add a new parameter to the request that the render phase can read. The example code is as follow:

```
actionResponse.setRenderParameter("parameter-name", "value");
```

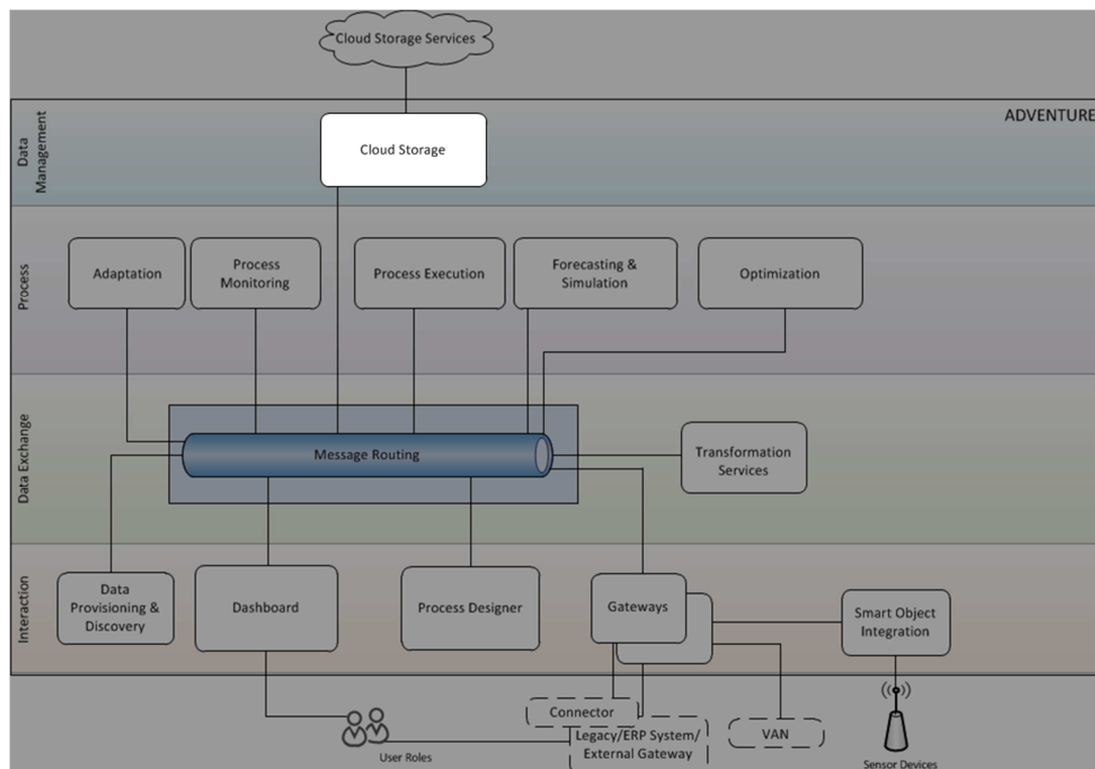
From the render phase, this value can be read using the regular parameter reading method:

```
renderRequest.getParameter("parameter-name");
```

### 3.1.5 Summary

All the ADVENTURE functionalities are presented through the Dashboard. The Dashboard component will use Liferay platform as the Enterprise Portal to create standardized and consistent UI. Each component can create a separate Portlet to display its functionalities. With the purpose to ensure seamless communication and cooperation among all components, all the Portlets (that might be implemented with different technologies) are added to the Dashboard based on Liferay platform. Liferay is configured by a combination of settings and the information is stored in its database. Supplementary technologies such as Vaadin will be selected to enhance the performance of Dashboard.

# Cloud Storage



## 3.2 Cloud Storage

Abstract:

Cloud Storage provides independent buckets<sup>3</sup> to store binary, (semi-)structured and semantic data

It supports simple CRUD (create, read, update, delete) operations for all bucket types and advanced queries (OData, SparQL,...) for (semi-)structured and semantic data

Each component can create multiple buckets. Buckets can be shared among different components and can restrict their access via simplistic access control options.

### 3.2.1 Major Design Decisions

Cloud Storage has to manage data (binary, (semi-)structured or semantic) provided by several components. The component will offer simple CRUD (create, read, update, delete) operations for all types of data, e.g. returning values of a data set or updating properties of a data object. Additionally, the Cloud Storage will provide more advanced interfaces for structured and semantic data, supporting more complex queries.

The Cloud Storage will provide a simplified access control list-feature (ACL), which may also be used by other ADVENTURE components for controlling data access.

These different data types usually have different requirements. For example, storing large amounts of binary data files requires optimal data throughput while a semantic database needs to enable graph-based queries, while data-throughput is a negligible factor. Consequentially several technologies for the different data types will be used as a base by the Cloud Storage component.

Since the Cloud Storage has to store data from several components, all data will be stored in so called buckets. One component may use several different data buckets and may specify their access level (private, publically writable, publically readable) for sharing buckets with other components. Each bucket has a “bucket type”, which specifies the nature of the storage space, e.g. “binary data storage” or “semantic data storage”.

---

<sup>3</sup> A bucket is a data storage space which is fully isolated from other spaces and may be used by a component to store its data.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>48</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

## Bucket Types

Within ADVENTURE, a set of four different bucket types will be implemented:

**Structured Data Bucket Type:** This will be used to store typical application data such as settings or component data and will provide a table based structure on top of relational databases. Potential base technologies range from MySQL to Postgres.

**Semi-Structured Data Bucket Type:** This bucket type will be used to store typical data in a document-oriented way without a fixed data schema. It is usually referred to as “NoSQL” storage and may be realized by technologies such as CouchDB or Apache Cassandra.

**Binary Data Bucket Type:** This bucket type will allow a document-centric storage for binary data. It may be used to e.g. store videos, PDFs or any other type of binary data. – Queries will be based on the document name or ID, e.g. by requesting the content of document “brochure.pdf”. Potential base technologies include Amazon S3 or (distributed) file systems.

**Semantic Data Bucket Type:** This bucket type will allow the storage of semantic information, e.g. for managing semantic factory descriptions. – Queries will be based on a semantic query language such as SparQL. Possible base technologies include Jena or Sesame.

### 3.2.2 Technology Comparison and Selection

This subsection will compare existing technologies for the Cloud Storage component. Those technologies will be used as a base for the development phase.

#### 3.2.2.1 Selection Criteria

The selection criteria for Cloud Storage technologies were defined in *D3.2 Functional Specification* in section 5.3.5 on page 81. The description of the generic parameters can be found in *D3.2 Functional Specification* in section 5 on page 28. Now follows the description of the Cloud Storage specific parameters:

**Scalability:** Since ADVENTURE may need to manage a large amount of members that will grow continuously, it is necessary to store the increasing amount of data in an efficient way.

**ACID compliant:** ACID stands for atomicity, consistency, isolation and durability. This properties guarantee that a transaction is processed reliably by a database.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>49</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- Atomicity means that in case of multiple related operations either all will be executed or nothing. If an error occurs for one of these operations, all operations in this transaction will be rolled back.
- Consistency means that it has to be guaranteed that after an operation all the data in the database is correct, if the data was correct before it.
- Isolation defines how changes will be executed in case of multiple operations. This prevents operations from working with outdated data and from overriding the actual and correct data.
- Durability guarantees that the data will be stored permanently after a successful transaction.

**Redundant data storage:** Data redundancy means that the same data is unnecessarily stored several times in the database. In relational database systems it means for example a field that is repeated in two or more tables. The aim is to reduce or even eliminate data redundancy, unless it is created for backup or replications, which are intended. For the selection this criterion will be ignored, because the data redundancy is not dependent on the used technology, but on the defined data schema.

**CRUD operations:** CRUD stands for create, read, update and delete and forms the base operations of a database management system.

**Backup:** Backups of the data can be created and stored in other backup storages.

**Replications:** Replication means the storage of the same data at several data storages and the synchronization between them.

**Relations / References:** Relations or references reduce the redundant data, because data can be stored once and referenced afterwards with for example a foreign key relationship.

**Transactions:** Transaction is defined as a sequence of database operations that belong together and can be considered as a logical unit. In execution they guarantee the consistency of the database in case of errors.

**Costs for data storage:** From an economic point of view the costs for the data storage should be included in the selection decision.

**Version control (Binary storage):** Version control stores older versions of documents and binary files and provides the possibility to restore them, if needed.

**Large file support (Binary storage):** Virtual Factories might need to exchange documents like specifications, with many images and definitions, for example to

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>50</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

describe the parts that they need. Hence, the binary data storage should support the storing of large files in an efficient way.

**Tagging (Binary storage):** Tagging provides the possibility to name binary files, which simplifies the data search.

**Key/Value storage (Semi-Structured Storage):** Key/value storage serves for storing simple data without big structure in a very efficient way.

**Schema less (Semi-Structured Storage):** Databases that work schema less are more flexible to add new data properties.

**SparQL support (Semantic storage):** In order to create semantic queries, the semantic storage should support SparQL, which is the quasi-standard language for semantic requests.

### 3.2.2.2 Possible Technologies

For the realization of ADVENTURE Cloud Storage component it is needed to identify adequate technologies to store binary, structured, semi-structured and semantic data. The assessment of these data storages will be carried-out independently. Additionally an open data format has to be found to transfer the queries and data between the Cloud Storage and the other ADVENTURE components.

*Note: The selection in this subsection is performed based on the different bucket types (structured data, semi-structured data, binary data and semantic data). For managing those data types, a lot of proven technologies exist. For example, the market provides several hundred different ways for storing structured information in relational databases. As such, the following selection will compare only the 4-6 most promising technologies for each area in order to keep the document sharp and short.*

#### 3.2.2.2.1 Structured Data

Structured data will represent data that is manageable in tables and rows, which are typical to relational databases. As such, the Cloud Storage component will reuse a relational database technology as a base for its data management. For managing structured information, the following technologies are compared:

**MySQL** – MySQL is the world's most used Open Source relational database management systems. It is very stable and runs on several operation systems as Mac OS, Linux, Windows, iOS and even Symbian. Additionally many tools exist to manage it.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>51</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Google Cloud SQL** – Google Cloud SQL is part of Google’s App Engine. Within this App Engine offers a MySQL database with JDBC support. All administration and maintenance activities are done by Google. A high reliability and availability is ensured by data replication to multiple data centers.

**Microsoft Azure SQL database** – The Microsoft Azure SQL database is a cloud based relational database offered by Microsoft. It based MS SQL server technology. The data is replicated to multiple servers and the storage scales regarding the user requirements.

**PostgreSQL** – PostgreSQL is an object-relational database management system, which is available for several systems. It is largely compliant with the ANSI SQL standard SQL 2008.

The following table (Table 3) shows all potential technologies as well as their rating:

Table 3 – Technology selection criteria and comparison of technologies for Structured Data Storage

Parameter	Importance (-, --, +/-, +++)	MySQL	Amazon RDS	Google Cloud SQL	MSSQL Server	Microsoft Azure SQL Database	PostgreSQL
<b>Generic Parameters</b>							
Maturity & Stability	+++	+++	+++	+++	+++	+++	+++
Regularly Updated	+	+++	+++	+++	+++	+++	+++
Technical Up-to-Dateness / Appeal	+	+++	+++	+++	+++	+++	++
Open Source	+/-	YES	YES	YES	NO	NO	YES
D3.3-Technical-Specification				Author: UVA and Partners		Date: 2012-09-24	Page: <b>52</b> / 372

Copyright © ADVENTURE Project Consortium. All Rights Reserved.



Non-Infecting	++	NO	NO	NO	NO	NO	YES
Code-Quality	+	N/A	N/A	N/A	N/A	N/A	N/A
Extensibility	+	+/-	+/-	---	---	---	++
Community	+	+++	+++	+++	+++	++	+++
Performance	+++	+++	+++	+++	+++	+++	++
Reuse of existing developments	++	+++	+++	+++	+++	+++	+++
EU project origin	+/-	NO	NO	NO	NO	NO	NO
Platform (Portability)	++	+++	N/A	+++	---	N/A	+++
Open Standards Compliance	+	+++	N/A	+++	+	N/A	++
Interoperability	++	+++	+++	+++	+	+	++
<b>Specific Parameters</b>							
Scalability	+	++	++	++	++	++	++
ACID compliant	+++	+++	+++	+++	+++	+++	+++
Redundant data storage	++	+/-	+/-	+++	+/-	+++	+/-
CRUD operations	+++	+++	+++	+++	+++	+++	+++
Backup	-	+/-	++	+/-	+/-	+/-	+/-
Replications	-	+/-	++	++	+/-	+++	+/-
Relations / References	++	YES	YES	YES	YES	YES	YES
Transactions	+/-	YES	YES	YES	YES	YES	YES
Costs for data storage	+	+/-	+/-	+/-	+/-	+/-	+/-

### 3.2.2.2.1.1 Conclusion

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>53</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### 3.2.2.2.2 Semi-Structured Data

**Microsoft Azure Table Storage Service** – Microsoft Azure Table Storage is a NoSql database service. It serves access over a rest interface or libraries for several programming languages such as .Net, Java and PHP.

**Apache CouchDB** – CouchDB is a document-based database that can be accessed by a REST interface. It is written in Erlang and has a plug-in-architecture, which allows adding extensions. It contains a master-master merge replication mechanism.

**Apache Cassandra** – Cassandra is an Open Source distributed database management system. It is a structured key/value store and is used to store a big amount of data on different systems. It is written in Java.

*Table 4 - Technology selection criteria and comparison of technologies for Semi-Structured Data Storage*

[illegible]

Technical Up-to-Dateness / Appeal	+	++	++	++	+	+
Open Source	+/-	NO	NO	YES	YES	YES
Non-Infecting	++	NO	NO	YES	YES	YES
Code-Quality	+	N/A	N/A	N/A	N/A	N/A
Extensibility	+	---	---	+++	++	+++
Community	+	+/-	+	+++	++	++
Performance	+++	+++	+++	+++	+++	+++
Reuse of existing developments	++	+++	+++	+++	+++	+++
EU project origin	+/-	NO	NO	NO	NO	NO
Platform (Portability)	++	N/A	N/A	+++	++	+++
Open Standards Compliance	+	N/A	N/A	NO	NO	NO
Interoperability	++	+++	+++	+++	++	+/-
<b>Specific Parameters</b>						
Scalability	+	+++	+++	+++	+++	+++
ACID compliant	+++	N/A	N/A	N/A	N/A	N/A
Redundant data storage	++	+	+	+	+	+
CRUD operations	+++	+++	+++	+++	+++	+++
Backup	-	N/A	N/A	N/A	N/A	N/A
Replications	-	+++	+++	++	+++	+++
Relations / References	++	+++	+++	+++	+++	+++
Transactions	+/-	N/A	N/A	N/A	N/A	N/A
Costs for data	+	+/-	+/-	+/-	+/-	+/-

storage						
Key/Value storage ((Semi-)Structured Storage)	+++	YES	YES	YES	YES	YES
Schema less ((Semi-)Structured Storage)	+++	YES	YES	YES	YES	NO

### 3.2.2.2.1 Conclusion

To store semi-structured data MongoDB will be used, because it is very stable and has a good performance. Additionally, it can be run on an own server, an own private cloud or on Microsoft Azure as public cloud.

### 3.2.2.2.3 Semantic Data

**Sesame** – Sesame is an Open Source de-facto standard framework for RDF data. It supports two query languages, SeRQL and SparQL. It runs in a Java virtual machine and uses one of the database management systems to store the data, PostgreSQL, MySQL, Microsoft SQL Server or Oracle database.

**Virtuoso Universal Server** – Virtuoso is multi-model cross-platform data server. It contains RDF data management with SparQL support.

**AllegroGraph** – AllegroGraph is a commercial RDF database. It exists as a free version with a limitation of 5 million triples. It runs on Windows, Linux and OS X and serves libraries for several programming languages including C# and Java.

**Jena** – Jena is an Open Source RDF framework. It supports SparQL as requesting language. It is very similar to Sesame with the exception that it supports a better support for reasoning, but for this reason the scalability suffers, so that Sesame is the better choice in case of a large amount of data.

**BrightstarDB** – BrightstarDB is a schema-free RDF store for the .Net platform. It supports SparQL 1.1 and OData and it is ACID compliant.

*Table 5 - Technology selection criteria and comparison of technologies for Semantic Data Storage*

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>56</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Parameter	Importance (- - - +/- +++)	Sesame	Virtuoso	Allegro Graph	Jena	Brightstar DB
<b>Generic Parameters</b>						
Maturity & Stability	+++	+++	+++	+++	+++	+
Regularly Updated	+	+	+	+	+	+
Technical Up-to-Dateness / Appeal	+	++	++	++	++	++
Open Source	+/-	YES	YES	NO	YES	YES
Non-Infecting	++	YES	NO	N/A	YES	YES
Code-Quality	+	N/A	N/A	N/A	N/A	N/A
Extensibility	+	++	+	---	++	++
Community	+	++	+	+/-	++	-
Performance	+++	+++	+++	+++	+++	+++
Reuse of existing developments	++	+++	+++	+++	+++	+++
EU project origin	+/-	NO	NO	NO	NO	NO
Platform (Portability)	++	+++	+++	+++	+++	-
Open Standards Compliance	+	NO	NO	NO	NO	NO
Interoperability	++	+++	++	+++	+++	-
<b>Specific parameters</b>						
Scalability	+	++	++	++	+	+++
ACID compliant	+++	N/A	N/A	N/A	N/A	+++
Redundant data storage	++	++	++	++	++	+++
CRUD operations	+++	+++	+++	+++	+++	+++

Backup	-	-	-	+	-	-
Replications	-	-	-	+	-	-
Relations / References	++	+++	+++	+++	+++	+++
Transactions	+/-	N/A	N/A	N/A	N/A	YES
Costs for data storage	+	N/A	N/A	N/A	N/A	N/A
SparQL support (Semantic storage)	+++	YES	YES	YES	YES	YES

### 3.2.2.2.3.1 Conclusion

Sesame was selected as semantic storage, because it is very stable, Open Source and it is a de-facto standard to store RDF.

### 3.2.2.2.4 Binary Data

Binary Data contains all data stored in files such as documents, images or configuration files of applications. For that reason, technology is needed which offers an easy to use and scalable storage. For storing binary data, the following technologies are compared:

**Amazon Simple Storage Service (S3)** – Amazon S3 is a key value storage which is accessible through Web service interfaces like REST and SOAP. Amazon provides scalability, high availability and promises an availability of 99.99%.

**Microsoft Azure Blob Storage Service** – MS Azure Blob Storage offers a service to store large binary files in the Cloud. All stored files are replicated automatically. The storage is available via Web services interfaces.

**File server** – A file server is a computer, which provides a shared access to storage resources. It is attached to a network and accessed by attached workstations. It could be installed in a dedicated or in a non-dedicated manner. In the first case, the dedicated server is specially configured as files service. There are several ways and protocols to access the data, i.e. FTP or HTTP.

**Apache Subversion (SVN)** – SVN is an Open Source versioning system which is distributed under an Apache License. It allows maintaining different versions of

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>58</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

source code or documents. A SVN repository can also be installed on an Amazon EC2 instance, so it could be used in local and cloud environments.

*Table 6 - Technology selection criteria and comparison of technology for Binary Data Storage*

Parameter	Importance (- - - +/- +++)	Amazon S3	Microsoft Azure Blob Storage Service	Fileserver
<b>Generic Parameters</b>				
Maturity & Stability	+++	+++	+++	+++
Regularly Updated	+	N/A	N/A	N/A
Technical Up-to-Dateness / Appeal	+	+++	+++	+++
Open Source	+/-	NO	NO	YES
Non-Infecting	++	NO	NO	NO
Code-Quality	+	N/A	N/A	N/A
Extensibility	+	+++	+++	+
Community	+	+++	++	++
Performance	+++	+++	+++	+/-
Reuse of existing developments	++	+++	+++	+++
EU project origin	+/-	NO	NO	NO
Platform (Portability)	++	N/A	N/A	+++
Open Standards Compliance	+	N/A	N/A	+
Interoperability	++	+++	++	+
<b>Specific parameters</b>				
Scalability	+	+++	+++	+/-
ACID compliant	+++	N/A	N/A	N/A

Redundant data storage	++	+++	+++	+/-
CRUD operations	+++	YES	YES	YES
Backup	-	YES	NO	NO
Replications	-	YES	YES	YES
Relations / References	++	N/A	N/A	N/A
Transactions	+/-	N/A	N/A	N/A
Costs for data storage	+	+/-	+/-	+/-
Version control (Binary storage)	+	YES	NO	NO
Large file support (Binary storage)	++	YES	YES	YES
Tagging (Binary storage)	+/-	NO	NO	NO

#### 3.2.2.2.4.1 Conclusion

Amazon S3 was chosen as a binary storage, because it offers a high availability and scalability. Additionally it is a broadly established solution for binary data storage.

#### 3.2.2.3 Technology Selection

**Structured data:** MySQL database was selected to store structured data, because it is very stable, well tested and Open Source. Additionally, it has a good performance, scalability and as it can be seen in Table 6 covers all the criteria. It can be also used with the OData Format. In addition, by using MySQL it is possible to migrate data easily between different Cloud Providers and local systems with minimal migration effort. This ensures high data portability which enables a high flexibility in future development and avoids a vendor lock-in.

**Semi-structured data:** MongoDB was selected as semi-structured data storage, because it is very stable and has a good performance. It is internally in use by at least one project partner that is involved in the development of the Cloud Storage. Thus, it is well tested and the necessary know-how is still available. It is Open Source and uses the GNU Affero General Public License<sup>4</sup>, a non-infecting license. It is

<sup>4</sup> <http://blog.mongodb.org/post/103832439/the-agpl>



available for Windows, Linux, OS X and Solaris and drivers exist for many programming languages. The number of drivers continues to grow, because the community develops new ones for further programming languages. It can be run on an own server, an own private cloud or on Microsoft Azure as public cloud. The database supports replications by itself, so that the synchronization is still integrated.

**Semantic data:** Sesame was selected to store semantic data, because it is a de-facto standard to store RDF. It works with several RDMS and is platform independent. It has a better performance then Jena for large amount of data and a better stability then BrightstarDB. It is Open Source and uses LGPL, a non-infecting license. At least one partner has used it, so that the implementation can be supported with some know-how.

**Binary data:** Amazon S3 was selected as binary storage, because it offers a good interoperability by standardized interfaces like REST or SOAP. It offers high availability and scalability. The account on the services is performed in a pay-as-you-use manner and thus there are no up-front costs for initial hardware purchase required. Further, it is a broadly established solution for binary data storage and is well documented and with a large community.

#### 3.2.2.3.1 Data Protocols

The data will need to be queried and retrieved. For this purpose, an easy to use API will be provided by the Cloud Storage component for basic CRUD operations, allowing users to receive an element by its ID, to store new elements or to update/delete elements. Additionally, more advanced query facilities will be provided for some bucket types:

**Structured Data Bucket Types:** Structured data bucket types will be supporting basic CRUD operations. Additionally, queries may be performed using the OData standard to abstract from the used database and database request language. This increases the flexibility of the Cloud Storage to add or exchange databases.

**Semi-Structured Data Bucket Types:** Those bucket types will be supporting basic CRUD operations. Additionally, queries may be performed using the self-defined xml structure. This XML structure will be used to abstract from the database and database request language. This reduces the dependence of the Cloud Storage from a specific database or database language.

**Binary Data Bucket Types:** Binary data bucket types will be supporting basic CRUD operations only.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>61</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Semantic Data Bucket Types:** Semantic data bucket types will be supporting SparQL queries allowing complex query creation.

#### 3.2.2.4 Missing Elements and Implementation Needs

The selected technologies only cover the basis for storage and a part of the data transmission. The whole logic around it is missing and has to be implemented.

Interfaces to Message Routing have to be implemented to provide the communication with ADVENTURE.

The bucket creation and management is missing for each database type. The translation from and to the message protocol has to be implemented for SQL, NoSQL, binary and semantic queries. The access control list has to be provided. Based on it the bucket based internal access control and user based access control for binary storage has to be implemented.

For binary data the version control feature has to be created and integrated in Amazon S3.

The administration UI to configure the Cloud Storage component and the access list has to be provided and integrated in the Dashboard.

The Data Archiving Engine has to be implemented and integrated into the selected storage types.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>62</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

3.2.3 Technical Component Specification

3.2.3.1 Component Structure

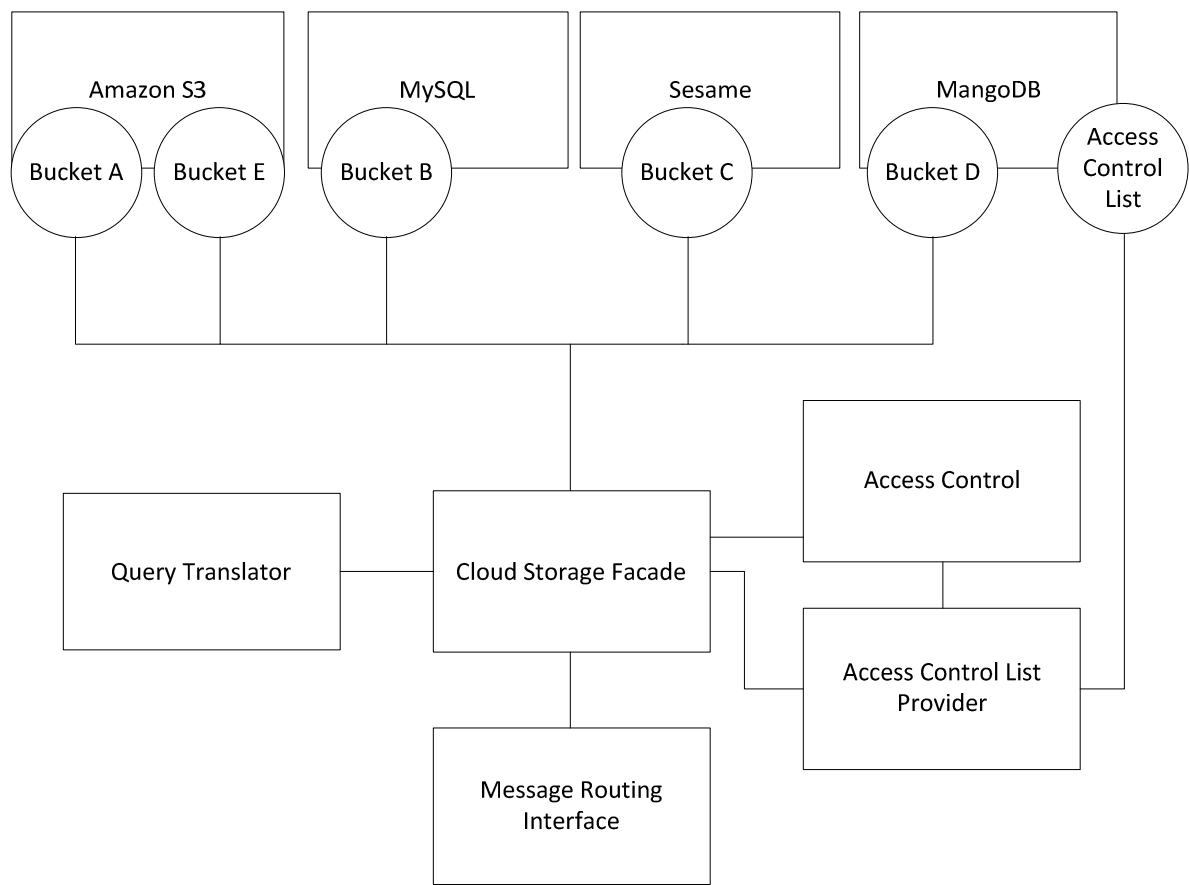


Figure 12 - Cloud Storage component structure

The Cloud Storage consists of the following parts:

**Message Routing Interface:** This implements the Message Routing API (See section 3.4.3.2) and realizes the communication with ADVENTURE. An event handler will be triggered, if a message will be received and the Cloud Storage Facade will start to handle the message.

**Cloud Storage Façade:** The Cloud Storage Façade encapsulates the main logic of the Cloud Storage component. Everything converges at this point. It manages the buckets, interprets the messages and executes the commands sent in the message. Additionally, it checks, whether the data has to be transformed and if the needed

access rights are granted. To achieve this it uses the Query Translator and the Access Control.

**Query Translator:** The Query Translator converts the data from the messaging format into the specific database query format and back.

**Access Control:** All access control will be used by the Cloud Storage Facade and checks, if components are authenticated to access a specific bucket. Additionally, it checks for binary storage, whether the users have the rights to access specific binary data. To get the component and user rights the Access Control uses the Access Control List Provider.

**Access Control List Provider:** This provides the rights of the component or user given by his identifier.

**Access Control List:** The ACL is stored in a (semi-)structured bucket, because it is the most suitable due to the document structure and the easy schema expandability, where a subject can be represented as one document.

**Buckets:** Buckets are independent data storages that can be created and used by components. Depending on the type they are realized as own database instances, separate tables or keys with a specific prefix. Each component can create an own separate bucket, so the Cloud Storage has to manage several buckets of each type, as it can be seen in Figure 12, where two binary buckets exist.

### 3.2.3.2 Internal Sequence Diagrams

This section will give an insight about the internal component communication sequence, based on a list of example sequences. It's extracted from the use case scenarios of D3.2.

#### Bucket based access control

As shown in Figure 13 a component can access a bucket only if the Access Control authorizes the component. This will be the case if the bucket has a specific “public” flag (publicRead or publicWrite) or if it exists a specific entry in the ACL. For this purpose for each bucket access the Cloud Storage control checks the access rights of the component, using the Access Control subcomponent. The Access Control reads the rights for the given component from the ACL and checks if the needed right exists. Then it returns a Boolean to show whether the access is granted.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>64</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

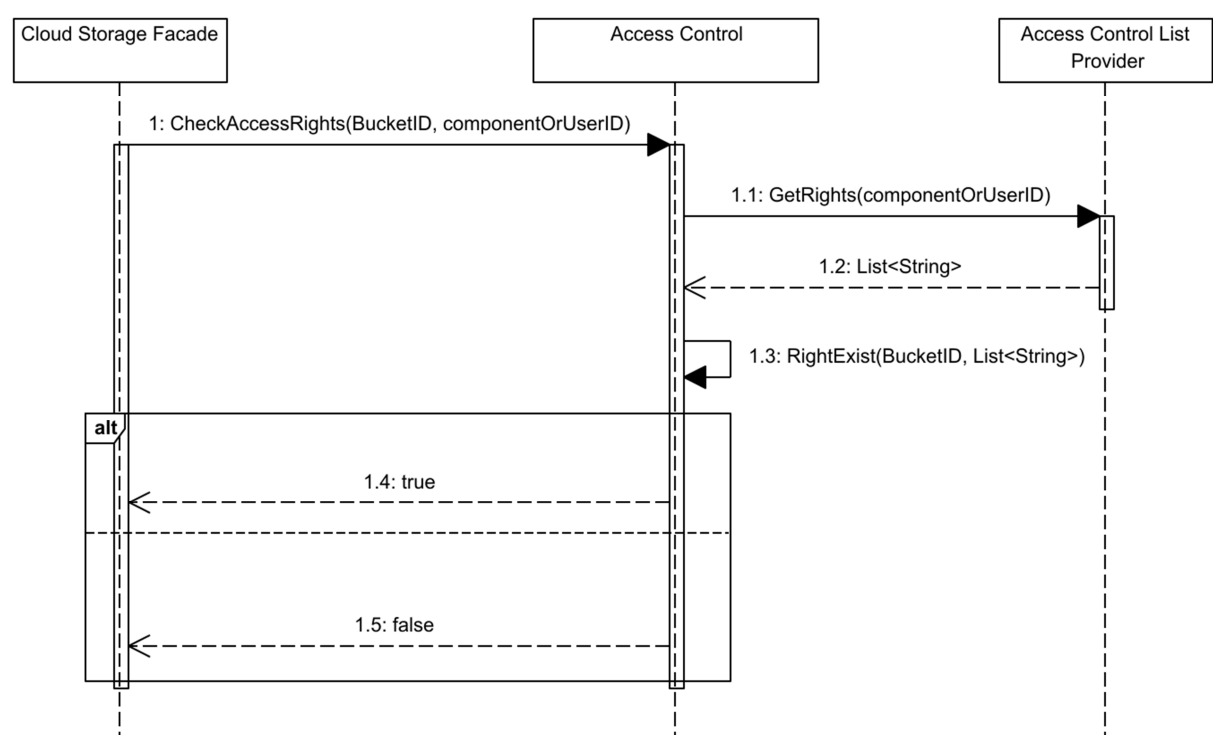


Figure 13 - Sequence diagram for bucket based access control

CreateBucket

In this scenario a component sends a message to create a binary bucket. The Message Routing Interface throws an event that a message has been received. The Cloud Storage Facade sub-component checks, which kind of bucket have to be created. A Binary bucket will be created in Amazon S3 and returns a BucketID. Then the Cloud Storage Facade sets the needed write and read rights for the component that has sent the message and it returns the BucketID to it.

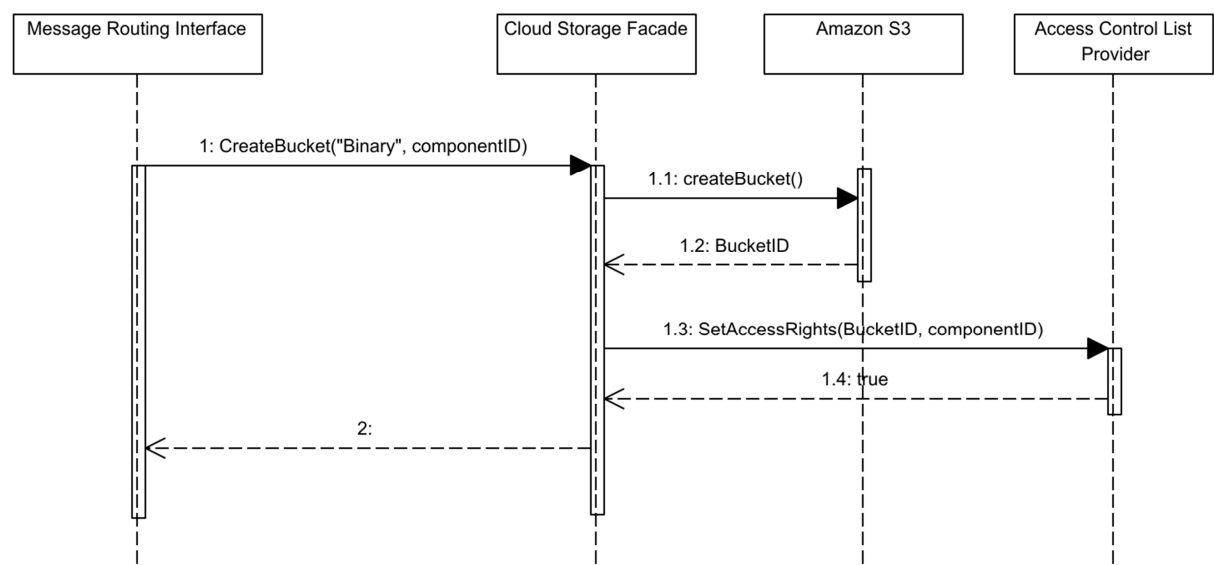


Figure 14 - Sequence diagram for bucket creation

DeleteBucket

Only the component that has created the bucket has the right to delete it. The entry in the ACL will be set during the creation. If an administrator added specific right to another component it can delete the bucket too. In this scenario a component sends the corresponding message to the Cloud Storage component. The Cloud Storage Facade sub-component checks the needed rights. If the sender has sufficient rights the bucket will be deleted and a success will be notified back. Otherwise an unsuccessful notification will be sent back.

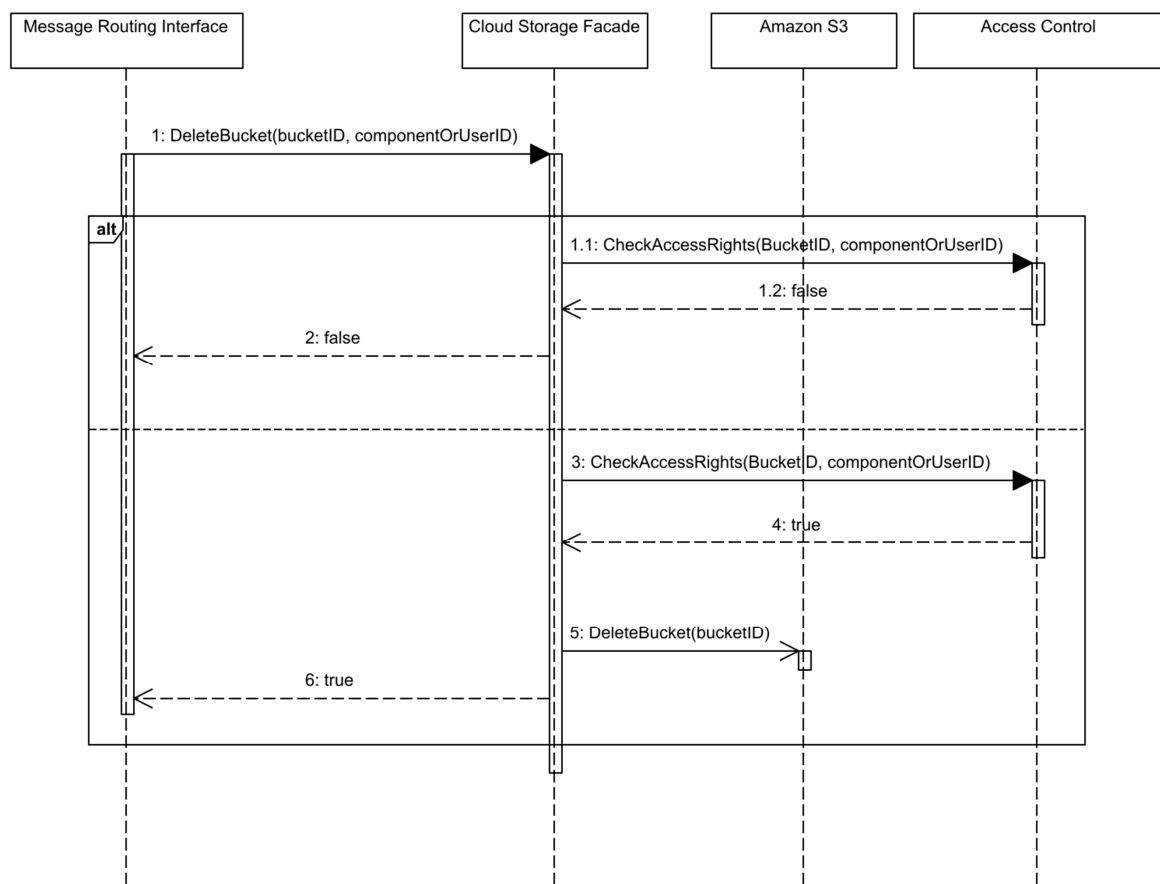


Figure 15 - Sequence diagram for bucket deletion

GetAccessRights

Each component can request the stored user rights in the ACL. To get the List it is needed to transmit the Id of the user. It sends the corresponding message with the component or user identifier. The Cloud Storage Facade gets the rights from the ACL and sends them back.

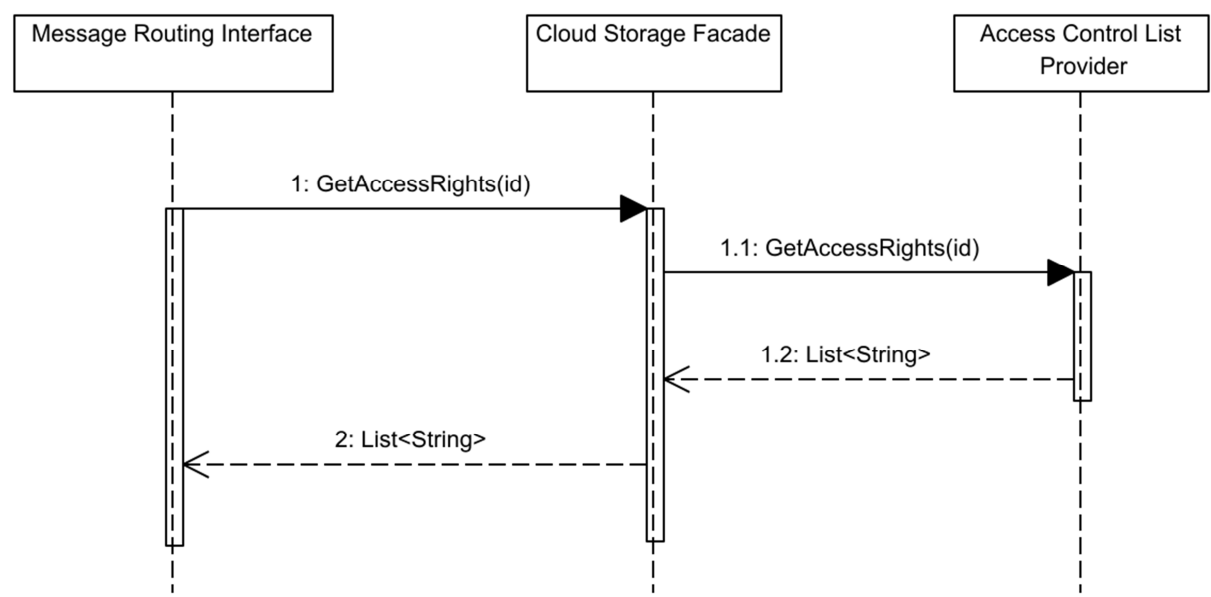


Figure 16 - Sequence diagram for access rights request

SetAccessRights

Every user that has administrator rights can add, delete or change the entries in the ACL. When a message to change access rights is received, it will be checked first, whether the user has sufficient rights to change the Access List. If this is not the case, “False” will be sent back. Otherwise the rights will be set and “True” will be sent back.



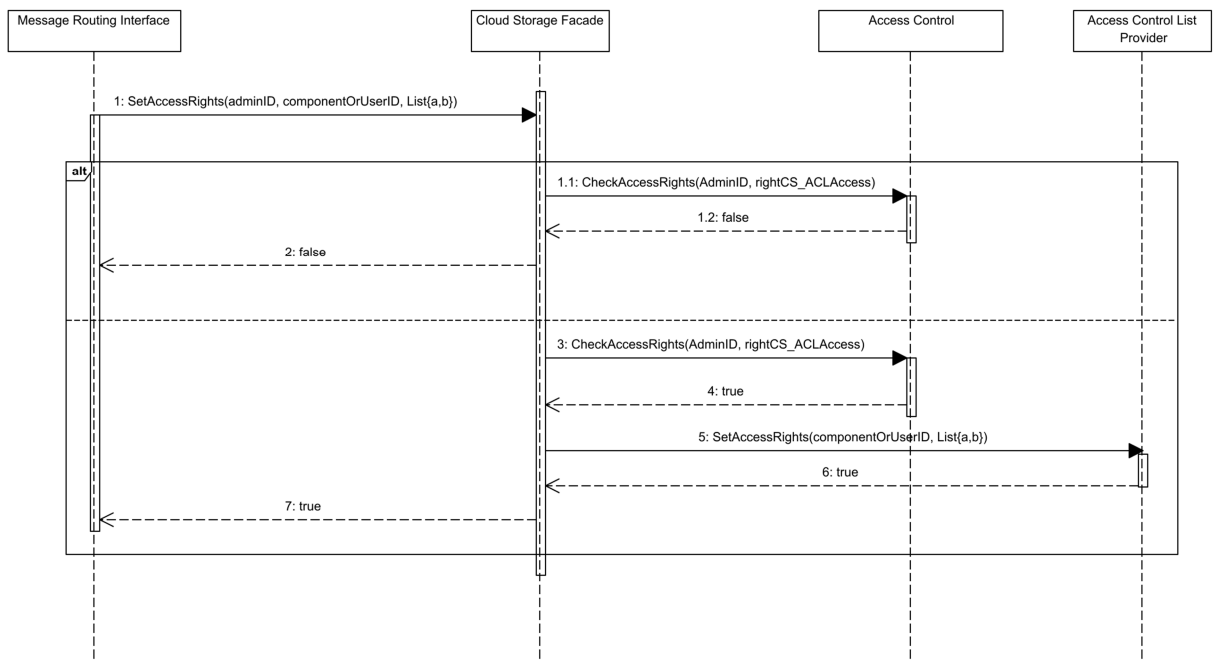


Figure 17 - Sequence diagram for adding and updating access rights

ExecuteQuery

In the scenario in Figure 18 a component sends a message to query the SQL bucket. Receiving the message, the Cloud Storage Facade checks the access rights of the sender. If the sender has insufficient rights it sends a “False” response. Otherwise it forwards the message to the Query Translator, which extracts the query from the message and translates the query to the database format, if needed. In this scenario the message contains the query in OData format and a SQL query is needed. So the Query Translator transforms it and returns the SQL query. After that the Cloud Storage Facade executes the SQL query and gets the response back from the database. This response will be passed to the Query Translator to transform it to OData format and create the response message in the defined XML response format. This message will be sent back.

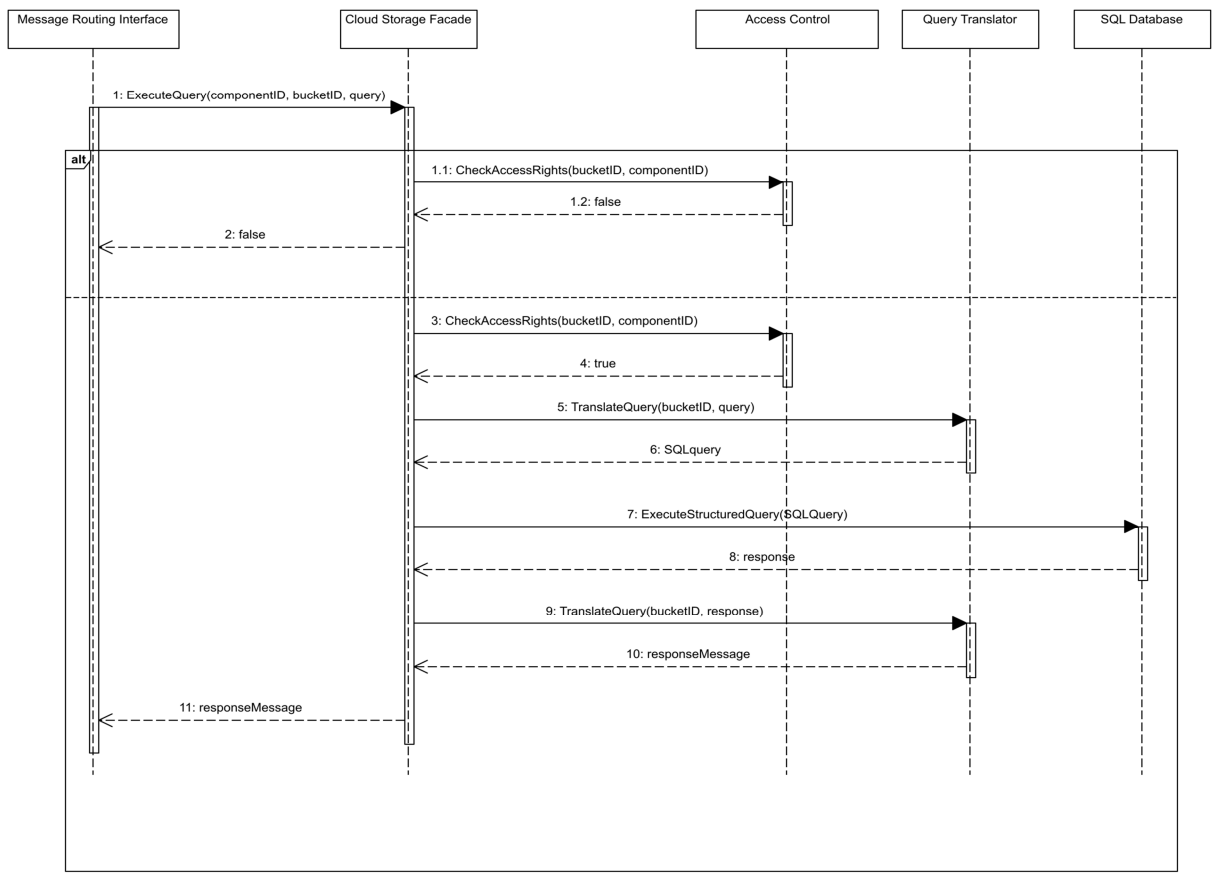


Figure 18 - Sequence diagram for bucket querying

3.2.4 Specification of Interfaces, Protocols and Formats

To communicate with the other ADVENTURE components the Cloud Storage uses the Message Routing API. The interfaces will be called using commands within the message. To clarify the functionality of the Cloud Storage component the internal API specification will be shown here, that will be used after the interpretation of the message. Section 3.2.4 shows the communication protocol for Cloud data queries.

3.2.4.1 API specification

The Cloud Storage component will provide a list of API methods that can be indirectly used by the other ADVENTURE components via the message routing component. The API methods are defined as follows:

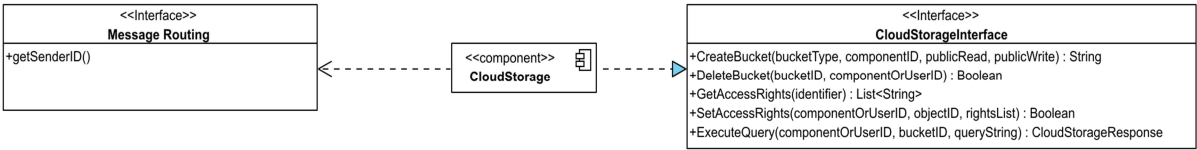


Figure 19 - Cloud Storage Message Interface

CreateBucket

This method will create a new bucket for an application for managing data:

```
public String CreateBucket( String bucketType, String componentID,
    Boolean publicRead, Boolean publicWrite )
```

Parameters

- bucketType: Type of the bucket that should be created.  
values=“Structured”, “SemiStructured”, “Binary”, “Semantic”
- componentID: Identifier of the component that wants to create the bucket  
It will be provided by the Message Routing component.
- publicRead: Flag, for public read access. Default: false
- publicWrite: Flag, for public write access. Default: false

Return Value

The bucket identifier

Remarks

The access rights for the component will be set automatically by the Cloud Storage component in the ACL.

Message-Example

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudRequest xmlns:tns="http://www.fp7-adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd">
  <tns:createBucket>
```

```
<tns:bucketType>Structured</tns:bucketType>
<tns:publicRead>false</tns:publicRead>
<tns:publicWrite>false</tns:publicWrite>
</tns:createBucket>
</tns:cloudRequest>
```

**Response:**

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudResponse xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd ">
  <tns:createBucket>
    <tns:success>true</tns:success>
    <tns:bucketId>BUCKET1234</tns:bucketId>
  </tns:createBucket>
</tns:cloudResponse>
```

**DeleteBucket**

This method will delete a bucket including all its data.

```
public Boolean DeleteBucket( String bucketID, String componentOrUserID)
```

**Parameters**

bucketID: Identifier of the bucket that should be deleted

componentOrUserID: Identifier of the component or user that wants to delete the bucket

**Return Value**

True, if bucket is deleted, otherwise false

**Message-Example****Request:**

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudRequest xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd ">
  <tns:deleteBucket>
    <tns:bucketId>BUCKET1234</tns:bucketId>
  </tns:deleteBucket>
</tns:cloudRequest>
```

**Response:**

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudResponse xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd ">
  <tns:deleteBucket>
    <tns:success>true</tns:success>
  </tns:deleteBucket>
</tns:cloudResponse>
```

**GetAccessRights**

This method will return a list of access rights for a specific bucket or user

```
public List<String> GetAccessRights( String identifier)
```

**Parameters**

identifier: Identifier of the component or user, whose access rights should be provided

**Return Value**

List with access rights, if no rights are exist an empty list will be returned

**Message-Example****Request:**

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudRequest xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd ">
  <tns:getAccessRights>
    <tns:subjectId>ID1234</tns:subjectId>
  </tns:getAccessRights>
</tns:cloudRequest>
```

**Response:**

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudResponse xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd ">
  <tns:getAccessRights>
    <tns:success>true</tns:success>
  </tns:getAccessRights>
</tns:cloudResponse>
```

```

<tns:subjectId>ID1234</tns:subjectId>
<tns:privileges>
  <tns:name>Dashboard:createUsers</tns:name>
  <tns:name>Designer:createProzesses</tns:name>
</tns:privileges>
</tns:getAccessRights>
</tns:cloudResponse>

```

### SetAccessRights

This method will add or change entries in the ACL.

```
public boolean SetAccessRights(String identifier, String identifierToChange, List<String> rights)
```

### Parameters

**identifier:** Identifier of the administrator that wants to add or change the entries.

**identifierToChange:** Identifier of the component or user, whose access rights should be added or changed

**rights:** The list with the rights that will be set for the user or component with given id. If entries for that component or user still exist, they will be overwritten. If the list is empty the rights will be deleted. The rights are defined Strings that can be defined in the components that want use them. In order to keep them apart, a prefix for the component name will be added.

### Return Value

True, if rights were set, otherwise False

### Message-Example

#### Request:

```

<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudRequest xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd">
  <tns:setAccessRights>
    <tns:subjectId>ID1234</tns:subjectId>
    <tns:privileges>
      <tns:name>Dashboard:createUsers</tns:name>

```

```
<tns:name>Designer:createProzesses</tns:name>
<tns:name>Designer:createProzesses</tns:name>
</tns:privileges>
</tns:setAccessRights>
</tns:cloudRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudResponse xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd ">
  <tns:setAccessRights>
    <tns:success>true</tns:success>
    <tns:subjectId>ID1234</tns:subjectId>
  </tns:setAccessRights>
</tns:cloudResponse>
```

ExecuteQuery

This method will execute the given query.

```
public String ExecuteQuery(String componentID, String bucketID, String query)
```

Parameters

- componentID: Identifier of the component that wants to query the bucket.
- bucketID: Identifier of the bucket
- query: The query in the specified XML format. (See section 3.2.4.1)

Return Value

Result or error in the specified XML format. If no result is expected a Boolean will be send back to show that the operation was performed successfully.

Message-Example

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cloudRequest xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd CloudStorage.xsd ">
  <tns:executeQuery>
    <tns:bucketId>BUCKET1234</tns:bucketId>
```

**Response:**

**Response error case:**

#### 3.2.4.2 Content Formats and Protocol Definitions

An *XML* document will be used to transfer data to and from the Cloud Storage via the Message Routing component. The xml-schema, shown below, contains all the parameters for invoking the API methods specified in the sections above.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>76</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



The *XML* contains a “queryData” element to include the database queries. Simple CRUD queries can be expressed using the xml-schema. For more complex SQL queries the “queryData” element will be used to transmit OData. In case of semantic queries SparQL can be embedded in the “queryData” tag and for NoSQL queries can be used the SemiStructuredMessageFormat that is defined in the xml-schema and is shown at the end of this section.

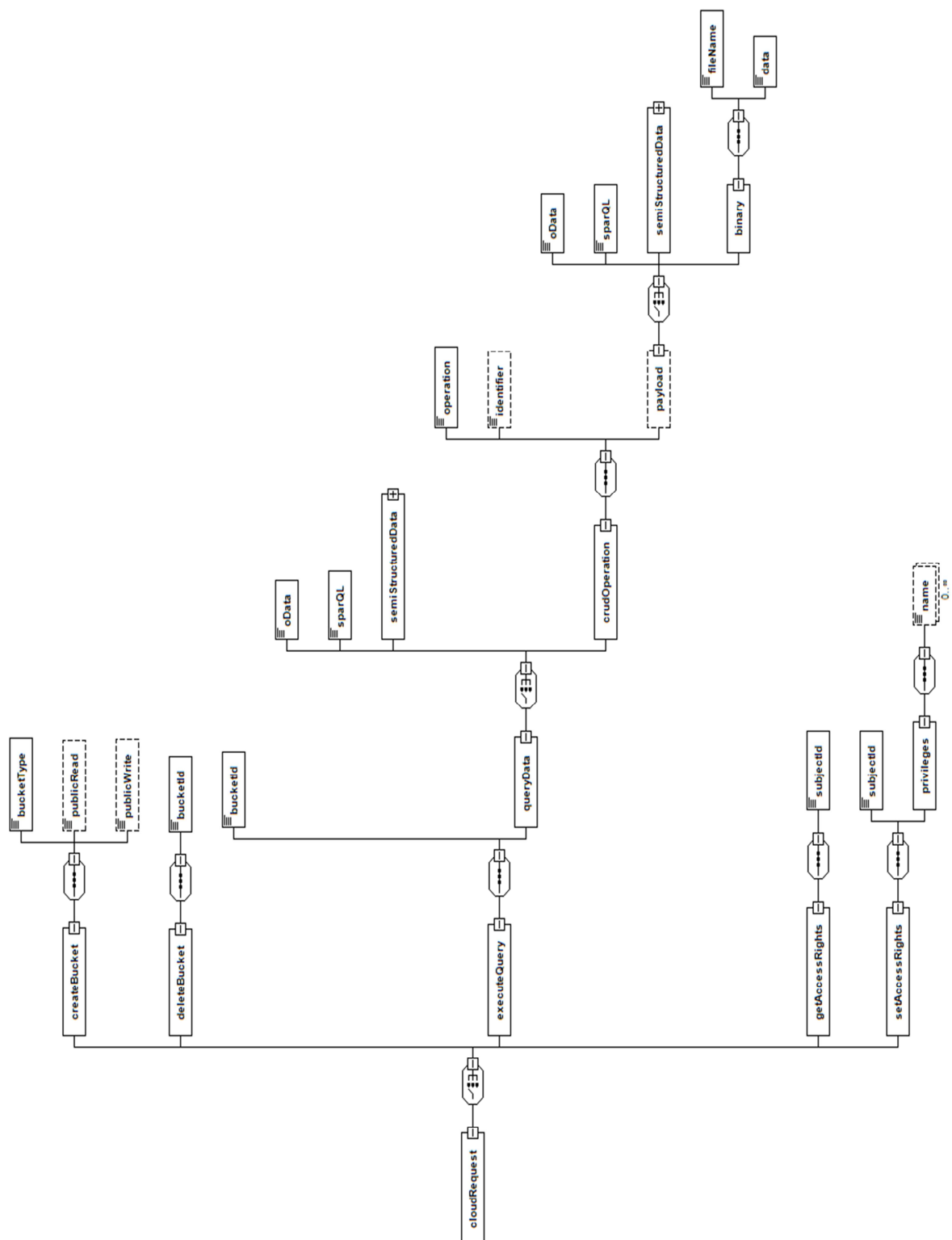
**Cloud Storage message format XML-Schema:**

*<http://www.fp7-adventure.eu/xmlSchema/CloudStorage/CloudStorage.xsd>*

**Request-Format:**

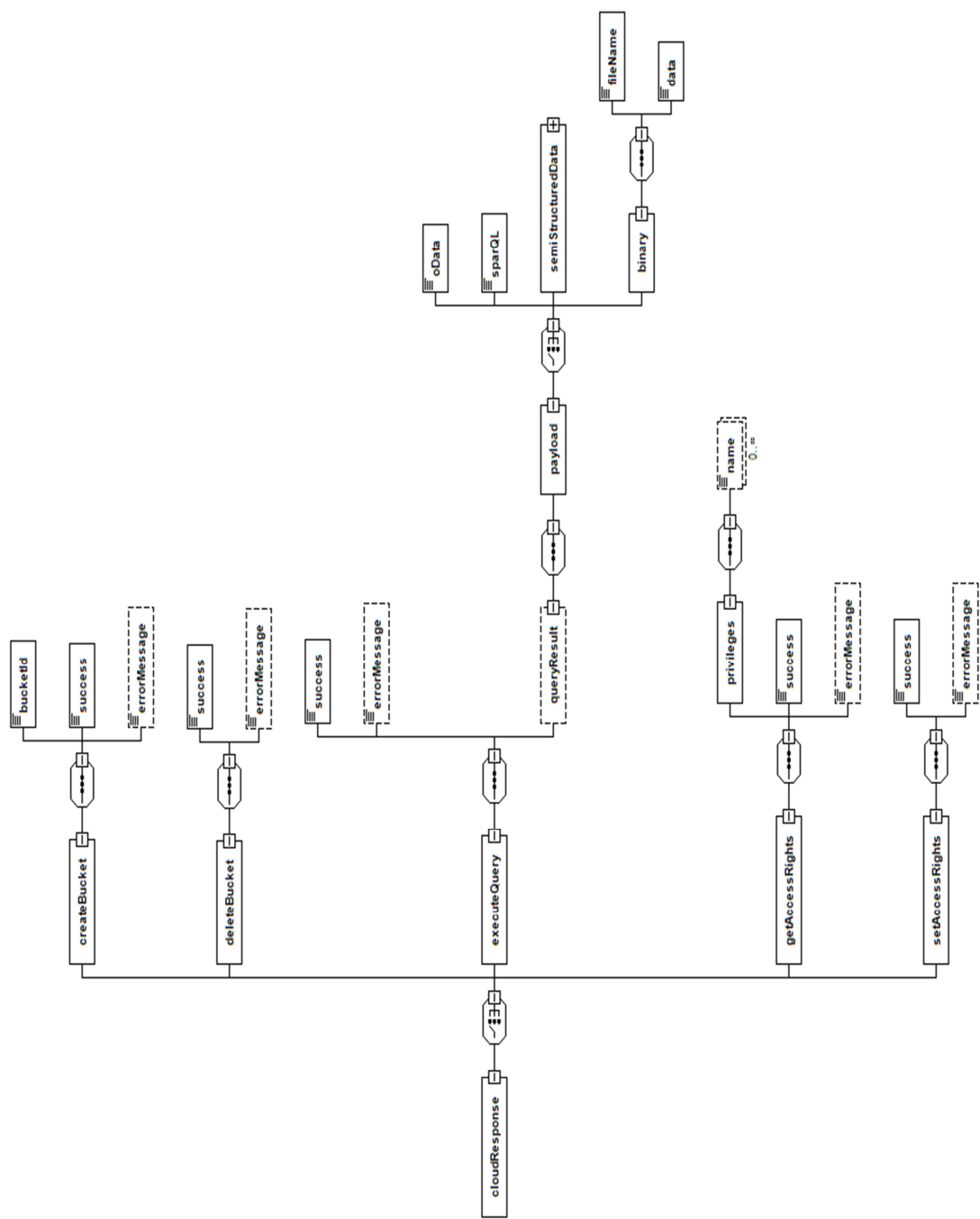
This format will be used by the other ADVENTURE components to send queries to the Cloud Storage Component. Under the root node there exist 5 sub-nodes that can be chosen depending on which interface has to be addressed.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>77</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



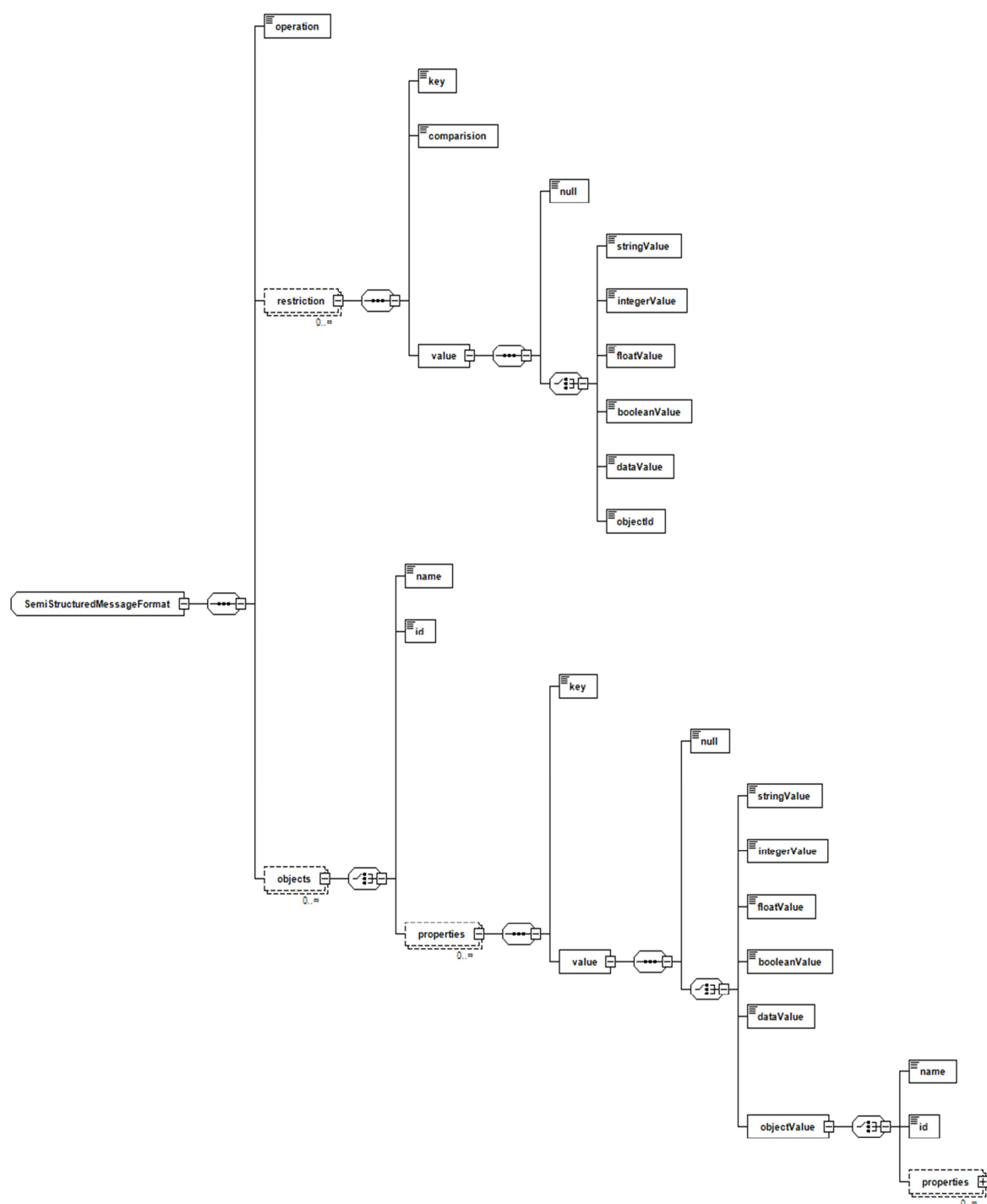
Response-Format:

This format will be used by the Cloud Storage component to send responses and data to the other ADVENTURE components. Under the root node 5 sub-nodes exist that will be used depending on the requested interface.



**SemiStructuredMessage-Format:**

This is the format to transfer Semi-Structured data. It consists mostly of Key/Value pairs, which can be grouped into objects.



### OData Example:

The result of this request shows all factories in the database.

```
Request:
http://www.fp7-adventure.eu//Organizations/Factories

Response:
<?xml version="1.0" encoding="utf-8"?>
<feed xml:base="http://www.fp7-adventure.eu//Organizations/Factories/"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns="http://www.w3.org/2005/Atom">
  <title type="text">Factories</title>
  <id>http://www.fp7-adventure.eu//Organizations/Factories/</id>
  <updated>2012-08-27T07:40:14Z</updated>
  <link rel="self" title="Factories" href="Factories" />
  <entry>
    <name>Azevedos Industria</name>
    <city>Porto</city>
    <country>Portugal</country>
  </entry>
</feed>
```

### SparQL Example:

The result of this request shows the country names and the company names of all companies that are associated to eClass Logistics.

```
PREFIX abc: <http://fp7-adventure.com/Ontology#>
SELECT ?country ?company
WHERE {
  ?x abc:countryname ?country ;
    abc:isLocationOf ?y.
  ?y abc:companyname ?company ;
    abc:isInEClass abc:Logistics .
}
```

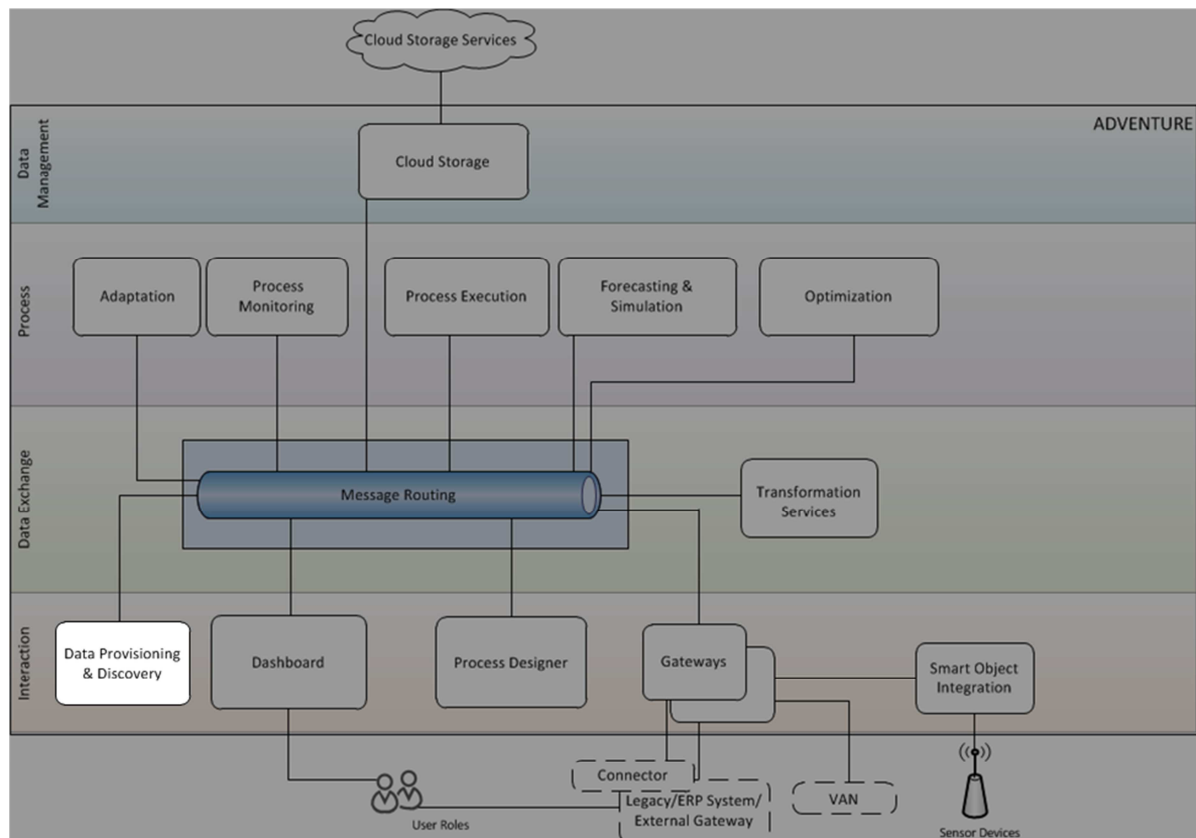
## 3.2.5 Summary

The Cloud Storage component will use MySQL, MongoDB, Sesame and Amazon S3 to store different data. Each component can create a bucket in one of these databases to store its data. A bucket can be made public to be used by several components. Alternatively, the ACL can be used to grant access for only certain components such as Process Designer and Process Execution could share Process Models. The ACL has an own interface, where rights can be added, removed and queried. So each component can use the ACL to query the rights of a specific user.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>81</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

As in all other components the communication will be realized via the Message Routing component using the defined message format.

# Data Provisioning and Discovery



### 3.3 Data Provisioning and Discovery (DPD)

Abstract:

Introduces a customizable and extensible model for describing ADVENTURE members and their ADVENTURE-related assets (e.g. services) into a common, coherent information space, supporting the other ADVENTURE tools in their information retrieval needs.

Provides tools for provisioning of new ADVENTURE members information into the system during their join phase.

Provides tools for discovery of existing ADVENTURE members and services matching selection criteria during the play phase or to find potential partners.

#### 3.3.1 Major Design Decisions

The Data Provisioning and Discovery (DPD) Module is a key infrastructure component of the ADVENTURE architecture. Almost every activity in the system relies somehow on its functionality. Therefore its reliability and scalability are major factors for the overall system quality of service. To ensure this, the module design is aligned with the fundamental design decision, shared by ADVENTURE modules, to build upon Service Oriented Architectures paradigm and the cloud technology.

Section 3.3 “Cloud Architecture” in ADVENTURE *D3.1 Global Architecture Definition* discusses the fundamental design principles that need to be applied in cloud solutions. Section 4.3.3 “Technical Foundation” of the same document identifies the concrete design constraints to be taken into account. Combined, these two sets define the design scope for the module that is elaborated further in this section.

**Data maintenance servicing:** The lifecycle and availability of operational and configuration data used by DPD will be independent of the lifecycle and availability of the DPD system itself. This isolation allows for improved availability of the services processing the data and helps to achieve higher reliability. Therefore, the operational data used by DPD will be externalized and maintained by the Cloud Storage service. A master copy of the module configuration will also be managed by the Cloud Storage and fetched upon initialization to setup the system for use.

#### Statelessness and autonomous processes

The DPD module will feature stateless service design with no conversational state between its operations. Session data and user context will be sent with each request

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>84</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



to an operation by the service client (i.e. no sticky sessions), and will be managed by the Messaging service and the cloud storage (as temporary session data storage) as appropriate. This design decision allows a request to be served by any instance of DPD and therefore requests from the same client to be served by different DPD instances. In effect, that will lead to transparent failover for clients in case of downtime for a service node and load balancing.

### Operational reliability

The processing algorithms in DPD will be designed to survive system failures and downtimes reliably without compromising operational data consistency. The module design will rely on the delayed messaging functionality provided by the Messaging service to fetch messages left pending due to a node reboot. The result will be a reduced number of not-served messages due to unavailability. With this design decision even a single service node can catch up the messages to be processed over time and with the only visible effect of delayed responses. This approach combined with a system designed for horizontal scaling would handle such situations faster and possibly even completely transparently, without visible delays, to the clients, hence the next decision.

### Horizontal scalability

Horizontal scalability is the capability of the virtual infrastructure to provide additional nodes as required (a.k.a scaling out<sup>5</sup>). Advanced virtual infrastructures can dynamically scale out to address increased consumption and scale in (the reverse operation) when the consumption decreases. It is essential to ensure that the application deployed on such virtual infrastructure can adapt to such environment changes. To do that, the module will be designed for parallelism internally and externally (between the module service nodes). Internal parallelism will account for thread safety and external parallelism will take care of the concurrent processing of messages to the module. DPD will realize the first through well-known techniques in that domain and the latter by subscribing the service nodes to the messaging infrastructure and using a dedicated service endpoint node for realizing a load balancing and system control mechanism.

---

<sup>5</sup> [http://en.wikipedia.org/wiki/Scalability#Scale\\_horizontally\\_.28scale\\_out.29](http://en.wikipedia.org/wiki/Scalability#Scale_horizontally_.28scale_out.29)

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>85</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### Efficient asynchronous communication

In potentially long running transactions, the component's communication strategy will be to utilize a publish-subscribe pattern and push back response, once ready (as opposed to alternatives for constant asynchronous query for ready state). This will significantly reduce wasted network traffic and will increase the overall performance and responsiveness of the system. The Messaging service will provide the required infrastructure to realize this and the correlation mechanism between request/response messages detached in asynchronous communication. Further, DPD design will feature an integration layer with the messaging service in order to support this design decision.

### Lightweight service APIs

In addition to the message-oriented communication strategy that the DPD component will employ, it will also implement service interfaces to offer direct communication through lightweight REST Web APIs for traditional pull communication. The API will provide the necessary protocol to browse and manage the data resources managed by DPD building upon the standard HTTP. This will ensure scalable service-oriented system architecture of autonomous, distributed service nodes and also provide for other options for integration with third-party applications, that might re-purpose the information supplied by ADVENTURE: for example for Business Intelligence operations and other statistics.

### Delegated access control

The component will communicate possibly private resources between the user and the data store and this requires secure communication channels which will be realized by the Message Routing component. DPD will provide service to authenticated users and agents and will delegate access control lists maintenance to the Cloud Storage component, providing the user context, according to the authorization policy in the system.

### Minimal data model

DPD is designed to be as content-agnostic as possible, and to be customizable to for use in different domains and with different levels of complexity, as required by its adopters. Therefore the module's design will define a minimal data model and will rely on technologies and techniques that enable reuse and integration of the abundance of already available classification systems, taxonomies, ontologies and

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>86</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

knowledge bases that organizations are already using for description. A key element in realizing this strategy is the elaboration of a mechanism to reliably associate the key elements of the DPD domain model to external definitions.

### **Efficient search interface**

The open information space defined above, needs to have a coherent data representation and enable querying in a standardized language with features that support efficient data retrieval and visualization (like limiting of the results, e.g. by paging, ordering, etc.).

Since the information space is open and can grow considerably with the popularization of an ADVENTURE deployment and addition of many new partners, the primary focus of the search mechanism will be efficiency and end-user satisfaction rather than completeness. However, the module will be also designed to allow for long running queries upon end-user consent.

### **Information space consistency**

Another key design decision is that the system administrators are going to be responsible for managing the information space intact, i.e. ensure no syntactic or semantic conflicts and failures or versioning issues, resolved ambiguities, etc. The DPD will not directly offer tools or support to ensure that.

DPD will ensure data models completeness and correctness to the extent that the partner profile models generated with its tools are going to be coherent with the information space syntactically and semantically. The data correctness itself is solely responsibility of the using party.

It is responsibility of DPD to ensure that the common information space is consistent, not polluted and rendered useless due to insertion, update or deletion of inconsistent data. DPD tools will only allow publication of data that is compliant with the minimal meaningful data model required to integrate a partner profile in the information space and successfully find it. Failures in the data integrity upon insertion will be transparently handled as part of this mechanism and reported back to responsible. That is, a partial data model can be introduced but is considered a working draft and therefore not published, until all minimal required data is available.

The data forming the information space is persisted in the Cloud Storage and made available for queries via its Semantic Bucket type. Links to binary files (such as

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>87</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

attachments) are also part of the DPD partner data module and DPD will manage links to them provided by the integration interface for content management. In order to ensure consistency DPD will hook to listen for important events in the lifecycle of attachments, especially deletion, and react accordingly to update the respective data models.

### Internal componentization

The DPD system is broken down into a hierarchical loosely-coupled subcomponent structure with well-defined interfaces and communication protocols. This is essential design method to promote replaceability of sub-parts of the system without major disruptions (and ultimately adoption in wider range of environments).

## 3.3.2 Technology Comparison and Selection

This subsection outlines technology selection criteria and compares existing technologies – potential candidates for the realization of Data Provisioning and Discovery component. Within the subsection the areas that need to be implemented or improved to fully meet ADVENTURE specific requirements are also identified and presented. The selected technologies will be used as a base for the development phase in the realization of the technical design of the DPD component.

### 3.3.2.1 Selection Criteria

The selection criteria for DPD technologies were defined in *D3.2 Functional Specification* in section 5.4.6 on page 113. The description of the generic parameters can be found in the *D3.2 Functional Specification* in section 5 on page 28. The description of the DPD specific parameters is listed below:

*Table 7 – Technology selection criteria for Data Provisioning and Discovery*

Name	Description
Standardized query language	Required to provide portability across data management systems that will store the data, without disruptive changes to the software itself. Examples are SQL, SPARQL, XPath, OData query language.
Support for complex queries	Queries might become very complex, especially when based on dynamically selected data, for example to hit the best service matches. Hence, a support for complex queries is

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>88</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

	needed. Support for long data lists pagination is also a plus.
Semantic, semi-structured and structured data models.	These are the types of data models that are needed for managing DPD entities.
Non-disruptively extensible data model	The data model technology for describing the DPD managed entities has to be extensible to new domains, classification systems, taxonomies, etc. This needs to be realized seamlessly without disrupting the normal system operations and without any changes in the system. For example if a classification for CO2 footprint is added, it has to become readily available for annotations without further changes required. The 'non-disruption' here also means that the system interfaces (service/UI) will not be modified as well. A promising technology that suits those needs is for example RDF.
Annotations support	One of the key requirements and use cases in DPD is to add metadata to system objects that will be then stored, managed and queried.
Reasoning support	Reasoning is a powerful feature in the semantic management systems that adds value to the existing data, by providing implicitly encoded facts in it and thus further expanding the knowledge base to unanticipated limits. Since it also comes at a performance cost, the tradeoff has to be evaluated and is thus identified as "a nice to have".
Lightweight service standards compliance	As any other ADVENTURE module, DPD will comply with SOA principles. It is also unique with the data it supports and the specific opportunities for a 'long tail' based on that. Lightweight services are getting significant momentum and support for them would help foster adoption better. Further, the operational requirements in ADVENTURE can be successfully implemented using lightweight service approach like REST. Standards and best practices compliance is also an important aspect from both SOA and adoption perspective.
Cloud readiness	DPD is provided as Software as a Service (SaaS) and the software supporting it must be also cloud ready.
Remote data storage service access	The data will be externalized to cloud storage and this means that data access protocols in DPD need to support remote management (e.g. SPARQL endpoints, OData and REST APIs for example).
Business Intelligence capabilities(analytics, reporting, data mining)	The key goal of DPD is to provide a browsable and searchable repository for members' assets descriptions. This data will be used both for operations management in adventure and also for Business Intelligence operations.

Usability	Since DPD also features a UI, to facilitate the users to provide ADVENTURE, usability is another important requirement. Usability will be measured in terms of how easy it is to start from generic information and drill down to fine-grained details. The best results will be produced by software that can provide linked data entities that provide a “browse-the-Web”-like experience and proactively recommend useful paths and data for further exploration. Worst results would be to build sophisticated queries manually at each step.
LinkedData principles support	The linked data principles are related to the usability as defined above and help to greatly improve the browsing experience for human users and also the data management for agents. Compliance here will be provided only to the extent needed for adding seamlessly additional information from external sources instead of cementing the boundary. Hence the 'nice to have' weight.

### 3.3.2.2 Possible Technologies

Organizations' profile management, service capabilities descriptions and other attributes of the DPD domain model are features that can be found in different software applications and domains. Historically, there have been standardizations efforts like OMG Trading Object Service, OMG Party Management Facility and others. However, most common limitations are that these are software components that are mainly tailored to specific domains (i.e. barely general purpose mechanisms), far too often tightly integrated in the software systems, for which they are built, or simply outdated, all of which makes their utilization/customization for ADVENTURE hard or even impossible. Another major concern is that they do not fully cover the whole domain model required by DPD. It is virtually impossible to elicit some technologies, which could be used directly or easily by extension and in compliance with the functional and technical properties of the DPD component.

Therefore the focus in the state of the art study in this case is to find out what models, modeling technologies and tools were used by different relevant research projects in the area of business collaborations, in order to address certain aspects applicable for the DPD domain model (even partially). On the other hand, other technologies that might be used for description and management of DPD are also investigated and analyzed. Collectively they constitute the baseline, on which the technology selection for further implementation will be made.

The projects on which the study focused are:

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>90</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Net-Challenge:** “Innovative networks of SMEs for complex products manufacturing”, is an EU research project in the area of collaborative business networks, supported by the 7th FP of the EC. Specific objectives include the design and development of an integrated framework, composed of a methodology; reference business processes and IT tools, to support SMEs create and manage such networks. Net-Challenge features a collaboration portal (a customized implementation of the Open Source Liferay portal), and a portal plugin for partner profile management that allows simple manual search operations, as well as template support. Web site: <http://www.netchallenge.org/>.

**SEAMLESS:** Seamless is an EU FP6 research project which vision is towards a Web-based Single Electronic European Market (SEEM) where companies can dynamically collaborate without cultural and technological constraints. SEEM allows an objective comparison of profiles and offers of company of any size and location, and thus opens the eBusiness space to the many small companies. The collaboration framework developed within the project is intended to establish the rules for companies and users to access the SEEM, enter and update profiles and offers, search for possible partners, contact them, negotiate, exchange information and documents. The Seamless framework reconciles mismatches between potential partners on semantic level. It features Global ontology to describe a partner profile in a common model and a UI that is built dynamically based on it. Web site: <http://www.seamless-eu.org/>.

**OMG’s Trading Object Service (TRADE), Party Management Facility (PMF) –** This is a set of standard specifications from Object Management Group (OMG) that are dedicated to the definition of a model and abstract API for relationships management between collaborating parties and the targets of this collaborations.

**OMG Trading Object Service:** This OMG specification facilitates the offering and the discovery of instances of services of particular types. A trader is an object that supports the trading object service in a distributed environment. It can be viewed as an object through which other objects can advertise their capabilities (“export”) and match their needs against advertised capabilities (“import”). Export and import facilitate dynamic discovery of, and late binding to, services. Web site: <http://www.omg.org/spec/TRADE/1.0/>.

**OMG Party Management Facility:** This OMG specification defines a set of generic and extensible service interfaces for managing partner information (among other

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>91</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

objectives) and party relationships based on role-aware compositions. Web site: <http://www.omg.org/spec/PARTY/1.0/>.

**NEFFICS:** This is an EU research project in the area of Internet of Things and Enterprise Environments, supported by the 7th FP of the EC. It defines a Business Model Framework that characterizes the business of an organization. The framework proposes seven building blocks that represent the core components of a business model: value proposition, target customer, value chain, competencies, partner network, relations, and profit formula. Web site: <http://neffics.eu>.

**GoodRelations:** Is a lightweight ontology for annotating offerings and other aspects of e-commerce on the Web. GoodRelations is officially supported by both Google and Yahoo. It provides a standard vocabulary for expressing offerings of products and services and for supporting all commercial and functional details of e-commerce scenarios, e.g. eligible countries, payment and delivery options, quantity discounts, opening hours, etc. Web site: <http://www.heppnetz.de/ontologies/goodrelations/v1.html>.

**USDL:** The Unified Service Description Language (USDL) is a specification proposed by the W3C USDL Incubator Group with conclusion for further development as a W3C standard recommendation. It is proposed as a modular “master data model for services” to describe various types of services ranging from professional to electronic services. It aims at a holistic service description putting a special focus on business aspects such as ownership and provisioning, release stages in a service network, composition and bundling, pricing, location and legal aspects among others, in addition to technical aspects. As part of the incubator group activities, the Linked Data (<http://www.linkeddata.org>) principles have been taken into account into the language design and currently parts of the specification are published as Linked data on the Internet under the name Linked USDL (<http://linked-usdl.org/>), directly linking it to other published Linked Data (e.g. Good Relations, MSM and FOAF to name a few). Web site: <http://www.w3.org/2005/Incubator/usdl/>

**Schema.org:** Schema.org provides a collection of shared vocabularies that Webmasters can use to mark up their pages in ways that can be understood by the major search engines: Google, Microsoft, Yandex and Yahoo!. It is an effort regularly synchronized with W3C community via the W3C group Web Schemas (<http://www.w3.org/wiki/WebSchemas>). Schema.org provides the vocabulary for several interrelated key concepts that are part of the DPD domain model – organization, product and person with a good deal of precision of the description. The

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>92</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



support for the service concept however is very limited to a few predefined enumerations. There's a parallel activity called Schema.RDFS.org that provides different representation formats over the core schema.org vocabulary, including RDFS, which allows for reference from other ontologies and extension to fill the gaps. It is schema.rdfs.org that shall be referred to from this point on. Web site: <http://schema.org>

The following table (Table 8), reduced to reasonably meaningful properties as per the change of focus outlined above, shows a comparison of the potential modeling technologies as well as their rating:

*Table 8 - Technology selection criteria and comparison of technologies for Data Provisioning and Discovery*

Parameter	Importance (- - - +/- +++)	Net-Challenge	Seamless	OMG TRADE & PMF	NEFFICS	Schema.org	Linked USDL	Good Relations
<b>Generic Parameters</b>								
Maturity & Stability	+++	+/-	+/-	++	+/-	+	++	++
Regularly Updated	+	-	-	-	-	++	+	+
Technical Up-to-Dateness / Appeal	+	+	+	-	+	++	++	++
Open Source	+/-	NO	YES	YES	YES	YES	YES	YES
Non-Infecting	++	NO	YES	YES	YES	YES	YES	YES
Code-Quality	+	+	+	+	+	+	+	+
Extensibility	+	+	++	+	++	++	+++	+++
Community	+	-	-	-	-	+	+	+
EU project origin	+/-	YES	YES	NO	YES	NO	YES	YES
Open Standards	+	+	++	++	++	++	+++	+++

Compliance								
Interoperability	++	+	++	+	++	++	++	++
<b>Specific Parameters</b>								
Standardized query language	+++	-	++	-	+	++	++	++
Support for complex queries	++	-	++	-	+	++	++	++
Semantic, semi-structured and structured data models.	+	+/-	+/-	+/-	+	+/-	+/-	+/-
Non-disruptively extensible data model	++	-	++	+	-	++	++	++
Reasoning support	++	-	++	-	-	++	++	++

Apart from the above described and analyzed projects, relevant external classification systems will be also exploited, such as:

- NACE Code:** Is a pan-European classification system which groups organizations according to their business activities. It assigns a unique 5 or 6 digit code to each industry sector. Its RDFS representation is available online: <http://ec.europa.eu/eurostat/ramon/ontologies/nace.rdf> . NACE is tightly related to the ISIC v4 classification (the International Standard Industrial Classification of All Economic Activities, United Nations Statistics Division, which is a product scheme classification used by the United Nations to create statistics) by references to its codes. The RDFS (LinkedData) assets of ISIC v4 can be found online: <http://thedatahub.org/dataset/isic-v4>.
- Global Product Classification (GPC):** Is a rules-based, four-tier classification system for grouping products. It is part of the GS1 system and a key enabler for the Global Data Synchronization Network (GDSN) and category management. GS1 is an international not-for-profit association, dedicated to the design and implementation of global standards and solutions to improve the efficiency and visibility of supply and demand chains globally and across sectors. The GS1 system of standards is the most widely used supply chain standards system.

GS1 launched an initiative to align GPC with the United Nations Standard Products and Services Code (UNSPSC) (see below).

- **UNSPSC (United Nations Standard Products and Services Code):** Is a global, multi-sector classification system supporting primarily spend analysis and procurement. Its RDFS representation is available online: <http://www.cs.vu.nl/~mcaklein/unspsc/unspsc84-title.rdfs>
- **The Google Product Taxonomy:** Is a tree of categories that describe product families. At the highest level, it is arranged into a few verticals, which refer to broad categories of products (Electronics, Home & Garden, etc.). The “children” of these broad categories are more specific product families (Electronics > Audio) or specific products (Electronics > Audio > Audio Players & Recorders >). The taxonomy is available as excel spreadsheet or plain text with legacy format: <http://support.google.com/merchants/bin/answer.py?hl=en-AU&answer=160081>

The applicability of the data model employed by the reviewed technologies to the DPD domain model is discussed below with relation to partner company profile, services and products:

*Table 9 – Applicability of the data model employed by the reviewed technologies to the DPD domain model*

Partner organizational profile			
<b>Seamless</b>	Seamless company profile model is good (comprehensive) and extensible but do it yourself approach, exploiting the CORE ontology defined within the project scope. It is also more focused on the business and collaboration aspect, leaving the technical details aside. Currently, it is no longer evolving and there is known ontology that have the potential for bigger impact.		
<b>Net-Challenge</b>	Net-Challenge company profile is designed as general purpose model but is somewhat restricted by the choice of supporting technologies. The extensibility is only in terms of simple key-value pairs. Complex queries and scaling of the data model would be challenging. No standardized vocabulary or ontology is in use.		
<b>NACE, UNSPSC</b>	NACE (implying also ISICv4) and UNSPSC have the goal to classify economic activities. They cannot serve as a comprehensive profile model but rather complement other models by reference. They could be used to classify an organization in a certain economic activities domain.		
<b>Linked USDL</b>	Linked USDL spans over multiple aspects of partner profile. It is		
D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: 95 / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

	quite extensible (via LinkedData principles) by design and contains the a minimal useful set of concepts and properties. It provides a reference from its service concept to the FOAF ontology concept Agent, to include the collaboration partner details. The Agent concept is a bit too simplified and needs to be changed for a more useful concept.
<b>Schema.org</b>	The RDFS incarnation of Schema.org provides an ontology for organizations and their products and services that covers a lot of interesting aspects of a partner profile. It's incomplete for the DPD purposes and mainly focuses on the business description. However it's based on linked data principles and an extensible RDF technology, which provide for the minimum required to extend it as appropriate to capture all relevant details.
<b>GoodRelations</b>	Covers most of the key aspects relevant to business partner profiles via its BusinessEntity concept. It proclaims to be compatible with the concept Organization from schema.org and has similar semantics.
<b>Services</b>	
<b>Linked USDL core</b>	Describes business, technical and operational aspects as required and is also easily extensible to cover more aspects. The services description in Linked USDL Core is very comprehensive and captures details like interaction protocols, preconditions and post-conditions that are generic enough to span over both technical and non-technical services. By far this is the most comprehensive of the reviewed service models.
<b>Seamless</b>	Provides support for describing business services through CORE ontology developed. In particular, products and services can be coded to express the company offer (to be searched as supplier) and the company demand (to be searched as customer). There is no technical support for SaaS, for example service invocation endpoints.
<b>GoodRelations</b>	Describes the different aspects of services, including all commercial and functional details. It provides an extensible model for Vertical Industries, but is rather limited on describing technical details. Linked USDL Core ontology refers to GoodRelations to complement the business aspect of a service description,
<b>Products</b>	
<b>Seamless</b>	As already said, the description of products is based on the CORE ontology, but the notion of the service in terms of SaaS is not supported.
<b>GPC</b>	GPC is a set of common categories that provides a common language to buyers and sellers for grouping products in the same manner globally. GPC is mandatory for global data synchronization. GPC works by defining a hierarchy starting by industry sector or segments. In GPC, a brick defines categories of similar products. A bricks can be defined by brick attributes. Still this mechanism is not

	very well supported.
<b>Google product taxonomy</b>	Google product taxonomy provides a classification for products but still on a very high level and limited in this sense. The extensibility is also not well supported.
<b>Linked USDL</b>	Has a built-in, generic concept for a Resource, which can be further extended and described as appropriate. Further, products in ADVENTURE described in terms of services (SaaS) can be described with Linked USDL core ontology by design.
<b>GoodRelations</b>	Describes the different aspects of products, including all commercial and functional details. It provides an extensible model for Vertical Industries, but is rather limited on describing technical details. It can be successfully used to complement the description of a resource defined by Linked USDL.

### 3.3.2.3 Conclusion

From the analysis performed above it became obvious that Linked USDL, GoodRelations and Schema.org complement each other nicely to provide a good coverage for most of the DPD domain model concepts and are supported by the technologies that will ensure extensibility and openness, all of which backed up by live communities and large industrial and organizational entities. Since Linked USDL provides the most comprehensive model of the three, it will serve as basis and will be complemented by the rest. The core model shall be further enriched by references to external established classifications like the reviewed NACE or UNSPC. However, it also became apparent that none of the reviewed technologies covers completely the DPD domain model and requirements. For example organization's processes are entirely out of the picture and there's also no applicable ontology for manufacturing task assignments properties like carbon footprint, task durations, etc. Therefore where Linked USDL, GoodRelations and Schema.org do not cover the ADVENTURE domain model and no other external and compliant model is available to fill the gap, own (ADVENTURE) vocabularies/concepts will be contributed accordingly.

### 3.3.2.4 Technology Selection

Despite that the state-of-the-art tools and frameworks that might be relevant to DPD feature capabilities like profile management and service modeling are good at supporting some of these, they barely cover collectively more than a few of the required features. In addition, there is no common standard and hardly even common guideline to follow, and basically there's no general purpose software that fits well

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>97</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

enough to the requirements of DPD. Respectively more implementation effort will be required, but this also presents an opportunity to pioneer in the field and contribute to the EU research with useful developments that fill a clear gap. The next sections will analyze the state-of-the-art and provide decisions on technologies and their use.

#### 3.3.2.4.1 Domain model and data modeling technologies

The domain model supported by DPD and described in the functional specification is very close to the domain models elaborated in focused activities like SEAMLESS and NEFFICS projects. This confirms its validity as generic model and requires good extensibility options from the modeling technologies that will be employed in order to ensure effective and efficient tailoring to specific domains and needs.

The state-of-the-art with regards to such requirements is usually favoring XML, RDF and JSON as options. The DPD module will rely on RDF as data modeling technology due to the fact that most of the classification systems, taxonomies and ontologies that would be of interest to ADVENTURE (NACE, UNSPCS, e@class, etc.) are or can be provided as RDF and will naturally form a common information space with the core DPD domain model.

**RDF** is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a “triple”). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications. This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations.

Another plus of RDF is its inherent extensibility and content agnostic nature. With a software that relies on its core data model and not on concrete content, changes in the schema supported by DPD would be completely transparent, which would provide for less disruption both during the development and also in the deployment/adoption phases. Further RDF supports a standard and powerful

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>98</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

query language – SPARQL with features and capabilities that resemble the well-known SQL but suited to the RDF graph data model.

DPD will expose a SPARQL endpoint to support for direct interaction with the RDF content. This is useful for supporting complex queries that would be hard to translate using the OData query language in more sophisticated operations (like recommendation of services based on some context) and for debugging purposes.

The minimal domain model supported by DPD is going to build upon the effort that has been carried out in the domain of software and business services with the Universal Service Description Language (USDL) and its LinkedData incarnation – Linked Data USDL. The latter successfully combines established external ontologies (e.g. GoodRelations) to enrich the core data model of USDL and also presents a showcase of how to further extend it for concrete needs.

Together with Linked USDL, DPD will take advantage of the lessons learned by other relevant research projects for describing business partners and will seek to combine these where appropriate.

As already described above, the knowledge domain delivered out-of-the-box with ADVENTURE will also feature several established classifications (none of which normative and mandatory) and their integration will show case how to further extend the base with other classifications of choice.

The knowledge base will be also featuring Dublin core as publishing model to be used for adding authoring metadata to process models managed within the system.

#### 3.3.2.4.2 Service API and content formats

One of the main goals of the DPD module is to provide a repository that describes businesses in terms of collaboration opportunities. In order to achieve maximum effect it will be natural to make it global e.g. integrated into the World Wide Web. Therefore it will favor HTTP and HTTP-derived protocols for interaction with its service API.

Further, DPD design will consider SPARQL and OData as service protocols and expose service endpoints on which clients can interact with backend data according to their respective rules.

**SPARQL** - is a standardized by W3C query language and protocol for RDF content. As a protocol it defines means for conveying SPARQL queries compliant with

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>99</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

SPARQL 1.1. Query language specification and updates compliant with the SPARQL 1.1 Update specification to a SPARQL processing service and returning the results via HTTP to the entity that requested them. As a language it defines the vocabulary, constructs and language rules to request information from a RDF graph.

A separate document defines the SPARQL 1.1 Graph Store HTTP Protocol which describes the use of HTTP operations for the purpose of managing a collection of graphs in the REST architectural style using standard HTTP methods. It provides a simple and well known API, but necessarily restricted in its operations due to the limited set of methods in the HTTP protocol. In contrast, SPARQL 1.1 protocol permits multiple modifications in a single operation, and can use complex SPARQL queries for selecting or constructing data to be inserted, or choosing data to be deleted. Also, the use of an update language facilitates operations over proprietary APIs and connections that may not involve HTTP. SPARQL supports responses payload formatting as JSON or XML with specified schema and content negotiation for the representation format as per the standard HTTP rules. The results content can be formatted for better consumption by slicing it into chunks, using the standard language vocabulary. This facilitates applications that need to present a UI that is suitable for browsing in small portions.

SPARQL service can also advertise its capabilities by a standardized metadata format and thus supports automated discovery and invocation for complex compositions in the 'long tail' of the DPD uses.

These properties make it especially suitable for the DPD content formats and requirements for seamless integration into the Web and powerful query capabilities.

The SPARQL service endpoint of the DPD service will be a generic, machine interpretable interface to the managed metadata. Using the SPARQL 1.1 Query protocol gives opportunities to build very complex queries and derive information beyond the obvious and consequently construct highly useful decision-support, autonomous applications. In the case of the DPD use cases this will constitute the backbone of important features like the recommendations mechanism for finding suitable partners for an activity.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>100</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



The specification of the SPARQL query language can be found at: <http://www.w3.org/TR/rdf-sparql-query/> and the specification of the SPARQL protocol can be found at: <http://www.w3.org/TR/sparql11-protocol>.

**OData** - OData is an open industry standard for presenting enterprise data in the Web, building upon the established Atom publishing protocol and format.

OData naturally supports REST-full services and provides a standard query language and options for data representations that natively fit into the HTTP concepts.

With regards to content formats, OData supports JSON and XML representations. Further, the suitable representation can be negotiated between the client and the server via the standard HTTP content negotiation mechanisms. In effect, this leads to dynamic allocation of representation format depending on the client preferences, where a client can be as generic as a standard Web browser.

OData content format builds upon Atom content format rules and enhances them with rules useful to present EAV(Entity-Attribute-Value) structured collections of entities.

Likewise SPARQL OData also advertises its capabilities via its metadata interfaces and also supports automated discovery and invocation for complex compositions in the 'long tail' of the DPD uses.

The OData specification can be found at: <http://www.odata.org/> and the OData OASIS standardization effort can be tracked via its Technical Committee at: [https://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=odata](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=odata)

Both SPARQL 1.1 and OData v3 protocols have similar capabilities and can be utilized for the purpose of the DPD use cases. On the other hand, the different technical approach and the different data model abstraction (SPARQL explicitly supports RDF while OData format is XML specific format) present strengths in different aspect.

SPARQL is the more powerful protocol to derive holistic conclusions based on the whole metadata set managed by DPD, while effectively doing Business Intelligence-like operations. OData, on its turn, is the most natural fit for a lightweight REST data service endpoint, well suited for exploring sequentially and presenting lower entry

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>101</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

level for consuming applications. In this way both have their uses in DPD - SPARQL for automated and highly effective recommendations and optimizations via a SPARQL Query endpoint and OData for UI applications suitable for browsing and exploring and managing the metadata as mandated by the ADVENTURE specific requirements via a OData service endpoint.

Instead of choosing between the protocols DPD will rather use them in the context outlined above, where each has more distinctive strength. The availability of both will reduce risks in case either of them turns out not to be feasible and will particularly strongly support the further adoption and reuse of the DPD module even beyond the project and thus is highly recommended.

#### 3.3.2.4.3 User interface

The human-computer interaction will built upon the lessons learned from the Net-Challenge project and further enhance with dynamisms and better utilization of client-side resources. The UIs (UI) for the data browser and the partner provisioning forms are going to be delivered as standalone software, remotely integrated in the dashboard. Their UI shall comply with the dashboard look-and-feel and will employ popular, compatible client-side UI frameworks. The primary choice is jQuery (see below) UI due to its massive adoption, simplicity, feature-rich UI set and client-side architectural positioning. Using such a client-side Web UI technology provides an excellent opportunity to reduce unnecessary traffic and load by using asynchronous, AJAX communication and consuming the generic REST API as any other client, without special tailoring requirements. Building upon this, the partner provisioning forms will employ Microdata (see below) in order to build the model employed by the service APIs entirely at the client with a generic DOM API. As opposed to the alternative to send HTML forms for processing to the server that will do the same before contacting the service API, employing Microdata model keeps this boilerplate code, UI-specific operation on the client side. Another advantage is that Microdata allows for abstracting from the HTML presentation structure so changes in it would not affect the data model. As a consequence, the mechanism would be reusable across different UI implementations or layout changes. Further, it has a standard mapping to JSON structured format and there are JavaScript libraries implementing it to parse an HTML and create a JSON model right out of it and make it readily available for further processing or direct consumption.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>102</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**jQuery** - is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML. Used by over 55% of the 10,000 most visited Websites, jQuery is the most popular JavaScript library in use today.

jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. The modular approach to the jQuery library allows the creation of powerful dynamic Web pages and Web applications. There is a substantial plug-ins collection available now.

**Microdata** - is a WHATWG (Web Hypertext Application Technology Working Group) HTML5 specification used to nest semantics within existing content on Web pages. Search engines, Web crawlers, browsers or other applications can extract and process Microdata from a Web page and use it to provide a richer experience for users. Microdata use a supporting vocabulary to describe an item and name-value pairs to assign values to its properties. Microdata helps client applications to better understand what information is contained in a Web page. Microdata is an attempt to provide a simpler way of annotating HTML elements with machine-readable tags and a more standardized unambiguous parsing model than the similar approaches of using RDFa and Microformats. At the same time it is compatible with numerous other data formats including RDF and JSON.

Search engines including Bing, Google, Yahoo! and Yandex rely on the Microdata markup to improve the display of search results, making it easier for people to find the right Web pages. A shared markup vocabulary makes easier for Webmasters to decide on a markup schema and get the maximum benefit for their efforts. Taking this into account, search engines have come together to provide a shared protocol (sitemaps.org) and collection of schemas that Webmasters can use. Schema.org provides such a collection of shared vocabularies.

In the above context, building upon Microdata technology creates future opportunities for using the indexing and discovery capabilities of the largest search engines to find ADVENTURE partners.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>103</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### 3.3.2.5 Missing Elements and Implementation Needs

The technologies of choice described above provide a basis for modeling the DPD domain model and the framework for realizing its functionality. In summary the implementation needs are the following:

Tools for browsing and searching the DPD domain data via interaction with human users and programmatic APIs.

Tools for data provisioning and management handling holistically the profile in each particular context – business partner details, services, products and processes.

Tools for attachments management.

Annotations support tools handling the interaction with the user and the metadata knowledge base.

Tools for configuration of the module integrated with the Dashboard, targeting customization operations by administrators and consumption at runtime by the module's components.

Integration with the other ADVENTURE modules offering data services.

Scalable architecture supporting tools for realizing cloud deployments.

The component breakdown explored in the next section elaborates further these needs in terms of technical components.

## 3.3.3 Technical Component Specification

### 3.3.3.1 Component Structure

The component breakdown of the DPD module is presented in Figure 20. For more clarity, the components are spit into several layers and in addition external client components are shown.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>104</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

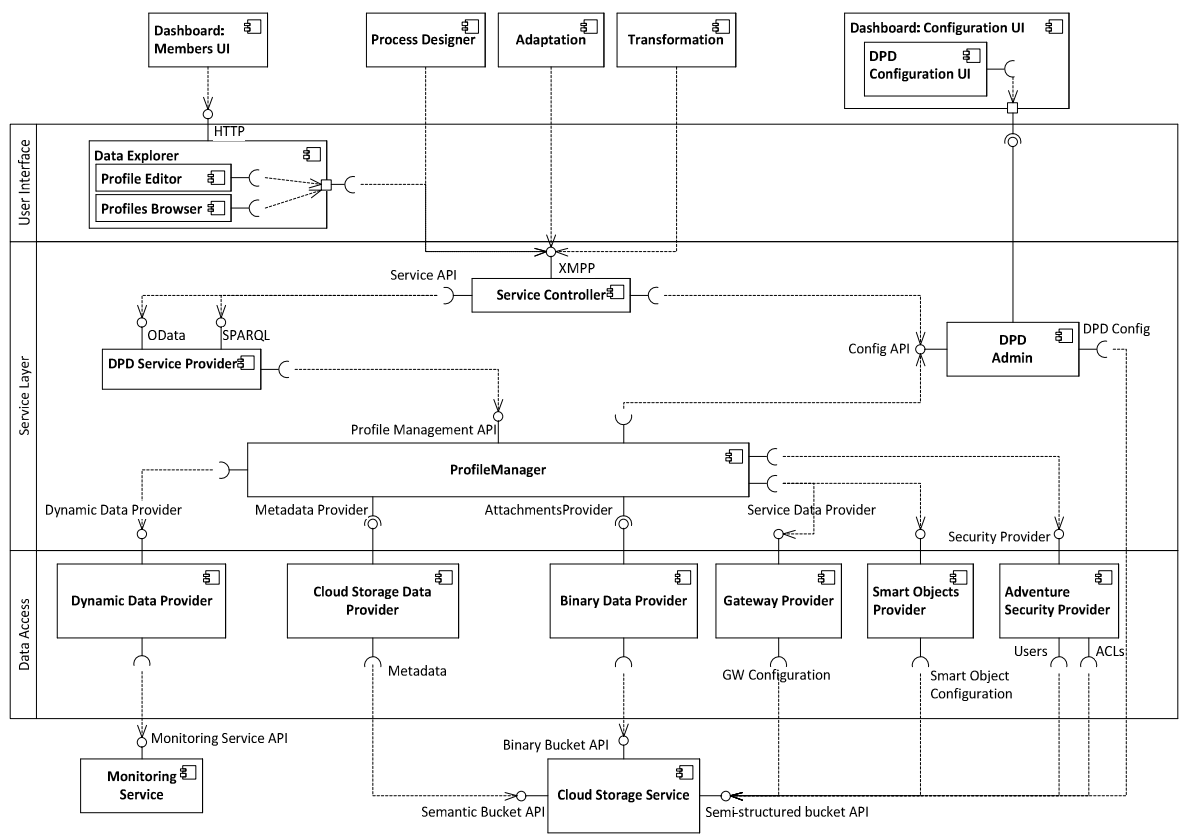


Figure 20 - DPD component structure

3.3.3.2 User Interface Layer

DPD UI layer is represented by **Data Explorer** component that consists of two UI components – Profiles Browser and Profile Editor.

Data Explorer

The Data Explorer is the component that realizes the scenarios for data provisioning and discovery in a browsable repository for ADVENTURE members. It can be used by human users for the provisioning of new member’s information, exploring the potential collaboration opportunities or get to know the user base and do some business intelligence manually. The component is realized as a standalone Web application, hosted with the DPD service and loosely coupled with the Dashboard, which will integrate it as a remote site in its *Members* area.

The Data Explorer is further broken down into subcomponents realizing the concrete use cases in its context. The major ones are Profiles Browser and Profile Editor.

## Profiles Browser

In order to provide a browsing service to its users the Profiles Browser will consume the information served by the DPD service via the OData protocol.

In brief, the UI of the component will be designed for several scenarios:

**“Freestyle” browsing:** the user starts from a predefined search result listing all profiles a user can see with a predefined set of attributes exposed: organization name, date added, owner (Dashboard user), and business domain. The user is then presented with the options to scroll the list up and down, filter it further or jump to the details of a particular member (e.g. offered services, processes, etc.).

**Search:** the user invokes the simple search that looks up by keywords. In this case the application will return a preformatted result that gives the context of the hit (e.g. “processes” if a process model name was among the hits) and a link for displaying the found object in the corresponding profile. The other option is advanced search where the user can specify a query using the available concepts and attributes and common operators for them. The returned result set in this case will comprise all attributes that were in the search criteria formatted as table columns and can be customized for reduced or extended details.

**Profile details view:** the user has selected a concrete profile among search hits to review for details. The details view consists mostly of static data that constitutes the domain model of a DPD profile object and additionally dynamically fetched, monitoring data. The user will be able to easily jump between concepts and also tools using the details as a starting browsing point. This scenario is realized by a view mode of the Profile Editor component which is discussed next.

## Profile Editor

The purpose of the profile editor is to present member profile details and to provide the eligible users with the ability to manage a member profile in terms of editing or deleting the details. It's used during the initial provisioning phase for an ADVENTURE member or later when the information needs to be updated/changed/viewed. The component realizes two modes – view and edit. In view mode the component presents a read only interface for browsing details and in edit mode it uses a collection of appropriate editors for the details that can be changed.

The Profile Editor comprises a set of specific metadata editors that will be integrated like plug-in. The metadata editors are context sensitive and present only contextually

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>106</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

relevant metadata datasets for annotation. The contextually appropriate metadata is identified by the configuration of the DPD (not shown on the diagram for simplicity) where the administrator selects the right contexts where it should appear. Based on that, and the simple and more sophisticated annotation editors, the metadata is managed for the different concepts defined in the minimal domain model instance for a member. The data available through the Profile Editor can generally be split into two categories: static and manageable, and dynamic and informative. The static information does not change or changes rarely (for example organization legal details). The dynamic information is retrieved from different information sources and is not subject to change. It presents concepts like current operational status properties for a factory service or process assignments for a member.

The data fields relevant to the profile data model in the Profile Editor UI will be tagged with Microdata, referring to external vocabularies that make up the minimal domain model in ADVENTURE profile part of DPD. On save, it will extract this metadata in structured form and generate the concrete model instance as a OData formatted JSON request payload that will be sent to the DPD OData service for processing. In order to present the initial view and submit the changes the Profile Editor will use the OData endpoint presented by the DPD service as appropriate to receive a JSON-formatted result set.

### DPD Configuration UI

The DPD configuration UI is a DPD-specific UI application, integrated into the Dashboard Configuration area, which communicates with the DPD Admin subcomponent to present and modify the module's configuration.

#### 3.3.3.3 Service Layer

The service layer comprises the essential DPD components that are designed for use in ADVENTURE and will also allow reusability and exploitability beyond the project.

### Profile Manager

The 'heart' of DPD is the Profile Manager component. It realizes all mechanisms for aggregation of information from disperse data sources and formatting it according to security and privacy constraints into a coherent common model and providing it via a well-defined API. The component is also responsible for handling the consistency of

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>107</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

the data in a transactional manner, so that the metadata for a given member instance is always intact as a coherent whole.

As a central component, Profile Manager communicates with many if not most other components. It exposes a local interface API and uses the APIs of the DPD Admin for configuration of its behavior and environment:

the Cloud Storage Data Provider for metadata management in the remote persistent semantic (graph) store,

the Dynamic Data Provider for fetching dynamic operational information and process models (the latter is not shown on the diagram for simplicity),

the Gateway Data Provider and Smart Objects Provider for provisioning the member services for description, indexing and search,

the Security provider for realizing and consulting for access control constraints and user identification,

the Binary Data Provider for managing links to binary attachments.

All these dependencies are defined via service-provider interfaces that the corresponding components need to implement in order to provide service to the Profile manager component. The component realization will be stateless, in order to provide for processing of concurrent client requests.

### DPD Service Provider

The DPD Service Provider component is responsible for servicing the local API provided by the Profile Manager component. It presents a service-oriented interface for integration in different scenarios and plug in the ADVENTURE architecture. The component provides support for the two main service protocols of choice - OData and SPARQL. In concrete deployment scenarios this component is the endpoint of a service node (possibly among other peer ones) and the libraries that constitute the processing mechanism are the Profile Manager and its dependencies. This (one or more) endpoint(s) is also to be integrated into the system's messaging infrastructure as a message endpoint dedicated to the Service Controller component (see below).

### Service Controller

The Service Controller component facilitates the deployment of DPD in different scenarios and the integration with the other ADVENTURE modules via the central messaging service as defined in the overall architecture.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>108</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



The component is the single service endpoint with an XMPP interface for communication with other client components (e.g. Process Designer, Adaptation, Transformation, Members UI or any other that needs the DPD service). It will be implemented following the singleton pattern and will provide internal load balancing to delegate a received request to a suitable Service Provider OData/SPARQL interface realized by one of the potentially numerous, autonomous Service Provider deployments (XMPP endpoints) behind, which are subscribed for further processing and response. The response is then forwarded back to the requesting client.

### **DPD Admin**

The component is responsible for the administrative handling of the DPD module. It manages the configuration, its access by the module's subcomponents and modification by external system management applications like the dashboard configuration.

#### **3.3.3.4 Data Access Layer**

This layer contains the components, which are responsible for provisioning of different types of remote and disperse in nature and format data, coming from different sources.

### **Cloud Storage Data Provider**

The component is the client API for accessing the Cloud Storage metadata bucket. It implements a Metadata Provider API as a service provider contract between Profile Manager and the component implementation, which makes it easier to substitute with any other compliant with that interface on a later stage.

### **Gateway Provider**

The component is the client API for accessing a Gateway configuration. It implements a Service Data Provider API as a service provider contract between the component realization and Profile Manager, which makes it easier for substitutions in different deployments. The Gateway Provider will present a subscription functionality that can be used to listen for important change in the ADVENTURE environment. For example, Profile Manager will subscribe and will be notified when a partner service gateway has been removed. This will then result in updating all affected profiles to keep them consistent.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>109</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### Smart Object Provider

The component is the client API for accessing a Smart Object configuration. It implements a Service Data Provider API as a service provider contract between the Profile Manager and the Smart Object implementations. The Smart Object Provider will present a subscription functionality that can be used to listen for important change in the ADVENTURE environment. For example, Profile Manager will subscribe and will be notified when a Smart Object gateway has been removed. This will then result in updating all affected profiles to keep them consistent.

### Dynamic Data Provider

The component is the client API for accessing the Monitoring Service. It implements a Dynamic Data Provider API as a service provider contract between the Profile Manager and this component realization. The role of this component is to get dynamic and read-only information like the one managed in the real-time Monitoring service (i.e. stock levels and availability for a period related to a particular service and factory) and processes where a partner is was/is involved in particular activities. The component formats this information into a coherent model, compliant with the rest of the DPD domain model and provides it to its clients. Profile Manager will normally be the client that will request this information and then consolidate it with the rest of a profile details to return back to a client (e.g. Profile Editor). For better efficiency DPD may index some of this information and update it regularly or upon events. The component will therefore provide for realizing consistency between the data sources.

### Security Provider

The component is the client API for accessing the access control lists management functionality in cloud storage and the user identities registered in the system. It implements the functionality required by the Profile Manager component for realizing privacy and isolation when handling data (incl. metadata). The component implements an API that is the contract between the Profile Manager and any other implementation that could be plugged to play the same role in different scenarios or deployments. The Security Provider interface defines the minimal API required by the Profile Manager to realize privacy and isolation for the data managed in DPD.

### Binary Data Provider

The component is the client API for provisioning and managing links to binary attachments stored in the Cloud Storage Binary Bucket. It will realize a subscription

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>110</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

for changes mechanism that will react on events in the attachments store and trigger update in the DPD metadata. For example when a binary file is removed from the binary bucket, the Binary Data component will notify listeners for the change and provide the identification of the missing file event detail. The subscribers for that type of event (Profile Manager) will then remove the link from all profile models where it has been used and thus keep them consistent in real time.

To keep links format consistent, the component will wrap also the functionality on browsing links - it will be consumed via the OData service endpoint exposed by the DPD service. The component functionality is presented via a defined service API – Attachments Provider. This allows for implementations for different content management service to be plugged in DPD without major disruptions.

### 3.3.3.5 Internal Sequence Diagrams

This section will give an insight of the internal component communication sequence based on a list of example scenarios. The sequence diagrams presented below show an overview of the interactions between the components. They do not show every single detail of this interaction but the most important for ADVENTURE use cases and main scenarios. Further, components that have secondary role in terms of the use case (i.e. service providers, or infrastructure components like the Service Controller) are not shown for better readability and focus.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>111</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

• Provide new member profile

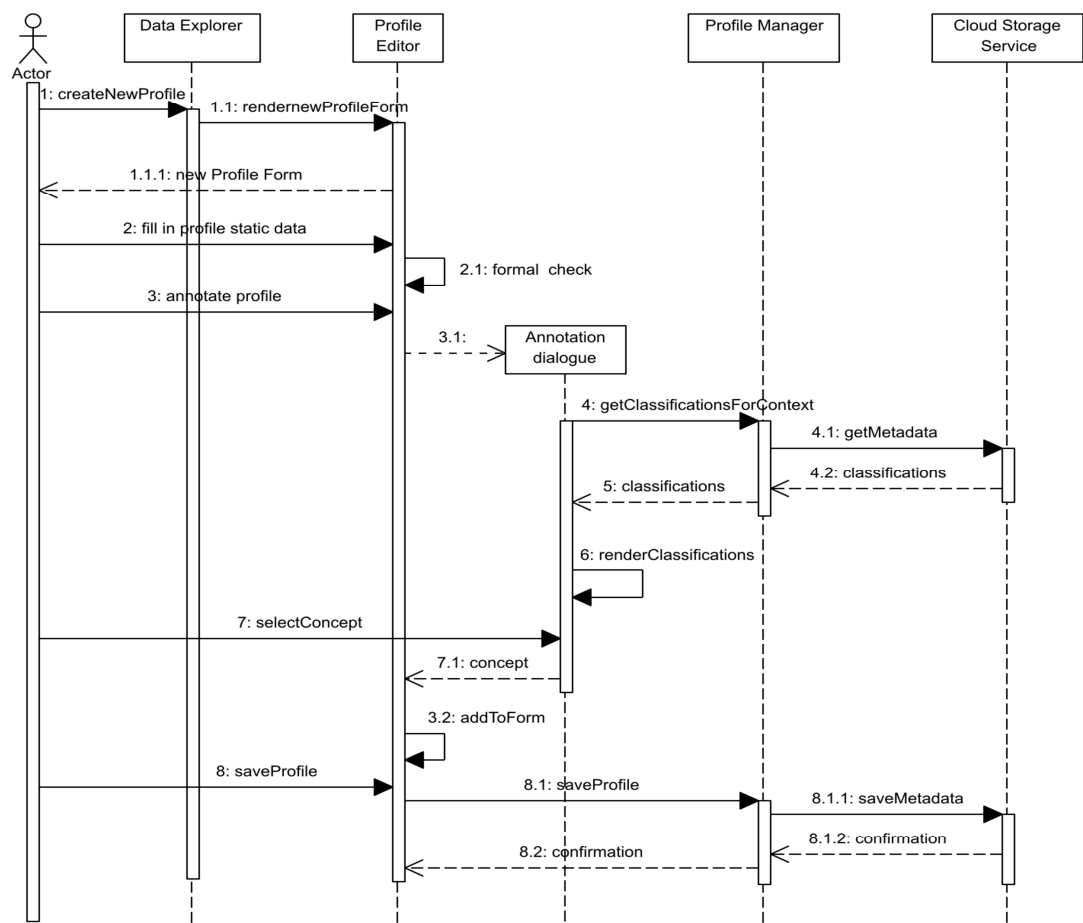


Figure 21 - Sequence diagram for Provide new member profile

In the described scenario, a user opens the Data Explorer Web UI application to start the process of provisioning of a new ADVENTURE member profile. The request for a new profile reaching the Data Explorer triggers the rendering of the Profile Editor UI for the end user. When the Profile Editor is rendered the user can start inserting data to the presented form fields. The Profile Editor performs a formal check to validate the input where possible. The user can also categorize the member organization by adding references to available classifications. To do that it sends an annotate profile message to the Profile Editor, which triggers the rendering of a new Annotation dialog for it. Before rendering the Annotation dialog checks with the Profile Manager to identify the relevant classifications that apply to the context in which it has been invoked and retrieves them. The Annotation dialog finally renders the classifications

in a form suitable for browsing and selecting terms and concepts. The user can then select a concept and the Profile Editor adds that to the form details.

Finally, the user saves the profile, by sending a message to the Profile Editor, which on its turn sends the data with a save request to the Cloud Storage Metadata bucket.

A final confirmation is received for the success of the save operation.

• Search member profile

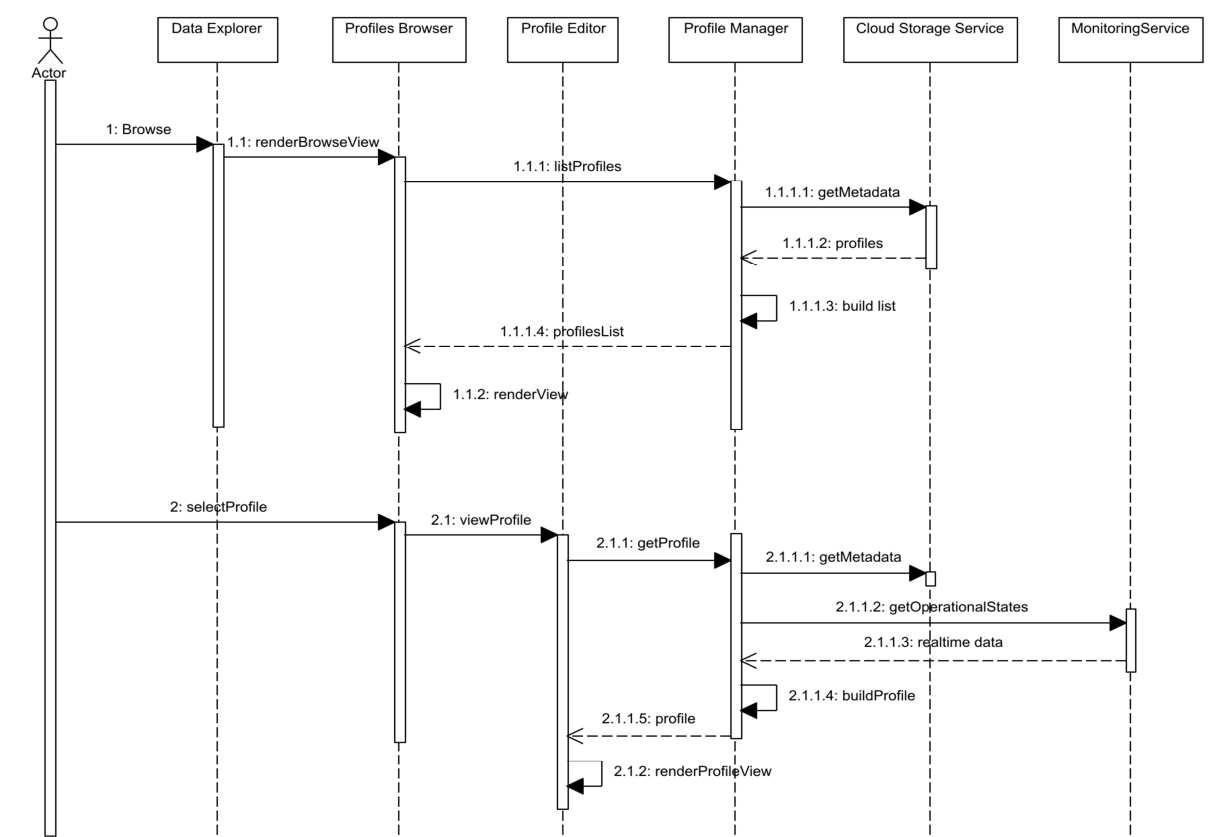


Figure 22 - Sequence diagram for Search member profile

In this scenario the user wants to find and view relevant ADVENTURE members’ profiles. The user may want to look at the ADVENTURE community in general or user may search for potential business partners. Only information for the ADVENTURE members that is “public” or shared for the user can be viewed.

The user sends a request (with some criteria specified) for profiles information to the Data Explorer that on its turn renders a Browser View of the Profiles Browser UI component. Further, the request is transferred to the Profile Manager that retrieves

the profiles and the respective metadata from the Cloud Storage (through the Cloud Storage Service), builds a list of the profiles found and sends this list back to the Profiles Browser, which displays the information received in a user friendly way.

The user may then choose a concrete profile and look at its details. A request is sent to the Profile Editor component that transfers that request again to the Profile Manager. The last retrieves the relevant for the profile static and dynamic data through connecting to the Cloud Storage Service (for static metadata) and the Monitoring service (to get the operational/dynamic data) respectively. The complete profile is then built and sent back for visualization to the Profile Editor where the user may look at the details of that ADVENTURE member.

• Find services for activity

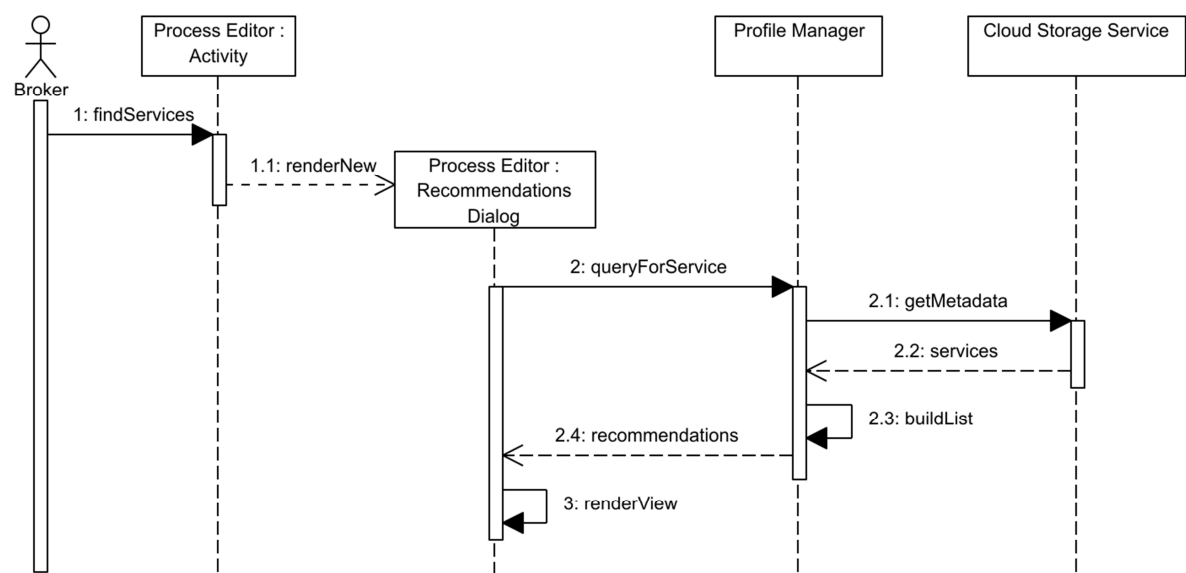


Figure 23 - Sequence diagram for Find services for activity

Finding relevant services for activities in the ADVENTURE process models is one of the main use-case scenarios. During the process design the user has specified through the Process Editor component the activities within the process, including some concrete requirements and as well non-functional constraints such as carbon footprint or quality. The user requests the system to retrieve relevant services (partners) that best fit to the concrete activity and the criteria specified. A Process Editor Recommendation Dialogue is rendered that sends a query for services to the Profile Manager component. Through a semantic search the Profile Manager retrieves the information/metadata from the Cloud Storage (through the Cloud

Storage Service) and builds a list of relevant services. The list is then sent to the Recommendation Dialogue that displays the recommended services and their information in a suitable for the user form.

### 3.3.4 Specification of Interfaces, Protocols and Formats

As specified in previous sections, DPD will present standard protocols and API and specific data model to operate.

#### 3.3.4.1 API specification

DPD presents two APIs to applications consuming its functionality. On one hand it offers OData as service API and on the other - SPARQL. Both constitute the DPD service API that is REST compliant and therefore is best described from the perspective of HTTP resources, requests and responses.

##### 3.3.4.1.1 OData API

The OData protocol requires the identification of several top-level resource types where usually users start data exploration. Orthogonal, or hierarchically there could be also other linked resources to create a directed graph-like data structure. The DPD OData API description will first present the resource hierarchy and then the operations that can be executed on it, using the core Organization resource as an example. In principle the operations semantic is defined in the OData protocol itself and it doesn't change in DPD service API and so it's applicable to any resource in the DPD data model.

##### 3.3.4.1.1.1 Resource hierarchy

URL path	Description
/Organizations	The Organizations which are the top-level business entities in ADVENTURE.
/Organizations/Factories	The factories are the physical entities that belong to an Organization and are the actual VF participants providing manufacturing services
/Organizations/Factories /Services	A manufacturing service offered by a Factory
/Organizations/Factories /Processes	The processes, which the Factory (and indirectly the Organization) owns or has a role in.

/Attachments	The attachments that can accompany a service/product or activity description.
/Organizations/Factories/Products	The Products that a Factory offers or seeks through the service it provides

### 3.3.4.1.1.2 Create new member organization

**Resource path:** /Organization

**Method:** POST

**Description:**

When a client needs to create multiple related Entries it can do so as independent operations or, if the Links between Entries allow it structurally, the users can perform a single POST with a tree of Entries. The tree is formed by using inline expansion in OData-Atom or OData-JSON format. All expanded Entries are considered new. Servers process a request with inline Entries by creating individual Entries and then linking them in the same way linking would have happened in an independent request.

**Example:**

An example create request with Atom formatted content.

Request	
Request line	POST http://adventure.org/odata/members/Organizations HTTP/1.1
Request headers	accept: application/atom+xml content-type: application/atom+xml
<pre>&lt;?xml version="1.0" encoding="utf-8"?&gt; &lt;Entry xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"   xmlns="http://www.w3.org/2005/Atom"&gt;   &lt;updated&gt;2010-02-27T21:36:47Z&lt;/updated&gt;   &lt;category term="DPD.Organization"     scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" /&gt;   &lt;Link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Factories"     type="application/atom+xml;type=feed" title="Factories" href="Organizations(1)/Factories"&gt;   &lt;m:inline&gt;</pre>	

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>116</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



```

<feed>
<Entry>
<updated>2010-02-27T21:36:47Z</updated>
<category term="DPD.Factory"
      scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
<content type="application/xml">
<m:properties>
<d:ID m:type="Edm.Int32">10</d:ID>
<d:legalname>Phillips Electronics Ltd</d:legalname>
<d:Description>The Phillips sub-organization that provides services in the field of
electornics</d:Description>

...other properties and links skipped for brevity...

</m:properties>
</content>
</Entry>
</feed>
</m:inline>
</Link>
<content type="application/xml">
<m:properties>
<d:ID>10</d:ID>
<d:legalName>Phillips</d:legalName>

...other properties and links skipped for brevity...

</m:properties>
</content>
</Entry>

```

**Response**

Status line	HTTP/1.1 201 Created
Response headers	Content-Length: 1072

	Date: Sat, 27 Feb 2010 21:39:54 GMT Location: http://adventure.org/odata/members/Organizations(10) Content-Type: application/atom+xml;charset=utf-8
<pre> &lt;?xml version="1.0" encoding="utf-8" standalone="yes"?&gt; &lt;Entry xml:base="http://adventure.org/odata/members" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns="http://www.w3.org/2005/Atom"&gt; &lt;id&gt;http://adventure.org/odata/members/Organizations(10)&lt;/id&gt; &lt;updated&gt;2010-02-27T21:39:54Z&lt;/updated&gt; &lt;Link rel="edit" title="Organization" href="Organizations(10)" /&gt; &lt;Link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Factories"   type="application/atom+xml;type=feed" title="Factories" href="Organizations(10)/Factories"/&gt; &lt;category term="DPD.Organization"   scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/&gt; &lt;content type="application/xml"&gt; &lt;m:properties&gt; &lt;d:ID m:type="Edm.Int32"&gt;10&lt;/d:ID&gt; &lt;d:legalname&gt;Philips&lt;/d:legalname&gt;  ...other properties and links skipped for brevity...  &lt;/m:properties&gt; &lt;/content&gt; &lt;/Entry&gt; </pre>	

#### 3.3.4.1.1.3 List organizations

**Resource path:** /Organizations

**Method:** GET

**Description:**

OData clients retrieve an organization list or individual entry issuing an HTTP GET request against its URI. Servers respond with the feed, Entry or service document in the response body in the proper format as instructed by the `accepts` request header.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>118</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

A feed individual can be requested by pointing at its index. For example, this request URL retrieves the first entry in the feed:

```
/Organizations(0)
```

While in many cases clients will directly follow links provided externally or found in a service document to find and retrieve a feed for a Collection, often a client will want to further control how the feed is returned. OData provide a series of optional conventions to allow clients to filter, sort and page over data in Collections, as well as to request for a subset of the properties to be sent and to expand related entries inline. For example, this request URL retrieves organizations with `owner` property value 'someuser', sorted by `name` in ascending order, and asks the server to only retrieve the `email` and `Website` properties:

```
/Organizations?$filter=owner eqsomeuser&$orderby=name&$select=email,Website
```

### Example:

An example request to list organizations with Atom formatted response.

Request	
Request line	GET http://adventure.org/odata/members/Organizations HTTP/1.1
Request headers	accept: application/atom+xml content-type: application/atom+xml
Response	
Status line	HTTP/1.1 200 OK
Response headers	Content-Length: 5685 Date: Sat, 27 Feb 2010 21:39:54 GMT Content-Type: application/atom+xml; charset=utf-8
<pre>&lt;?xml version="1.0" encoding="utf-8" standalone="yes"?&gt; &lt;feed xml:base=" http://adventure.org/odata/members" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns="http://www.w3.org/2005/Atom"&gt; &lt;title type="text"&gt;Organizations&lt;/title&gt; &lt;id&gt;http://adventure.org/odata/members/Organizations&lt;/id&gt;</pre>	

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>119</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

```

<updated>2010-02-27T20:03:28Z</updated>
<Link rel="self" title="Organizations" href="Organizations" />
<Entry>
<id>http://adventure.org/odata/members/Organizations(0)</id>
<updated>2010-02-27T21:39:54Z</updated>
<Link rel="edit" title="Organization" href="Organizations(0)" />
<Link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Factories"
type="application/atom+xml;type=feed" title="Factories"
href="Organizations(0)/Factories"/>
<category term="DPD. Organization"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<content type="application/xml">
<m:properties>
<d:ID m:type="Edm.Int32">10</d:ID>
<d:legalname>Philips</d:legalname>

...other properties and links skipped for brevity...

</m:properties>
</content>
</Entry>
<Entry> ... </Entry>
<Entry> ... </Entry>
<Entry> ... </Entry>
</feed>

```

#### 3.3.4.1.1.4 Delete organization

**Resource path:** /Organizations

**Method:** DELETE

#### **Description:**

Organizations are deleted by executing an HTTP DELETE request against a URI that points at the Entry. If the operation executed successfully servers should return 200 (OK) with no response body.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>120</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

If the Entry has dependent Entries such as Entries Linked to it, it is up to the server whether deletion should be cascaded, the operation should fail because of the presence of dependent Entries, or if Links are left dangling after the Entry is deleted.

#### Example:

An example request to delete organization the organization with index 0:

Request	
Request line	DELETE http://adventure.org/odata/members/Organizations(0) HTTP/1.1
Response	
Status line	HTTP/1.1 200 OK

#### 3.3.4.1.1.5 Update organization

**Resource path:** /Organizations

**Method:** PUT

#### Description:

To update the data in an Organization Entry clients execute an HTTP PUT request against the Entry's URI, with a new Entry resource included in the request body.

According to the AtomPub protocol specification, the PUT request replaces the existing Entry, so all property values in the Entry either take the values indicated in the request body, or are reset to their default value if not mentioned in the request.

#### Example:

An example request to list organizations with Atom formatted response.

Request	
Request line	PUT http://adventure.org/odata/members/Organizations HTTP/1.1
Request headers	accept: application/atom+xml content-type: application/atom+xml
<pre>&lt;?xml version="1.0" encoding="utf-8" standalone="yes"?&gt; &lt;Entry xml:base="http://adventure.org/odata/members" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"</pre>	

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>121</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

```

    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
    xmlns="http://www.w3.org/2005/Atom">
    <id>http://adventure.org/odata/members/Organizations(10)</id>
    <updated>2010-02-27T21:39:54Z</updated>
    <Link rel="edit" title="Organization" href="Organizations(10)" />
    <Link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Factories"
      type="application/atom+xml;type=feed" title="Factories"
      href="Organizations(10)/Factories"/>
    <category term="DPD.Organization"
      scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
    <content type="application/xml">
    <m:properties>
    <d:ID m:type="Edm.Int32">10</d:ID>
    <d:legalname>Philips</d:legalname>

    ...other properties and links skipped for brevity...

    </m:properties>
    </content>
  </Entry>

```

**Response**

Status line	HTTP/1.1 204No Content
Response headers	Date: Sat, 27 Feb 2010 21:39:54 GMT

**3.3.4.1.2 SPARQL API**

The SPARQL service API defines protocols for handling HTTP requests for query and update of the managed RDF datasets. The sections below will specify the protocol details for constructing query and update service requests and then the details for defining insert/delete statements (which translate to create/update and remove operations on resources) in the update requests.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>122</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### 3.3.4.1.2.1 Query data

SPARQL1.1. Protocol allows for sending queries to a SPARQL endpoint via standard HTTP GET method. Since the protocol mandates that the query is encoded in the request URL as URL-encoded value of the URL property it also specifies an alternative due to the known limitation for the size of the URL string. The alternative is to use a standard HTTP POST method for the same purpose.

A sample query with a GET request would look like:

```
GET http://adventure.org/sparql/?query=Encoded SPARQLQueryString
```

The query can be sent with a number of options in the form of query parameters, among which the URI of a named graph to evaluate against: *named-graph-uri*.

Representation format of the returned results can be negotiated via the HTTP request header. For example:

```
Accept: text/turtle, application/rdf+xml.
```

### 3.3.4.1.2.2 Update data

A dataset for an update request may be specified using the using-graph-uri and using-named-graph-uri parameters. The serialization of an example request sent to `http://localhost:8888/test` and specifying a dataset with default graph `http://localhost:8888/people` is shown below.

```
POST/test?using-graph-uri=http%3A%2F%2Flocalhost%3A8888%2Fpeople
```

```
HTTP/1.1
```

```
Host: localhost:8888
```

```
Accept: */*
```

```
Content-Type: application/sparql-update
```

```
Content-Length: 136
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
DELETE { ?person ?property ?value }
```

```
WHERE { ?person ?property ?value ; foaf:givenName 'Fred' }
```

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>123</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**3.3.4.1.2.3 Insert data**

Canonical form:

INSERT DATA QuadPattern

, where QuadPattern can be RDF statements template (implicitly meaning they are stored in the default context) or the statements can be wrapped optionally in a GRAPH block providing a context URL to relate the statements to. The canonical form including the optional GRAPH block is:

INSERT DATA { GRAPH<URI> {TriplesTemplate} }

Example (simplified to one property for readability):

<b>Insert data</b>
<pre>PREFIX dc: &lt;http://purl.org/dc/elements/1.1/&gt; PREFIX adv: &lt;http://adventure.org/members#&gt; INSERT DATA { GRAPH &lt;http://adventure.org/members/Organizations&gt; {   &lt;http://adventure.org/members/Organization1&gt;adv:hasFactory "Phillips Electronics" } }</pre>
<b>Data before</b>
<pre># Graph: http://adventure.org/members/Organizations @prefix gr:&lt;http://purl.org/goodrelations/v1.owl/&gt; . &lt;http://adventure.org/members/Organization1&gt;gr:legalName "Phillips" .</pre>
<b>Data after</b>
<pre># Graph: http://adventure.org/members/Organizations @prefix gr:&lt;http://purl.org/goodrelations/v1.owl/&gt; . @prefix adv: &lt;http://adventure.org/members#&gt;. &lt;http://adventure.org/members/Organization1&gt;gr:legalName "Phillips" . &lt;http://adventure.org/members/Organization1&gt;adv:hasFactory "Phillips Electronics".</pre>

**3.3.4.1.2.4 Delete data**

Canonical form:

(WITH IRIref )?

DELETE QuadPattern

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>124</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



( USING ( NAMED )? IRIref )\*

WHERE GroupGraphPattern,

Where, QuadPattern can be RDF statements template (implicitly meaning they are stored in the default context) or the statements can be wrapped optionally in a GRAPH block providing a context URL to relate the statements to. The canonical form including the optional GRAPH block is:

DELETE DATA { GRAPH<URI> {TriplesTemplate} }

Example (simplified to one property for readability):

<b>Insert data</b>
<pre>PREFIX dc: &lt;http://purl.org/dc/elements/1.1/&gt; PREFIX adv: &lt;http://adventure.org/members#&gt; WITH http://adventure.org/members/Organizations DELETE{?org ?p ?v} WHERE {   ?org ?p ?v;   adv:hasFactory "Phillips Electronics" } }</pre>
<b>Data before</b>
<pre># Graph: http://adventure.org/members/Organizations @prefix gr:&lt;http://purl.org/goodrelations/v1.owl/&gt; . &lt;http://adventure.org/members/Organization1&gt;gr:legalName "Phillips" . &lt;http://adventure.org/members/Organization1&gt;adv:hasFactory "Phillips Electronics".</pre>
<b>Data after</b>
<pre># Graph: http://adventure.org/members/Organizations @prefix gr:&lt;http://purl.org/goodrelations/v1.owl/&gt; . @prefix adv: &lt;http://adventure.org/members#&gt;.</pre>

3.3.4.2 Content Formats and Protocol Definitions

The data model on which the content formats for the supported protocols will build is outlined on Figure 24.

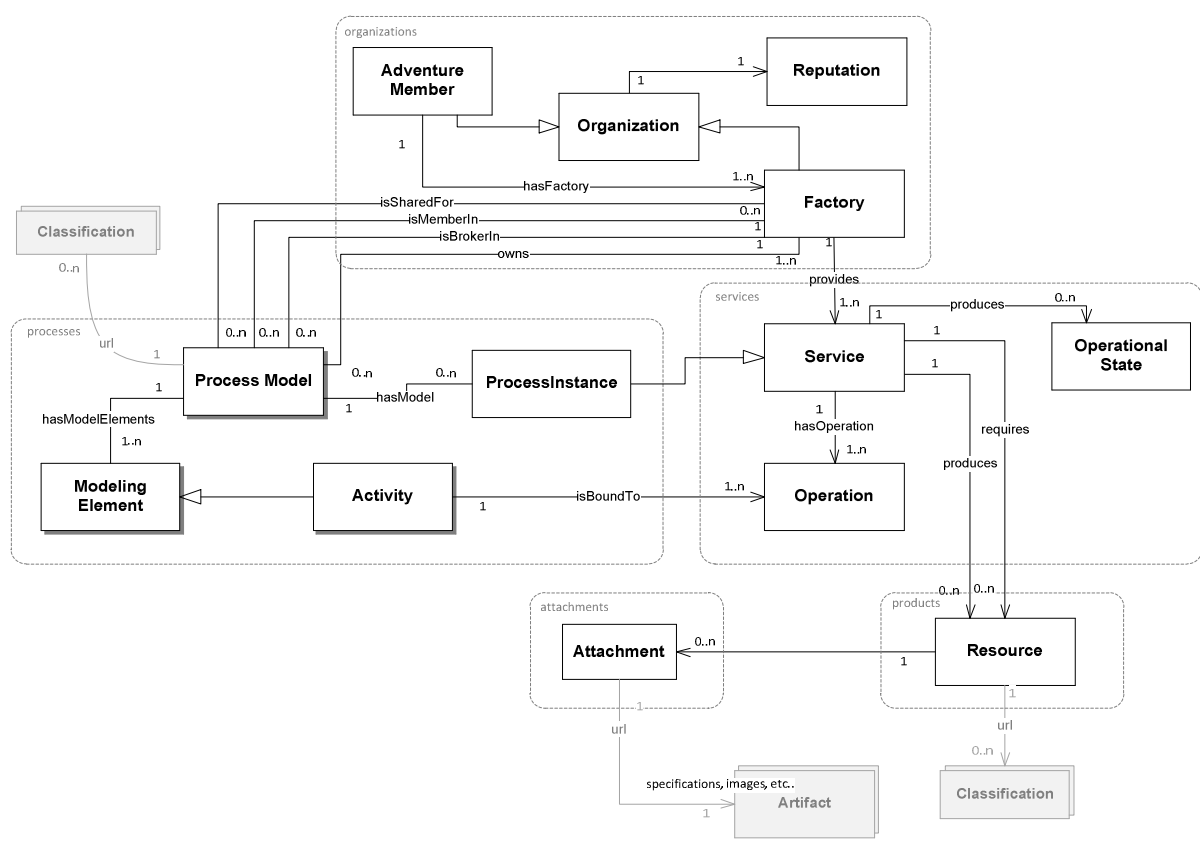


Figure 24 - DPD conceptual data model

Properties, except important associations are not shown for better readability. The dashed-line boxes represent different contexts for the domain objects. These would translate to named graphs in SPARQL and collections in OData. Note that this is one of the possibly numerous projections over the data model. Other possibilities that might need to be implemented for example are contextualizing by member profile (i.e. all data that belongs to a profile instance) or user identity (for realizing access control on the RDF data).

The OData JSON formatted request payload below shows an (incomplete) example of translation of the generic data model to the OData protocol in JSON format.

<pre>[{   "__metadata": {     "uri": "http://adventure.org/odata/members/Organizations(0)",     "type": "Members.Organization"   }, </pre>			
D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: 126 / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

```

    "ID": 0, "gr:legalName": "iSoft", "gr:hasICISv4": "12345",
    "Factories": {
    "__deferred": {
    "uri": " http://adventure.org/odata/members/Organizations(0)/Factories"
    }
    }
  }}

```

### 3.3.5 Summary

The Data Provisioning and Discovery component will use the most advanced state-of-the-art in collaborative organizations modeling and will present an innovative general-purpose SaaS model management implementation. It is designed using edge technologies for maximum effect and adoption. The component design will serve all requirements defined by ADVENTURE and also has the ambitious goal to provide a valuable contribution that spans way beyond the ADVENTURE project boundaries and lifetime, and to fill the gap that became apparent during the state-of-the-art analysis.

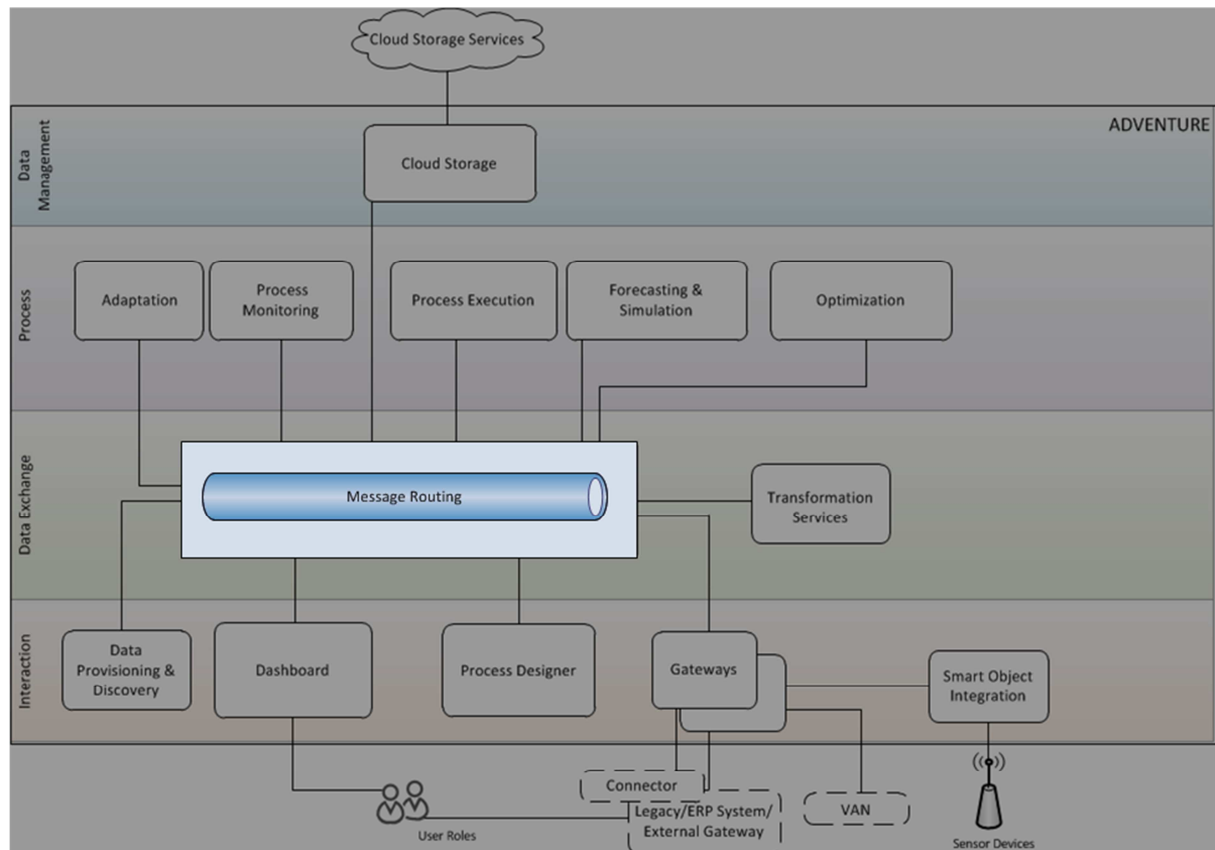
The DPD model will be designed, based on recent research and development in the field such as Linked USDL Core ontology, Schema.org and GoodRelations and will seek to combine those for best results along with own contributions where appropriate. The model shall also use established taxonomies and classifications like UNSPSC, NACE or ISICv4 (non-normative list) and will be generally open which one to select. Technically, the model will be serialized in the highly extensible RDF technology that underpins currently the efforts for a global Web of data.

The DPD software is designed to provide access to the model via well-defined protocols and most of all with wide adoption in mind. It exposes a set of popular REST-oriented service interfaces that implement the most popular protocols in the field – SPARQL and OData, as well as a lightweight UI. The software is designed for cloud platforms (i.e. delivered as SaaS) primarily, but can also be adopted as a tightly integrated component in a system.

Those design decisions and approaches will ensure that the prototypes will deliver a highly consumable software solution for modeling collaborative organizations that will create even more business opportunities than ADVENTURE itself when seen as standalone Web datasource.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>127</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

# Message Routing



## 3.4 Message Routing

Abstract:

Message Routing provides a messaging service that can be used for any kind of remote service call and for data delivery

It supports a holistic message format used for all intra-project communication

A class library for connecting to the Message Routing, and using it to receive and send messages is provided

### 3.4.1 Major Design Decisions

Message Routing has to work asynchronously. It has to be unobtrusive and should be easily usable as a service by ADVENTURE components.

Message Routing should have a very high throughput of messages, as all components will use those to call services in other components.

It must provide secure communication.

Message Routing also has to facilitate the routing of structured text messages (like XML or JSON) as well as binary files and has to acknowledge the completed transfer with a basic acknowledgement message to the sender.

It also has to buffer messages and binary files in case of a receiver not being able to receive the message / file or in case the receiver is offline.

### 3.4.2 Technology Comparison and Selection

This subsection will compare existing technologies for the Message Routing component, including RESTful Web services, WSDL Web services, XMPP, ServiceMix and Active MQ / Apache Camel. These technologies are further discussed as well as the rationale behind deciding for or against these solutions for exploiting them in Message Routing.

#### 3.4.2.1 Selection Criteria

The selection criteria for Message Routing technologies were defined in the *D3.2 Functional Specification* in section 5.5.5 on page 123. The description of the generic parameters can be found in the *D3.2 Functional Specification* in section 5 on page 28. Now follows the description of the Message Routing specific parameters:

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>129</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Reliable Messaging / Receipt Acknowledgement:** As ADVENTURE deals with business data, it must be ensured that Messages transferred by the Message Routing actually reach their destination. Hence, acknowledgement receipts must be returned. Moreover, speed is an important factor, as all sent messages will produce another message.

**Provide binary message exchange:** Files will have to be exchanged, and when Message Routing handles all messages between components, it should also be able to handle file exchanges.

**Secure communication protocol:** The ADVENTURE members should have confidence that their data is securely transmitted. Therefore, the solution should include the possibility to have encrypted communication.

**Signed messages:** In many cases, it is important to be able to guarantee that a message's origin is what it seems to be, especially as business orders can be triggered by ADVENTURE.

**Provide message buffering:** When internet connection isn't available, messages must be buffered and sent as soon as possible. Even if it seems to be a small problem nowadays, the Message Routing also handles messages from and to smart sensors, which might be frequently be offline.

**Point-to-Point Messaging for component instances:** The main functionality of Message Routing is to connect different components that might reside on different servers all over the Internet. Additionally, as a cloud-based architecture is used in ADVENTURE, several instances of one component might need to send messages to other components with the same role.

**Ability to treat users as message partners:** Some components such as Monitoring will send messages to the users directly if something needs to be reported to these users. If Message Routing can handle this, it is a bonus.

**Multi-recipient-messaging:** Some messages might need to be broadcasted or might need to target multiple components. Thus the messaging load might be reduced, as the message would only need to be transferred to the message servers once and can be sent to the receivers from there.

**Multi-instance-Server for cloud hosting:** As cloud architecture is planned to ensure reliability, the message routing solution should be able to work with multiple servers that are connected in a federation, otherwise the whole cloud architecture would rely on one server running.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>130</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Provide configuration UI for routing:** As UIs tend to take a lot of implementation time, having a graphical configuration UI as part of the underlying technology stack would be a bonus.

**Provide user-message UI:** If messaging to users is possible, having a messaging UI intended for users would be a bonus too. Very low priority.

**Message partner presence awareness:** If the technology allows knowing which components or users are accessible, message load can be kept down, as messages could be sent only when the receiver is accessible.

**Uses UTF-8:** D3.2 demanded for a technology supporting UTF-8, so there won't be any conversions needed in character sets and all components can expect to work with UTF-8.

**Lightweight infrastructure for remote SME endpoints:** As the messaging might be integrated in many systems, having a lightweight system that is easily set up and has minimum requirements is necessary. Additionally, as it needs to be included in all components, a lightweight system would be easier to be used by ADVENTURE.

**Communication partner registry included:** Message Routing will provide a way to show a registry of all component instances that are registered. If this is supported by the underlying technology, this would be a bonus.

### 3.4.2.2 Possible Technologies

In order to realize the Message Routing component several options have to be considered as base technology. The assessment of these technologies is presented in the following sections. Additionally a message format has to be defined or adopted from the chosen technology to submit metadata needed by ADVENTURE in the transmitted messages.

Please note: The selection of possible base technologies in this subsection is based on experiments that the consortium made with different technologies. The best individual recommendations have been investigated. Therefore, the following selection will compare only the 7 most promising technologies.

**REST / SOAP (WSDL) WebServices** - Using basic REST or WSDL based Web services is a widely accepted and most standards-conformant pattern for implementing all kinds of APIs all over the Web. Within ADVENTURE this will also be used in the Gateways and the Process Execution components, as those need to interfere with external services provided by partners. For the Message Routing these

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>131</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

standards don't help much, as the high quantity of exchanged messages demands for a constant connection and makes the long request times of HTTP a showstopper.

**XMPP** - XMPP is the abbreviation for eXtensible Messaging and Presence Protocol and is a standard for XML-based messaging over a socket connection upheld by connected clients. It is a stable and mature protocol. Facebook, Google and Apple use XMPP for their users' peer-to-peer messaging, which is a good indicator about its stability, maturity, performance and scalability. The extensibility of XMPP is organized by the XMPP Standards Foundation, which collects processes and formalizes the extensions into standards<sup>6</sup>. This is for example shown by Google Talk, which routes peer-to-peer voice chats or big file transfers over XMPP, and by other projects that also transfer video streaming data and much more performance-critical data. Interoperability is also well supported, as there's a client library for XMPP integration for every popular programming language, and there are multiple Open Source XMPP servers to choose from.

**ActiveMQ** - Apache ActiveMQ is a messaging bus based on a similar socket connection like XMPP made for high throughput and with many client libraries available for different languages and platforms. It also has support for REST, SOAP, JMS, XMPP and other technologies, and integrates Apache Camel to use Enterprise Integration Patterns. Additionally it supports SSL and JAAS (Java Authentication and Authorization Service), supports different ways of clustering the servers and is used in many Apache projects. For ADVENTURE, ActiveMQ provides a too broad set of configurable functionality and the overhead of too many possibilities and configurations is perceived as an impeding factor for ADVENTURE. Another problematic factor is the real-world usage of ActiveMQ. The projects references all lie within the Apache community. Real-world problems like using socket connections through a router Network Address Translation or a server firewall can be a problem. Additionally, ActiveMQ and Camel include a lot of different technology concepts that would need to be understood by a broad range of implementers.

**ServiceMix** - ServiceMix is an Enterprise Service Bus and consists of ActiveMQ, Apache ODE, which is a complete BPEL engine, and a dozen other Apache projects, configured with Apache Maven and bundled via OSGi and running on integrated servers. While ServiceMix is a true jack of all trades device, it is so complex that only few users can be reported, and those that use ServiceMix have decided to use the

---

<sup>6</sup> <http://xmpp.org/xmpp-protocols/xmpp-extensions/>



**ebXML** - ebXML is not a protocol, but an XML based message format that has had big impact in real world businesses. As such, it could be regarded as a format for the XML content sent in messages. As such, it could provide the basis for specific messages sent via a socket connection when using ActiveMQ or ServiceMix, as ebXML includes a message service Specification and a registry service specification, both of which would be useful for ADVENTURE. But as is not a protocol, it's not included into the matrix, because it cannot serve as a base technology for Message Routing.

**AS3** - Another protocol that was regarded because of its role in business communications. AS3 transfers files via FTP and is a commercial, closed source communication protocol. As such, it doesn't fulfill performance criteria and despite its high security standards it has not much to offer to ADVENTURE. Also, AS3 has had no big impact on real world businesses, where AS2 is more common, maybe because of the commercial aspect.

Parameter	Importance	REST / WSDL Web	XMPP	ActiveMQ + Camel	Service Mix ESB	AS2	AS3
D3.3-Technical-Specification			Author: UVA and Partners		Date: 2012-09-24		Page: <b>133</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.							

		Services					
<b>Generic Parameters</b>							
Maturity & Stability	+++	+++	+++	+++	+++	+++	+++
Regularly Updated	+	+	+	++	++	-	-
Technical Up-to-Dateness / Appeal	++	++	+++	++	++	-	-
Open Source	+	YES	YES	YES	YES	YES	NO
Infecting Open Source License	NO	NO	NO	NO	NO	NO	N/A
Code-Quality	+	N/A	N/A	N/A	N/A	N/A	N/A
Extensibility	+++	-	+++	+++	+++	-	-
Community	+	+++	++	+++	+++	++	-
Performance / Scalability	++	+	+++	+++	+++	+++	+++
Reuse of existing developments	++	+++	+	+	+	+	+
EU project origin	--	NO	NO	NO	NO	NO	NO
Platform (Portability)	+++	+++	+++	+++	+++	+	+
Open Standards Compliance	+	+++	+++	+++	+++	+	-
Interoperability (easy integration for all platforms)	+++	+++	+++	+++	+++	+	-
<b>Specific Parameters</b>							
Reliable Messaging / Receipt Acknowledgement	+++	+++	+++	+++	+++	+++	+++
Provide binary message exchange	++	-	+++	+++	+++	+++	+++
Secure communication protocol	+++	+++	+++	+++	+++	+++	+++
Signed messages	+/-	++	++	++	++	+++	+++
Provide message buffering	+++	-	+++	+++	+++	-	-
Point-to-Point Messaging for Component Instances	+++	+++	+++	+++	+++	+++	+++
Ability to treat users as message partners	++	-	+++	-	-	-	-

Multi-recipient-messaging	++	-	+++	+++	+++	-	-
Multi-instance-Server for cloud hosting	++	-	+++	-	+++	-	-
Provide configuration UI for routing	+	-	+++	-	-	-	-
Provide user-message UI	-	-	-	-	-	-	-
Message Partner Presence Awareness	+	-	+++	-	+++	-	-
Uses UTF-8	+++	+++	+++	+++	+++	+++	+++
Lightweight Infrastructure for Remote SME Endpoints	++	+++	+++	-	-	-	-
Communication Partner Registry included	+	-	+++	-	-	-	-

### 3.4.2.3 Technology Selection

HTTP based communication, i.e. SOAP or REST, was not selected because of the long request times of a single request. The same holds true for FTP and SMTP based communications that can be done with AS2 and AS3. The other technologies support constant socket connections that are necessary for the problem space and also fulfill all other generic and specific parameters from the functional specification, but AS2 and AS3 that basically don't fulfill most of the requirements and were disregarded very early.

ServiceMix was not selected, because of two main reasons. Firstly, it is a completely general framework that tries to solve every problem at once and for this includes loads of different technologies, frameworks and concepts. This is much too generic for the ADVENTURE problem space and the consortium had experienced problems with the breadth of functionality of ServiceMix in other projects before. This breadth of functionality directly translated into overhead in setup work, configuration work and learning, necessary for using the different concepts, languages and technologies that need to be understood and applied with the ServiceMix framework. Secondly, there is no notable real-world application that has been built using ServiceMix.

Between the last contenders - XMPP and ActiveMQ, the differences weren't that big. The complexity of ActiveMQ is higher than that of XMPP, but basically both fit the ADVENTURE problem space well. The decision fell for XMPP, because it is the

easier concept and because it suits the problem space without much configuration and provides most of the necessary concepts out-of-the-box. Additionally, the real world applications using XMPP, including the messaging systems of Google, Apple and Facebook, show that XMPP is a scalable and mature first class technology, with a good stability and performance. The ease of integration is shown by number of messaging clients in different programming languages on different platforms which all manage to work very well, interoperating with Google's or Facebook's messaging system. Interoperability is a given for XMPP as there's some form of client library for every major programming language, and there are multiple XMPP servers to choose from<sup>7</sup>.

**XMPP fulfills all the generic parameters:** XMPP was investigated also with respect to the specific parameters from the table.

- Reliable Messaging / Receipt Acknowledgement as described in XMPP-XEP-0184<sup>8</sup>.
- Provide binary message exchange as described in XMPP-XEP-0231<sup>9</sup>. If no out of band-messaging XMPP-solution exists for binary message transfer, this could be also realized using the Cloud Storage component.

**Secure communication protocol:** SSL is a default technology for this need, and is easily employable within XMPP.

**Signed messages:** All XMPP messages are automatically connected to information about the sender, but it is also possible to sign messages using XMPP-XEP-0027<sup>10</sup>, which employs Open-PGP to sign messages.

**Provide message buffering:** Message Buffering will be implemented using the Cloud Storage component or an implementation of XMPP-XEP-0231.

**Point-to-Point Messaging for Component Instances:** This is the core functionality of XMPP.

**Ability to treat users as message partners:** With XMPP, this is the default. Component instances as well as users will be supported. XMPP uses a system of groups, which represent sets of known message partners. ADVENTURE can use

<sup>7</sup> <http://xmpp.org/xmpp-software/>

<sup>8</sup> <http://xmpp.org/extensions/xep-0184.html>

<sup>9</sup> <http://xmpp.org/extensions/xep-0231.html>

<sup>10</sup> <http://xmpp.org/extensions/xep-0027.html>

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>136</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

these groups to differentiate between components, a component's type, users and user's organizations.

**Multi-recipient-messaging:** Most XMPP servers implement this feature, which is defined in XMPP-XEP-0033<sup>11</sup>.

**Multi-instance-Server for cloud hosting:** Most XMPP servers support multi-instance usage. The servers will most likely not need to be changed, in order to work as a cloud backend of the XMPP services.

**Uses UFT-8:** This is default for XMPP and was defined in Extensible Message and Presence Protocol (XMPP): Core<sup>12</sup>.

**Lightweight Infrastructure for Remote SME Endpoints:** XMPP is a communication standard and supplies libraries for most programming languages. These clients will be wrapped in an easy to use client library for ADVENTURE that can be dropped in components in a plug and play manner. The configuration for this library should be minimal. At least a client library for C# .NET and Java will be implemented.

**Communication Partner Registry included:** The server part of Message Routing will contain a registry of all partners that are registered with the ADVENTURE Message Routing. This is contained in some XMPP servers and if not, additional logic has to be implemented that cares for the communication partner registry.

**Message Partner Presence Awareness:** Presence awareness is an XMPP feature that can be used to determine if the needed component is online and if it can be used. An XMPP-server stays connected to the XMPP servers all the time and Message Partners can therefore determine if a component or user is online and can receive a message. This is default XMPP behavior.

**Provide configuration UI for routing:** This is also included in most XMPP servers and should be able to fulfill the concrete ADVENTURE requirements.

If an unforeseen problem occurs that cannot be solved using XMPP, the closest contender ActiveMQ can still be used, as it can also use the XMPP protocol. The work and research invested into XMPP would therefore not be lost.

### 3.4.2.3.1 Data Protocols

---

<sup>11</sup> <http://xmpp.org/extensions/xep-0033.html>

<sup>12</sup> <http://xmpp.org/rfcs/rfc3920.html>

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>137</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

XMPP transfers XML data. In this, binary data can be easily included in serialized form. The metadata that cannot be delivered through the basic XMPP message structure can be added as XML metadata. The structure of the message body is specified by the receiving components, as they need to define which data is needed in the XML message body in order to carry out tasks or respond appropriately.

#### 3.4.2.4 Missing Elements and Implementation Needs

The XMPP message format cannot be easily changed without adapting existing libraries and servers and therefore ADVENTURE specific data needs to be sent inside of the body of the XMPP message. Each component therefore can and must define its own message body. This should include a service name to be called and a respective array of key-values pairs for the parameters needed. The different component should provide an XML Schema (.xsd), so that other components can use the given format to send XML messages matching this schema. This will ensure that the receiver component can interpret the message.

When the different components have published their XML Schema files, the message client can include an object factory for the different messages, so it can provide a simple API to call other ADVENTURE components and automatically generate the necessary XML messages for the known services.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>138</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

3.4.3 Technical Component Specification

3.4.3.1 Component Structure

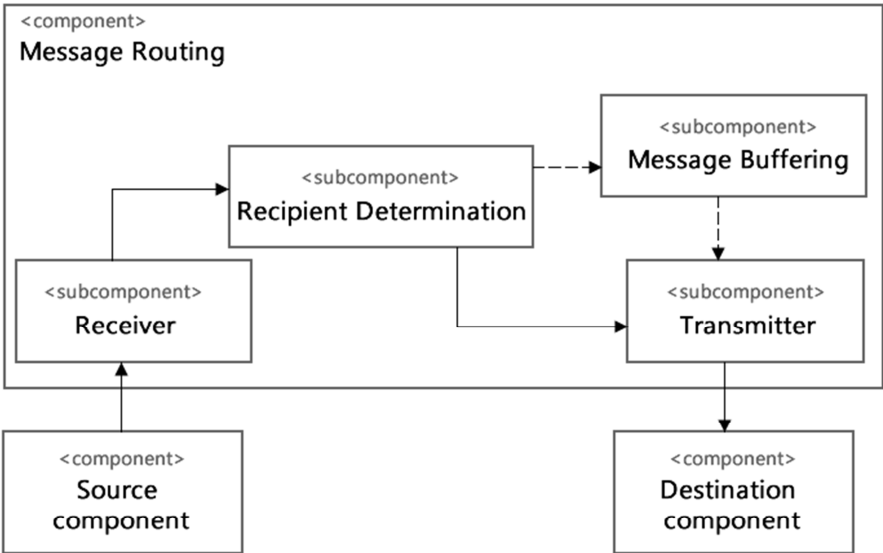


Figure 25 - Message Routing component structure

Message Routing consists of the following parts:

Source and Destination Component

At least for Java and C#/.NET a client library will be released that handles the communication with other ADVENTURE messaging partners and provides utility functions. If needed, an Objective-C library will also be provided.

This client library is needed, as the other components must connect to the Message Routing servers using a socket connection, implementing the XMPP standards. The only way to consistently provide this functionality to the ADVENTURE components is to include a library in the other components.

With this client library that implements the Message Routing API and realizes the communication with ADVENTURE, the components can easily send and receive messages in an event driven and well performing way. An event handler will be triggered, if a message is received and in order to give the client the opportunity to handle the message.

What is needed for configuring the client library is the authentication information like internal username and password for the connected component and a resource identifier that uniquely identifies the client object, as multiple clients for the same account could be connected at the same time, if for example multiple instances of the Cloud Storage component are deployed. The configuration of these credentials should be done through the configuration GUI of the Message Routing servers.

A list of URLs of servers that are part of the ADVENTURE XMPP network and that will be tried to connect to be also needed, but this will be integrated in the API. Additionally, it should be possible to connect to a custom server.

### **Receiver, Recipient Determination and Transmitter**

These components are part of the XMPP servers, which provide the glue between the client libraries used by the other components. They provide the connection point to these clients, uphold the socket connection with them, manage online presence, and forward messages to message partners.

OpenFire<sup>13</sup> has been selected for the first prototypes. It is an actively developed XMPP server that provides all necessary security and performance features needed. It also works out-of-the-box with multiple instances that can be hosted in the cloud, to fulfill all needed scaling requirements that might come up when there is a very high message load. Additionally, there is a plugin architecture for OpenFire, so available plugins can be used or own plugins can be added, if more functionality is needed.

### **Message Buffering**

XMPP defines a strict policy for clients to post a status that shows them as available when they are online to circumvent the buffering of messages. Messages can be buffered client-side this way. This does not work for ADVENTURE, where messages need to be buffered and sent as soon as the recipient becomes available.

---

<sup>13</sup> <http://www.igniterealtime.org/projects/openfire/>

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>140</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



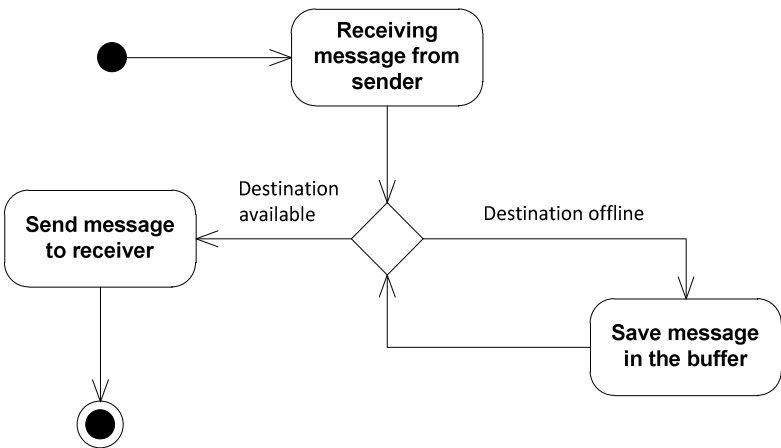


Figure 26 - Activity Diagram Message Routing with Buffering

Therefore the Message Buffering mechanism must be added in the client library wrappers. When the client library determines that the target of a message is not available, it sends the Message to a specific client running on a dedicated server component instead. In this message, all information needed is included, so the server component can save the message (including any file transfers) in a buffer and later on, when the destination component's status is available the buffering server can resend the message to the recipient. The buffer will be implemented in the Cloud Storage.

3.4.3.2 Internal sequence diagrams

This section will give an insight about the internal component communication sequence based on a list of example sequences. It's extracted from the use case scenarios of D3.2.

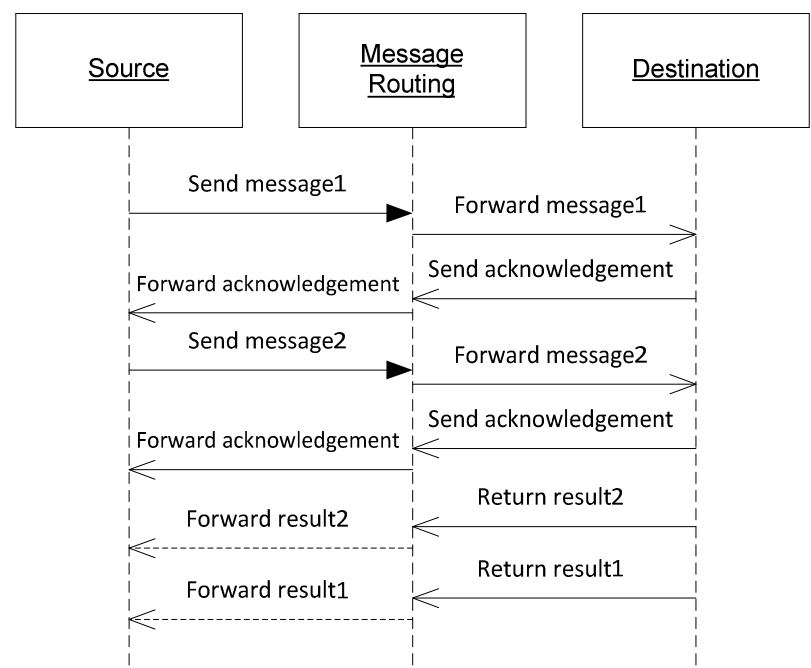


Figure 27 - Sequence diagram for Two Message Exchanges using Message Routing

This sequence diagram shows how the asynchronous acknowledgement system in the Message Routing will work. The source component sends two messages.

3.4.4 Specification of Interfaces, Protocols and Formats

To communicate with other components all components use the Message Routing API that is described in section API Specification and that is released as a class library. The interfaces that will be used by the components via the class library provided by Message Routing will be explained extensively, while the private APIs will only be shown as a class diagram. The section Content Formats and Protocol Definitions shows the basic message format and also includes the format with examples using OData.

3.4.4.1 API Specification

The Message Routing component will provide a list of API methods that can directly be used by the other ADVENTURE components using the client library.

### 3.4.4.1.1 Class MessageClient

#### MessageClient Constructor

```
public MessageClient(String username, String password, String resourceId =  
                      null, String serverurl = null);
```

#### Parameters

username: The username issued for the messaging account.

password: The password issued for the messaging account.

resourceId: A unique identifier for a specific instance of one account. For example, several instances of Cloud Storage could have the same username and password, but use different resourceIds. If then a message would be sent to the username of Cloud Storage, it would be determined automatically to which instance the message would be routed unless a specific instance ID is targeted. When not specified

serverUrl: When another URL for connecting to the Message Routing network should be used than the officially known URLs, this can be specified by setting this parameter to the desired URL. Default: null.

#### Remarks

The constructor creates a MessageClient instance that is connected to the ADVENTURE Message Routing. Username and password have to match to the account of the respective component. The serverUrl is the only other external information that has to be known by the component. If the serverUrl is not provided (`null` value), the component looks up the serverUrls needed by checking different Web services that can publish this information. The resourceId is the identifier for a specific instance of one account. For example, several instances of Cloud Storage could have the same username and password, but use different resourceIds. If then a message would be sent to the username of Cloud Storage, it would be determined automatically to which instance the message would be routed.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>143</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

```
public void Connect();
```

*Parameters*

none

*Return Value*

none

*Remarks*

The connect method needs to be called after the events of the Message Client have been subscribed to, so that authentication errors can be handled. When a client is connected, it's status get's set to available, when it disconnects from the servers, it's status gets set to unavailable by the servers.

**3.4.4.1.1.1 Method SendMessage(AID recipient, String messageContent, Boolean requestAck)**

```
public AdventureMessage SendMessage(String recipient, String messageContent
                                   , String recResourceId = null,
                                   Boolean requestAck = false);
```

*Parameters*

recipient: This is the target username to which the message should be sent.

messageContent: This is the main message content to be routed to the target component, which can contain any kind of XML, JSON, other structured text or even unstructured text.

recResourceId: This is the resource identifier, which identifies the programmatic object in case multiple objects were created that use the same username to communicate. Defaults to null, which means that the target is automatically determined or the message goes to all available instances.

requestAck: This flag signalizes to the library that an acknowledgement message should be sent by the message routing component in response to the receipt of the message and defaults to false.

### *Return Value*

This method returns a pointer to the sent AdventureMessage.

### *Remarks*

This is the main message function to be used. If the caller wants to check if it message was received, the caller needs to observe if an acknowledgement related to a sent messageId is received.

This method is asynchronous.

#### **3.4.4.1.1.2 Method SendMessageSync(AID recipient, String messageContent, Boolean requestAck)**

```
public AdventureMessage SendMessage(String recipient, String messageContent
                                   , String recResourceId = null,
                                   int timeout = 60000);
```

### *Parameters*

recipient: This is the target username to which the message should be sent.

messageContent: This is the main message content to be routed to the target component, which can contain any kind of XML, JSON, other structured text or even unstructured text.

recResourceId: This is the resource identifier, which identifies the programmatic object in case multiple objects were created that use the same username to communicate. Defaults to null, which means that the target is automatically determined or the message goes to all available instances.

timeout: The timeout is the time in milliseconds how long the MessageClient waits until it receives an answer. If the timeout is reached, the method returns a pointer to the sent AdventureMessage instead of the received AdventureMessage. The default timeout is 60 seconds.

*Return Value*

This returns the response of the target component, and NOT a pointer to the sent message.

*Remarks*

This is the synchronous version of the main message function described under Method SendMessage above. It works the same way, but is synchronous and returns, when the response was received.

**3.4.4.1.1.3 Event OnMessage(AdventureMessage msg)**

```
public void AddOnMessageEventListener(OnMessageListener subscriber)
```

```
public void RemoveOnMessageEventListener(OnMessageListener subscriber)
```

```
public interface OnMessageEventListener {
    public void OnMessage(AdventureMessage message);
}
```

*Event subscriber interfaces to implement*

OnMessageListener: This is the interface a class has to implement to be able to receive AdventureMessages delivered by the MessageClient class which is connected to the MessageRouting servers.

*Implemented Method Parameters*

message: This is the message wrapper and contains the message content.

*Implemented Method Return Value*

none

*Remarks*

The MessageClient instance that is connected to the ADVENTURE Message Routing receives messages for the logged in account. To be able to forward the software using the MessageClient these messages, an object needs to be registered as an event listener to the MessageClient. This object has to implement the interface

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>146</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

OnMessageEventListener and then be added to the list of objects that want to be informed about an incoming message. This can be done by calling `AddOnMessageEventListener` and passing the object that implements `OnMessageEventListener` as parameter.

#### 3.4.4.1.1.4 Event OnAcknowledgement (string messageID, AID from)

```
public void AddOnAcknowledgementEventListener(OnAcknowledgementEventListener subscriber)
```

```
public void RemoveOnAcknowledgementEventListener(OnAcknowledgementEventListener subscriber
)
```

```
public interface OnAcknowledgementEventListener {
    public void OnAcknowledgement(String messageID, AID from);
}
```

#### *Event subscriber interfaces to implement*

`OnAcknowledgementEventListener`: This is the interface a class has to implement to be able to receive `AcknowledgementNotifications` for sent messages requesting an acknowledgement answer delivered by the `MessageClient` class which is connected to the `MessageRouting` servers.

#### *Implemented Method Parameters*

`messageID`: The ID of the message that was acknowledged by the receiver.

`from`: An AID (ADVENTURE ID) from whom the message acknowledgement was sent.

#### *Implemented Method Return Value*

none

#### *Remarks*

The `OnAcknowledgement` method gets called on all event subscribers implementing the `OnAcknowledgementEventListener` interface that were added to the list of subscribers via `AddOnAcknowledgementEventListener` when the requested acknowledgement for a sent message

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>147</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

comes in. The acknowledgement will not trigger the signaling of the OnMessage Event.

#### 3.4.4.1.1.5 Event OnError(Exception exception)

```
public void AddOnErrorEventListener(OnErrorListener subscriber)
```

```
public void RemoveOnErrorEventListener(OnErrorListener subscriber)
```

```
public interface OnErrorEventListener {
    public void OnError(Exception exception);
}
```

#### *Event subscriber interfaces to implement*

OnErrorEventListener: This is the interface a class has to implement to be able to receive Exceptions when something unexpected happens.

#### *Implemented Method Parameters*

exception: The Exception object describing the Error, possibly also giving additional information as to how to handle the problem.

#### *Implemented Method Return Value*

none

#### *Remarks*

The OnError method gets called on all event subscribers implementing the OnErrorEventListener interface that were added to the list of subscribers via AddOnErrorEventListener when an error occurs.

#### 3.4.4.1.1.6 Event OnAuthError(XmlElement exceptionElement)

```
public void AddOnAuthErrorEventListener( OnAuthErrorListener subscriber)
```

```
public void RemoveOnAuthErrorEventListener( OnAuthErrorListener subscriber)
```

```
public interface OnAuthErrorEventListener {
```



```
public void OnAuthError(XmlElement exceptionXmlElement);
}
```

#### *Event subscriber interfaces to implement*

**OnAuthErrorEventListener:** This is the interface a class has to implement to be able to receive authentication exceptions, when the authentication information which consists of username and password are not valid.

#### *Implemented Method Parameters*

**exceptionXmlElement:** The received XElement describing the Error, possibly also giving additional information as to how to handle the problem.

#### *Implemented Method Return Value*

none

#### *Remarks*

The OnAuthError method gets called on all event subscribers implementing the OnAuthErrorEventListener interface that were added to the list of subscribers via AddOnAuthErrorEventListener when an authentication error occurs. This should happen only right after the connect method on the MessageClient was called.

#### **3.4.4.1.1.7 Event OnDisconnect()**

```
public void AddOnDisconnectEventListener( OnMessageListener subscriber)
```

```
public void RemoveOnDisconnectEventListener( OnMessageListener subscriber )
```

```
public interface OnMDisconnectEventListener {
    public void OnDisconnect();
}
```

#### *Event subscriber interfaces to implement*

**OnDisconnectEventListener:** This is the interface a class has to implement to be able to get notified when the connection was

disconnected, for example when the server is unavailable or the internet connection is down.

#### *Implemented Method Parameters*

none

#### *Implemented Method Return Value*

none

#### *Remarks*

The OnDisconnect method gets called on all event subscribers implementing the OnDisconnectEventListener interface that were added to the list of subscribers via AddOnDisconnectEventListener when an authentication error occurs.

#### **3.4.4.1.1.8 JabberClient GetJabberClient()**

```
public JabberClient GetJabberClient()
```

#### *Parameters*

none

#### *Return Value*

JabberClient: The original client of the wrapped API is returned, which offers more detailed APIs that were not wrapped.

#### *Remarks*

Many details about the wrapped library might have to be changed by the using component, therefore direct access to the underlying libraries implementation of the connection has to be accessible for the users of the client wrapper, in case something unforeseen is needed from the wrapped API. Using this method should not be necessary, as all needed functionality is provided by the MessageClient API.

**3.4.4.1.2 Class AdventureMessage***AdventureMessage Constructor*

```
public AdventureMessage(XMPPMessage xmppMessage);
```

*Parameters*

xmppMessage: The wrapped library's implementation of the XMPP Message object. The other components won't need this constructor as it's only used by the MessageClient.

*Remarks*

The other components will never need this constructor, as the AdventureMessages will be created in the SendMessage and SendMessageSych methods. Incoming XMPPMessages will always be wrapped into an AdventureMessage by the event that delivers the message.

```
public String getMessageBody();
```

*Parameters*

none

*Return Value*

none

*Remarks*

This will contain the Message body sent to the component. Usually, the component can deserialize the string with its self-defined XSD to generate an object to handle the message.

```
public bytes[] getFileAttachment(String key = null);
```

*Parameters*

The key of the attachment, if null is passed to the method the first attachment is returned.

*Return Value*

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>151</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The binary file attached to the message, or null if there is no binary attachment.

#### Remarks

The return type could be changed if there is a way to return the file with increased performance than returning the byte array.

```
public void setFileAttachment(DataHandler dataHandler, String key = null);
```

#### Parameters

dataHandler: A DataHandler that represents the bytes of the attached file

key: The key, under which the file will be appended to the message. If null is passed to this method, a random key will be generated. Default: null.

#### Return Value

none

#### Remarks

This method might be subject to change if a better way can be found to include a binary file performance-wise.

```
public AID getReceiver();
```

#### Parameters

none

#### Return Value

AID: The AID of the receiver.

#### Remarks

none

```
public AID getSender();
```

#### Parameters

none

#### Return Value

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>152</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

AID: The AID of the sender.

*Remarks*

none

```
public String getMessageId();
```

*Parameters*

none

*Return Value*

The message id of the message.

*Remarks*

none

```
public String getThreadId();
```

*Parameters*

none

*Return Value*

The ID of the message exchange.

*Remarks*

This is used to identify if this message is an answer to another message or is used by the receiver to answer a message.

```
public DateTime getReceived();
```

*Parameters*

none

*Return Value*

The Timestamp when the message was received.

*Remarks*

none

```
public DateTime getSent();
```

*Parameters*

none

*Return Value*

A DateTime-stamp when the message was sent.

*Remarks*

This will contain the Message body sent to the component. Usually, the component can deserialize the string with its self-defined XSD to generate an object to handle the message.

```
public boolean getAcknowledgementWanted();
```

*Parameters*

none

*Return Value*

True if an acknowledgement message is wanted, false otherwise.

*Remarks*

none

```
private String setAcknowledgementWanted(boolean acknowledgementWanted);
```

*Parameters*

acknowledgementWanted: Is set when the receiver should answer with an acknowledgement.

*Return Value*

none

*Remarks*

This will be set by the MessageClient.

```
public DateTime getAcknowledgementReceivedTime();
```

*Parameters*

none

*Return Value*

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>154</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The time when the acknowledgement was received. Null if no acknowledgement was received.

#### Remarks

none

```
public String serialize();
```

#### Parameters

none

#### Return Value

Returns the necessary XML metadata to transfer over XMPP including the message body as a String.

#### Remarks

Used internally by MessageClient.

```
public AdventureMessage deserialize(String serializedAdventureMessage);
```

#### Parameters

none

#### Return Value

Returns the necessary XML metadata to transfer over XMPP including the message body as a String.

#### Remarks

Used internally by MessageClient.

### 3.4.4.1.3 Class AID

This class extends the JID class from the wrapped libraries, which consists of username, serverUrl and resource identifier. As the serverUrl is known, only username and resource identifier are needed.

#### AID Constructor

```
public AID(String username, String resourceId = null,  
           String serverUrl = null);
```

*Parameters*

username: The username issued for the messaging account.

resourceId: A unique identifier for a specific instance of one account. For example, several instances of Cloud Storage could have the same username and password, but use different resourceIds. If then a message would be sent to the username of Cloud Storage, it would be determined automatically to which instance the message would be routed unless a specific instance ID is targeted. When not specified

serverUrl: If another URL than the known URLs for the ADVENTURE Message Routing should be used, it can be set as a parameter. Default: null.

*Remarks*

The constructor creates a AID instance, which extends JID properly so that the AID can be used with the original JabberClient returned by the GetJabberClient method of the MessageClient class instance.

```
public void toString();
```

*Parameters*

none

*Return Value*

Returns the String value that also would be generated by the JID class instance of the wrapped library, e. g. "username@adventure-fp7.eu/instanceid"

*Remarks*

A convenience class.



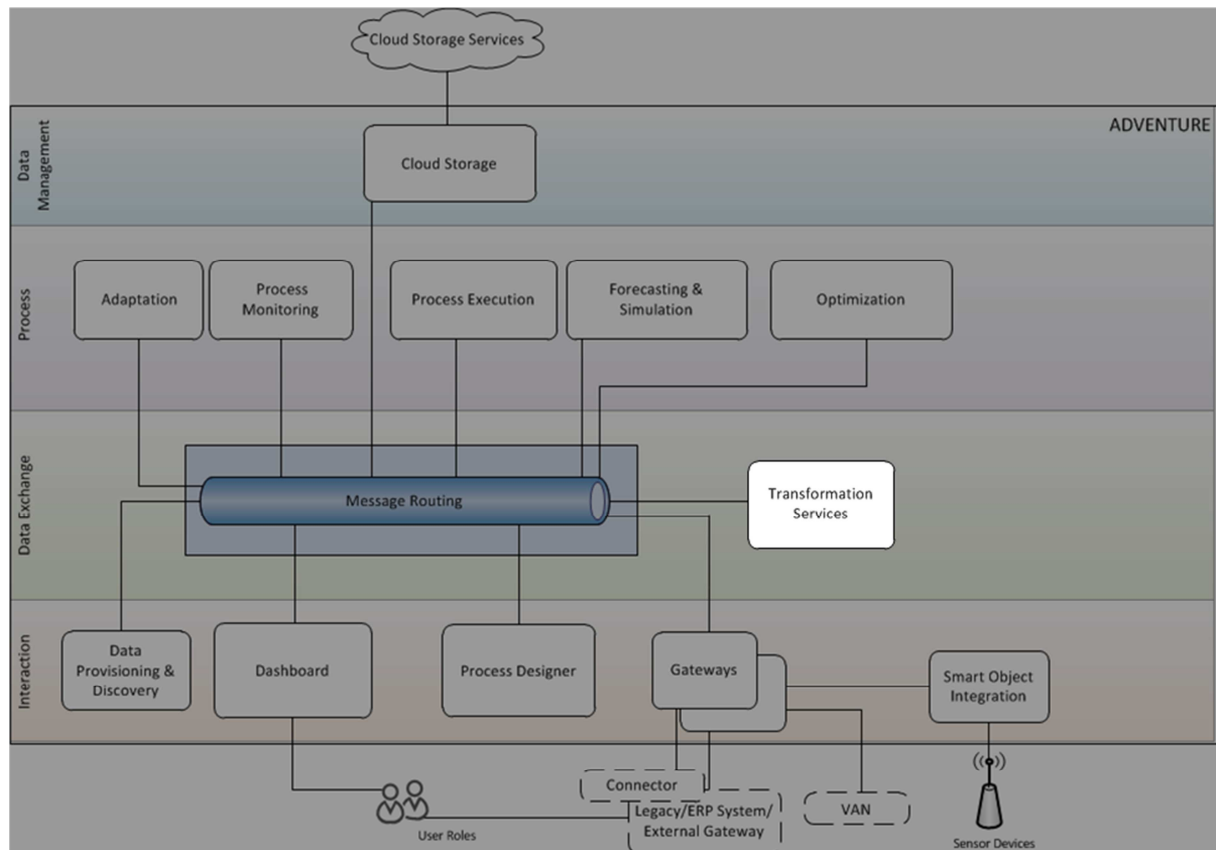
### 3.4.5 Summary

The Message Routing component addresses the problem of having multiple components connected in a scalable way with high performance and a simple way to be used by the other components.

The above problem will be solved by using XMPP, which is a simple and fast infrastructure that is heavily used by most messaging solutions available today. Open Source servers and client libraries are available for the technology, and are wrapped by a MessageClient library for the other components, providing a minimalistic API.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>157</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

# Transformation Services



## 3.5 Transformation Service

Abstract:

The Transformation Service in the ADVENTURE Platform is responsible for transforming information between different data content formats.

The transformation engine makes use of mapping files in order to perform its task. These mapping files will be executed by an external translation engine to perform the transformation.

Partner TIE translation engine will be customized to implement the transformation services in ADVENTURE.

### 3.5.1 Major Design Decisions

ADVENTURE is not a project to develop a new Transformation Service/Engine which is a multi-year task, but Transformation is needed to enable the contents necessary between ADVENTURE endpoints. Partner TIEs core business is related to data exchange and transformation within B2B environments and as such the implementation of the Translation Services will be based on TIE commercial products with adaptations necessary to fulfill ADVENTUREs precise needs, which is one of the reasons for TIEs inclusion in the project. They will be adapted in such a way that an open interface will be described to interact with them thus making these commercial products (implementing core translation functionality) exchangeable in the future.

A transformation engine is usually based on mapping files. A mapping file describes the syntax to be executed to transform a specific format A into format B. Mapping files commands and formats depends on the translation engine that has to execute them. Having this scenario in mind, a specific TIE external mapping editor will be used to edit or create the specific mapping syntax of the transformations to be used by the transformation engine.

To execute a transformation operation, it is necessary to have different types of information available to the system. This information usually describes the inputs and outputs formats and is used by the transformation engine to check the validity of the information provided as well as the transformation itself. In order to link this information to a specific translation operation, and its specific mapping file, ADVENTURE will provide a UI that company brokers can use in order to describe their specific transformation operations thus providing the system all the information

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>159</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

that the transformation engine will need at the moment of the transformation execution.

A Transformation operation will normally be used in order to transform the output information of an activity at a process model (during runtime) into the format expected by the next activity as input. This transformation operation can also be part of the data recovery/sending from/to a third party software through the gateways. The transformation services will be exposed at the Process Designer either as specific modeling (graphical) elements or will be set up as part of the business activities configuration (see Figure 66 - New transformation activity sequence diagram), so that a company's broker can use them within a virtual factory design and configuration. In the architecture document, transformations were directly linked to gateways, and an implicit transformation feature was stated. In order to make the transformation operations available in a more generic way, the transformation services might be also declared explicitly by means of an appropriate element at the process designer. In any case, the possibility of linking a gateway operation with a transformation continues to exist and will be part of the gateways configuration via the process designer.

In order to make the transformation service available to all the different brokers, it will be published at the ADVENTURE framework as a standard component, linked to the Message Routing functionality. In this way it will be accessible to every ADVENTURE component (especially Smart Process Execution and Gateways). When calling a transformation operation in the transformation service, the mapping and needed auxiliary files will be recovered from the Cloud Storage system and provided to the external translation engine. This can then execute the desired transformation successfully.

The transformation operation descriptions provided by the brokers will be stored within the Cloud Storage system in a noSQL bucket. All the necessary auxiliary files will also be stored in the Cloud Storage System within a binary data bucket. All this information will be created / updated by means of the company transformation operation description UI and recovered when the transformation has to be executed.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>160</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### 3.5.2 Technology Comparison and Selection

This subsection will describe the criteria to be taken into account for the creation of the Transformation service component, and the adaptation of existing translation technologies so that they can be used in ADVENTURE. In this sense this differs from the other sections since the choice is pre-defined.

#### 3.5.2.1 Selection Criteria

The selection criteria for Transformation Services technologies were defined in the *D3.2 Functional Specification* in section 5.6.5. The description of the generic parameters can be found in the *D3.2 Functional Specification* in section 5 on page 28. Now follows the description of the Transformation Services specific parameters:

**Transformation based on standards:** Describes the preferences in terms of basing the transformation definitions on transformation standards where available, e.g. XSLT for XML. This can result in some performance loss and functionality restrictions, but in return, allows using standard technologies, that have their own mapping mechanism (like XSLT). In fact, XSLT is the only such standard operating only on XML.

**XML supported:** The Transformation Service will support transformations for XML files, meaning that XML files can be used as source, destination, or both in transformation operations. It is expected that ADVENTURE focus will be on XML data types

**CSV supported:** The Transformation Service will support transformations for CSV-like formats (flat text files, with specific separation between fields), meaning that CSV-like files can be used as source, destination, or both in transformation operations. CSV files are a quite usual format for data extraction from 3rd party systems (especially in SMEs world). Note that here CSV covers a wide flat file concept and not just 'comma' separated 'data rows'.

**EDI supported:** The Transformation Service will support transformations for EDI like formats (e.g. EDIFACT, Tradacoms, Odette...), meaning that EDI messages can be used as source, destination, or both in transformation operations. EDI is the most widely used protocol and data file format in the B2B/ERP application area.

**Database supported:** The Transformation Service will support access to Databases as data sources, destination, or both. In complex transformation scenarios,

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>161</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Databases can also be used as intermediate steps to recover needed data to add or check.

**Semantic annotation of input/output formats:** The ADVENTURE broker will be able to annotate semantically the different information describing a Transformation Service operation. This semantic annotation of information will be provided by means of common taxonomies or vocabularies.

### 3.5.2.2 Possible Technologies

Partner TIE is working in data transformation for B2B systems integration for over 25 years, having different products and technologies dealing with this specific topic. Currently these different technologies have been integrated into TIEs latest common framework for data transformation and message delivery on B2B environments - TIE SmartBridge (TSB). Major efforts in the development of TSB have been focused on performance, stability, reliability, and extensibility through configuration of transformation mechanism and mapping file description and thus providing a stable basis for ADVENTURE.

TSB is a B2B message exchanging broker that is able to transport a *B2B XML*, EDI or other syntactical message from a source point to its destination, performing the adequate message transformations and routing. The main objective of TSB is to assure the correct delivery of the B2B messages to the end point, thus implementing high reliability mechanisms that allow disaster recovery. At this moment, TIE solutions for data integration and transformation are being used by more than 3000 companies (a large percentage of them are SMEs) in order to afford delivery of XML and EDI based messages between business partners.

The Translation engine will be based on TIE's product TSB that integrates different transformation technologies and strategies. The technology selection criteria described in D32 is fulfilled by TSB. No further comparison of transformation engines will be performed since this is the rationale for TIE in the related Task T4.4.

The TSB generic mapping editor will be used in order to allow users to develop and/or edit the needed mapping files.

TSB needs adaptations to be used in ADVENTURE project (interface implementation, internal data flow definitions, etc.). The interfaces between ADVENTURE and TSB will be defined in an open way, so that additional translation

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>162</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

engines can be used to substitute TSB in the future, giving the partners the flexibility of being non-dependents on this specific commercial product.

### 3.5.2.3 Technology Selection

TSB will be used as core for the transformation services.

### 3.5.2.4 Missing Elements and Implementation Needs

TSB covers both parts of map editor and of Translation engine. Relevant functionalities in ADVENTURE need to be developed in order to make these external elements to work in a coordinated and integrated way with the rest of the ADVENTURE platform. Main missing elements are:

Communication with the ADVENTURE platform.

Description of the transformation operations and storing of mapping and auxiliary files into the Cloud Storage System.

Link to the DPD component, in order to use the semantics annotation functionalities provided for annotating the different elements of the transformation operations description.

Retrieval of information needed to execute a transformation. Based on the operation information sent to the transformation service, it has to be able to find and retrieve the necessary information (mapping files, etc...) from the Cloud Storage system in order to provide it to translation engine, so that it can do its work successfully.

In order to send the retrieved information to TSB, an open API and communication protocol has to be defined. Each time the Transformation service will be invoked, all the auxiliary data for the transformation has to be retrieved and packed in the format defined by this API, so that TSB (or other Transformation Service) could understand it. Depending on the deployment scenario, the external translator engine could be reached locally, or could be installed somewhere in the cloud (this will be the most common deployment scenario). A communication protocol and parameter passing format needs to be defined and implemented to make such integration possible.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>163</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

3.5.3 Technical Component Specification

3.5.3.1 Component Structure

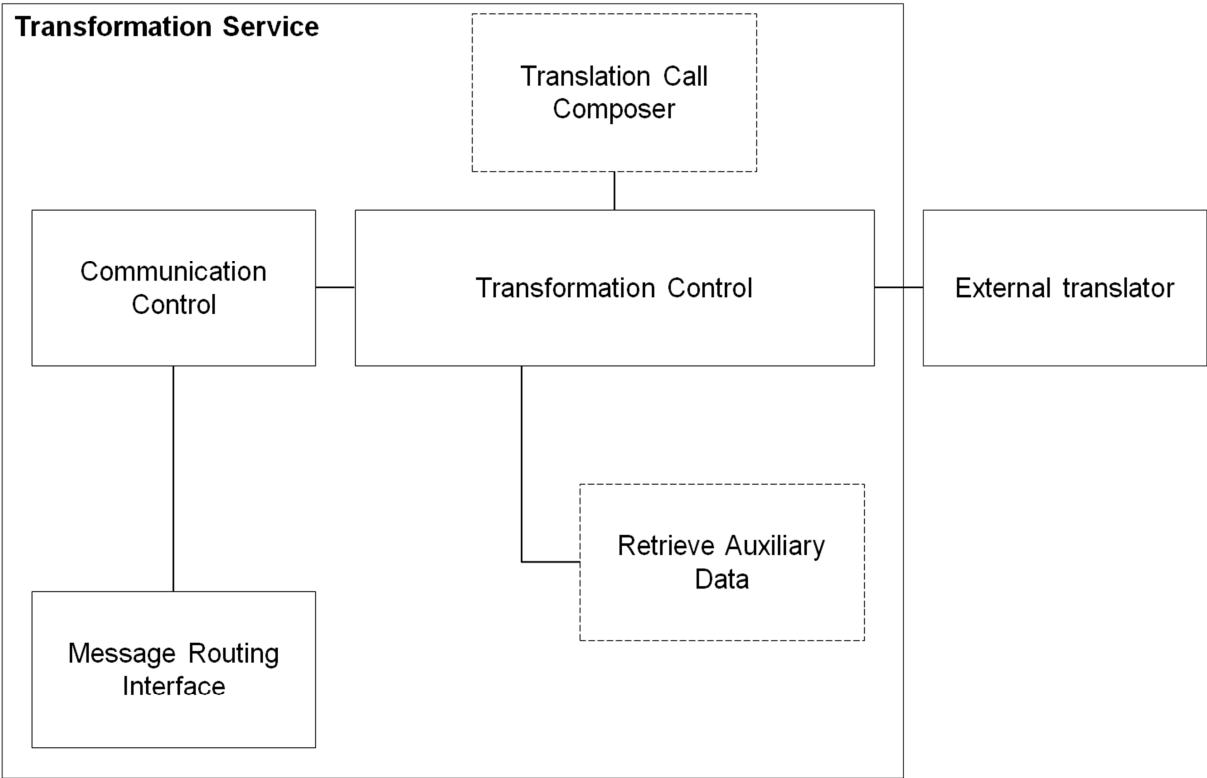


Figure 28 - Transformation Service component structure

The Transformation Service will consist of the following parts:

Message Routing Interface

This implements the Message Routing API and provides communication with ADVENTURE. An event handler will be triggered if a message is received and Communication control element starts to handle the request.

Communication Control

The communication control handles the different requests/responses that are sent to/from the ADVENTURE platform.



## Transformation Control

The transformation control is the software element that, based on the specific operation requested, will retrieve all the additional data necessary (mapping file, input format description files, etc...) to provide it to the translation engine. In some cases the transformation service is called with not all the parameters necessary to recover the information (e.g. if the transformation target is in some internal format). In this case the transformation control will search for the appropriate destination format, recover the necessary information, and provide it to the translation engine.

The transformation control will also implement an API and communication protocol so that an external translation engine could be interfaced which adopted this API and allow it to perform the translation operation. By means of the API all the necessary information (mapping files, source and destination formats, etc...) will be provided to the external translator software engine.

## Retrieve Auxiliary Data

Component used by the Transformation Control. It retrieves all the information associated to a specific transformation from the Cloud Storage (mapping file, input format description files, etc...).

## Translation Call Composer

Auxiliary component used by the Transformation Control. It prepares the data structures to be included in the translation calls based in the External Translator API definition.

## External Translator

The external translator is implemented by an instance of TSB, configured to act as a translator engine. It will receive a message or call, providing it with all the necessary information. It will return the translated data. If an error occurs, then it will trigger an exception through the Translator interface. The Communication control will deal with the exceptions and return the appropriate message via the Message Routing component.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>165</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

3.5.3.2 Internal Sequence Diagrams

This section will give an insight about the internal component communication. It is extracted from the use case scenarios of D3.2.

Transform

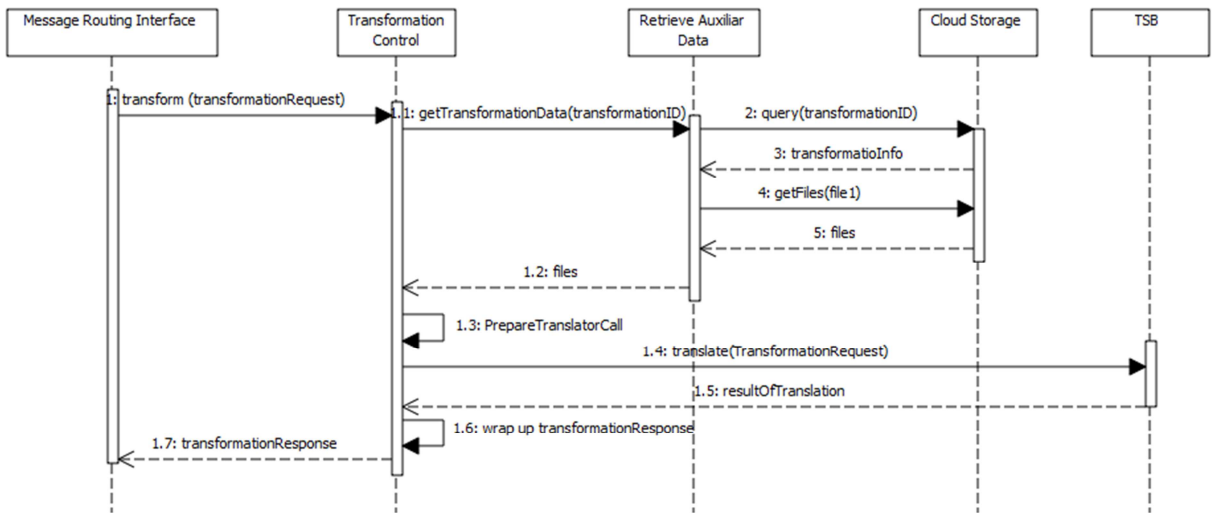


Figure 29 - Transformation internal sequence diagram

When a transformation request is received, the Transformation Control is the responsible of receiving the message. The Transformation Control will then call the functionalities that will interact with the Cloud Storage in order to retrieve all the information describing the transformation action requested. This is done in two different steps:

In the first step all the information describing the transformation operation is retrieved (metadata).

In the second one, and based on the results of the first step, the files containing the transformation format descriptions and the mapping files are also retrieved.

With all this information and the source data to be transformed, a call to the TSB translation interface is performed. TSB will return the result of this operation (the data in the new format, or an error). Finally, the result is wrapped in the format of a Transformation Service call return message, and is delivered to the Message Routing component, that will return the result to the caller component.

3.5.4 Specification of Interfaces, Protocols and Formats

To communicate with other ADVENTURE components the Transformation Service uses the Message Routing API. The interfaces will be called using commands (XML based) within the message. The following section describes the internal API specification that will be used after the interpretation of the message.

3.5.4.1 API specification

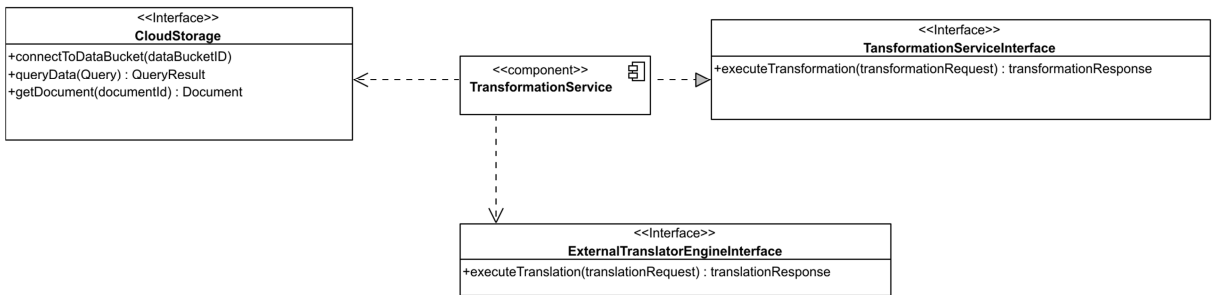


Figure 30 - Transformation Service interfaces (provided and needed)

executeTransformation

This method will allow the different ADVENTURE components to invoke a transformation action in the Transformation Service.

```
public TransformationResponse executeTransformation(
    String transformationRequest)
```

Parameters

transformationRequest: XML structure containing the specific details of the operation to be performed. The specification of the contents of this XML structure can be found in the Content Formats and Protocol Definitions section.

Return Value

TransformationResponse: XML Structure containing the results of the operation. The specification of the contents of this XML structure can be found in the Content Formats and Protocol Definitions section.

Message-Example

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:transformationRequest xmlns:tns="http:// www.fp7-adventure.eu/xmlSchema/TransformationS/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http:// www.fp7-
adventure.eu/xmlSchema/TransformationS/TransformationReq.xsd ">
  <tns:header>
    <tns:transformationID>TransformationID</tns: transformationID >
  </tns:header>
  <tns:payload>
    <tns:sourceData> <![CDATA[INFOTOBETRANSFORMED]]></tns:sourceData >
  </tns:payload>
</tns:transformationRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:transformationResponse xmlns:tns="http:// www.fp7-
adventure.eu/xmlSchema/TransformationS/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http:// www.fp7-
adventure.eu/xmlSchema/TransformationS/TransformationRes.xsd ">
  <tns:header>
    <tns:commandResult>OK</tns:commandResult >
  </tns:header>
  <tns:payload>
    <tns:resultData> <![CDATA[TRANSFORMEDINFO]]></tns:resultData >
  </tns:payload>
</tns:transformationResponse>
```

executeTranslation

This method will be send to the external translation engine, so that it can execute the translation action, by using the provided mapping file, source data, and auxiliary information provided at the method call

```
public TranslationResponse executeTranslation (
    TranslationRequest translationRequest)
```

Parameters

- translationRequest: XML structure containing the specific details of the transformation to be performed by the external translator engine. The specification of the contents of this XML structure can be found in the Content Formats and Protocol Definitions section.
- callerID: ID of the ADVENTURE caller element.

### Return Value

TransformationResponse: XML Structure containing the results of the operation. The specification of the contents of this XML structure can be found in the Content Formats and Protocol Definitions section.

### Message-Example

#### Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:translationRequest xmlns:tns="http://www.fp7-adventure.eu/xmlSchema/TransformationS/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/TransformationS/TranslationReq.xsd">
  <tns:header>
    <tns:commandID>CommandIdForTSB</tns:commandID>
    <tns:sender>BrokerXID</tns:sender>
    <tns:recipient>BrokerYID</tns:recipient>
    <tns:translationEngine>TSI</tns:translationEngine>
  </tns:header>
  <tns:payload>
    <tns:sourceData encoding="base64">
      <![CDATA[INFO_TO_BE_TRANSFORMED_ENCODED]]>
    </tns:sourceData>
    <tns:sourceDefinition> <![CDATA[SOURCE_DESC]]></tns:sourceDefinition>
    <tns:mappingData> <![CDATA[MAPPING_DATA]]></tns:mappingData>
    <tns:destinationDefinition>
      <![CDATA[DEST_DESC]]>
    </tns:destinationDefinition>
  </tns:payload>
</tns:translationRequest>
```

#### Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:translationResponse xmlns:tns="http://www.fp7-adventure.eu/xmlSchema/TransformationS/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/TransformationS/TranslationRes.xsd">
  <tns:header>
    <tns:commandID>CommandIdAtTSB</tns:commandID>
    <tns:sender>BrokerXID</tns:sender>
    <tns:recipient>BrokerYID</tns:recipient>
    <tns:commandResult>OK</tns:commandResult>
  </tns:header>
  <tns:payload>
    <tns:resultData> <![CDATA[TRANSFORMEDINFO]]></tns:resultData>
  </tns:payload>
</tns:translationResponse>
```

### 3.5.4.2 Content Formats and Protocol Definitions

An XML document will be used to transfer data to the TSB translator engine via the Translator interface. The XML-schema shown below contains all the necessary parameters for invoking the API method specified in the sections above.

#### Transformation Service message format XML/Schemas

<http://www.fp7-adventure.eu/xmlSchema/TransformationService/TSReq.xsd>

<http://www.fp7-adventure.eu/xmlSchema/TransformationService/TSResp.xsd>

#### Request-Format:

An XML document will be used to transfer data from an ADVENTURE component to the Transformation Service, by means of the Message Routing functionality. Messages following the XML-schema shown below, will be transferred as payload of the Message Routing call, and contains all needed parameters for invoking the API method specified in the sections above.

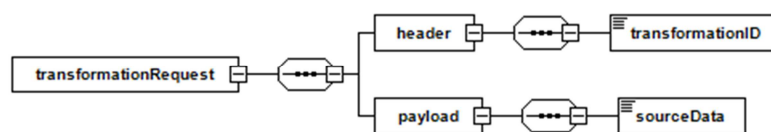


Figure 31 – Transformation request XML schema

#### Data elements

**transformationID:** Unique identifier of the transformation definition to be performed. This ID will be used to retrieve the details and auxiliary files needed to perform the transformation by TSB.

**sourceData:** Data to be transformed into the required format. This data structure contains a name attribute, and an encoding attribute, describing if the value data is encoded (base64) or not. In the case of big files, gateways will use the Message Routing features for big attachments, that will allow to temporary store the file, and use a URI reference (URI - Universal Reference Identifier) into the source data parameter, so that the big file could be retrieved just in the moment of using it.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>170</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

*Additional information collected from the incoming Message Routing call.*

- componentID: Identifier of the caller component, so that the transformation result can be routed to it.
- from: Identification of the sender user (or company).
- to: Identification of the recipient user (or company).

**Response-Format:**

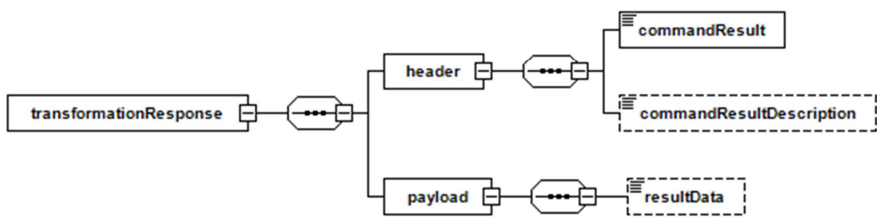


Figure 32 - Transformation response schema

*Data elements*

- commandResult: Result of the command executed (OK, ERROR)
- commandResultDescription: If error, this field describes the cause of the error, so that the upper process can be informed.
- resultData: Data transformed into the required format. it contains a name descriptor and an attribute describing if the value is encoded or not. In the case of big files, gateways will use the Message Routing features for big attachments, that will allow to temporary store the file, and use a URI reference (URI - Universal Reference Identifier) into the result data parameter, so that the big file could be retrieved just in the moment of using it.

*Additional information embedded in the Message Routing message response.*

- commandID: Unique Identifier for the current transformation operation, it can be used to access log of operation.
- from: Identification of the sender user (or company).
- to: Identification of the recipient user (or company).

Translator interface message format XML/Schemas

<http://www.fp7-adventure.eu/xmlSchema/TransformationService/TranslatorReq.xsd>

<http://www.fp7-adventure.eu/xmlSchema/TransformationService/TranslatorResp.xsd>

The translator interface defines the data structures that are to be exchanged between the Transformation Services component and the external translator engine (i.e. TIE SmartBridge).

Request-Format:

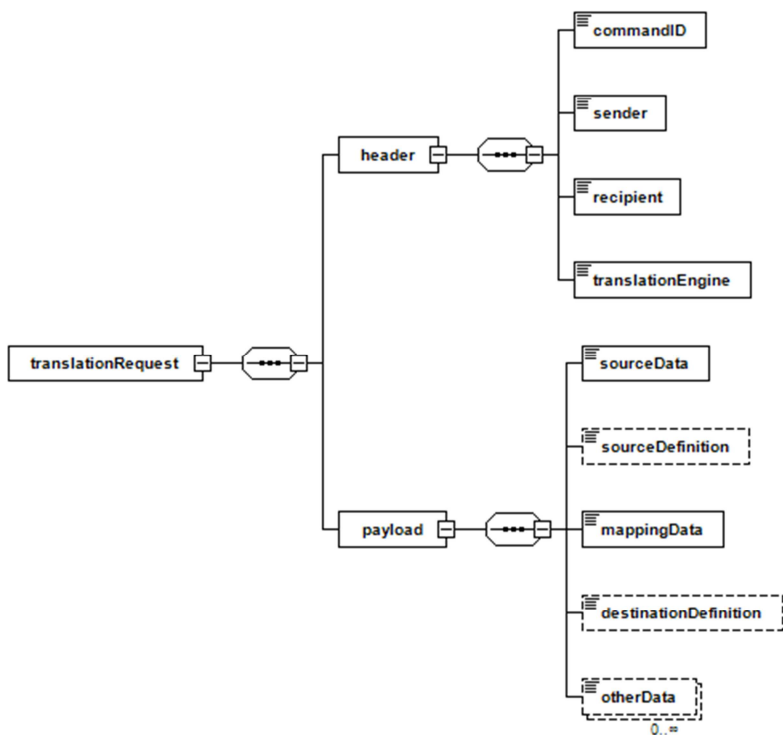


Figure 33 – Translator interface request schema

Data elements

- commandID: Unique Identifier for the current transformation operation.
- sender: Identification of the sender user (or company).
- recipient: Identification of the recipient user (or company).
- translationEngine: Unique identifier of the specific translation engine algorithm (a translation engine can implement more than one) to be used for the transformation. In case this data is not provided, the



translation engine will try to guess which type of it has to be used is based on the source and destination formats.

sourceData: Data to be transformed into the required format. Contains attributes for name and encoding.

sourceDefinition: Technical Description of the format of the sourceData. Contains attributes for name and encoding.

mappingData: Data describing the actions to be performed to transform the sourceData into the destination format. Contains attributes for name and encoding.

destinationDefinition: Technical Description of the destination format. Contains attributes for name and encoding.

otherData: Depending on the kind of transformation to be performed, additional data can be needed. This field will allow auxiliary data to be sent to the translation engine to be used during transformation, i.e. if some specific logo has to be embedded into the resultant file, it will be provided via the otherData parameter. Contains attributes for name and encoding.

Response-Format:

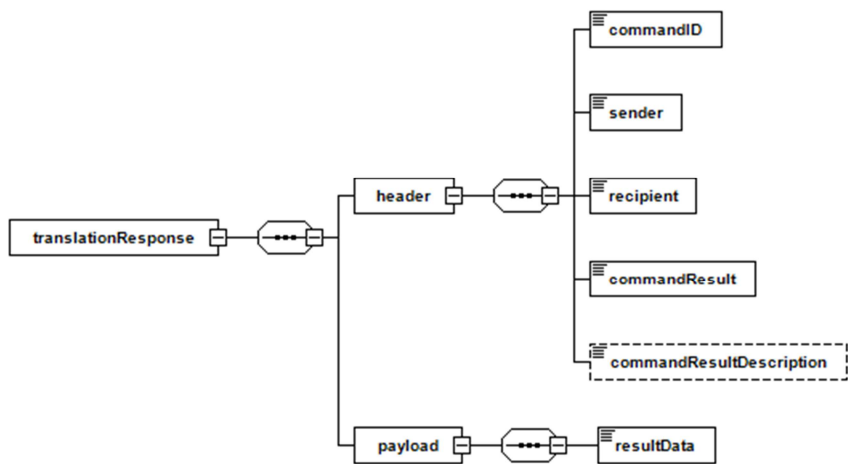


Figure 34 - Translator Interface response schema

Data elements

- commandID: Unique Identifier for the current transformation operation.
- sender: Identification of the sender user (or company).
- recipient: Identification of the recipient user (or company).
- commandResult: Result of the command executed (OK, ERROR)
- commandResultDescription: If error, this field describes the cause of the error, so that the upper process can be informed.
- resultData: Data transformed into the required format. Contains attributes for name and encoding (if base64).

Cloud Storage data formats

In order to describe and store the transformation operations definition data model below will be used. NoSQL buckets will be used to store the descriptive data. This descriptive data will make reference to specific stored binary files that will describe the source and destination formats, the mapping files, and other auxiliary files if necessary.

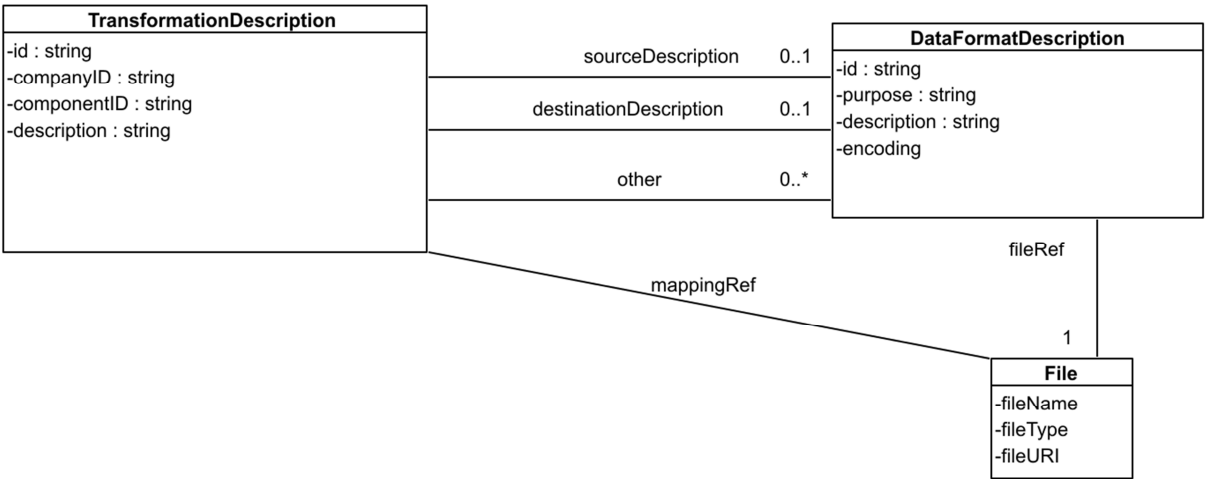


Figure 35 - Transformation operation description object model

The transformation description data will be used by the Process Designer when configuring a data transformation element. This same data will be also used by the Transformation engine itself when called, because it needs to retrieve all auxiliary

data from the Cloud Storage component to send it to the external translator engine (TSB).

#### *TransformationDescription*

id: Unique Identifier for the transformation description.

companyID: Identification of the Broker that has set up and used the transformation.

componentID: Id of the component (Transformation service) that will execute the transformation. Several transformation services can exist in the ADVENTURE platform.

description: Description of the transformation operation.

mappingREF: Reference to the file containing the mapping data in the Cloud Storage System.

#### *DataFormatDescription*

id: Unique Identifier of the data format.

purpose: Classification of the purpose of using the data format (e.g. orderStatus retrieval).

description: Textual description of the data format.

fileRef: Reference to the file containing the mapping data in the Cloud Storage System.

#### *File*

*fileName: Name of the file.*

*filetype: Type of the file*

*fileURI: URI to recover the file at AVENTURE Storage System.*

### 3.5.5 Data Extraction Service

The data extraction service is a component that extends the Transformation Service in order to allow consistent content extraction to be used internally by the different ADVENTURE components.

#### Abstract:

The Data Extraction Service is based on the ADVENTURE Transformation Service, and provides data (business data) content extraction from the different information formats that flows on ADVENTURE.

The Data Extraction Service output will always be a generic data format (summary), where information is semantically characterized.

The Data Extraction service will save the extracted information into the Cloud Storage System, so that it will afterwards be available for the different components dealing with business data (e.g. Dashboard or Monitoring).

#### 3.5.5.1 Major Design Decisions

ADVENTURE is a content agnostic platform, where the main functional elements work based on activities, processes, process status and events. This approach allows ADVENTURE to support any kind of process scenario without the constraints of prescriptive and ADVENTURE specific internal data formats.

However, on the other side, one of the objectives of the ADVENTURE is to be user friendly, and provide the final user with the kind of information that is understandable for them and close to their usual practices (orders, orderStatus, etc..) and the ability to store and later reuse this information (e.g. in visualization or analysis).

In order to harmonize these two visions, an auxiliary mechanism will be needed in ADVENTURE. This additional mechanism will be based on already existing ADVENTURE elements, ideally the existing Architecture and will be generic in its implementation. As an extension of the Transformation Service, the Data Extraction Service, will be used to declare transformations from proprietary data formats into a common (generic) one that the different ADVENTURE elements will be able to parse (in order to extract specific information from it).

The result of the data extraction will always have an established (proposed) XML generic format (characterized pair-value). This kind of content could be stored using

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>176</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

XML (or JSON) at a non-Relational bucket of the Cloud Storage System. In this way, any ADVENTURE component would be able to access this information (because they know what format to expect, and how to deal with it).

The proposed summary data model structure (because its purpose is not to contain all the data extracted from the original data element, i.e. document, but a subset of it) will support any kind of data extraction. In order to define the semantics of this summary data model, the elements contained will be linked to a common vocabulary (using like micro-formats, RDFa, etc), so that afterwards the elements could be aggregated by using this common characterization (e.g. list of orders based on orderNumber).

The Data Extraction Service will be able to be called from the custom connector of a Gateway. This will facilitate the flow of information between the different ADVENTURE models at execution time. ADVENTURE Gateways will be able to define Summary data model as the output of a gateway operation.

The definition of a Data Extraction service will be similar to the definition of a Transformation service. The main difference will be that the output of the Extraction service will always be a summary data format, and that special functionality will be added to describe the specific summary data elements, their semantic link, extraction paths, etc.

At execution time, content always comes from Gateways:

Data can be gathered from 3rd party systems as part of a call triggered from the Process Execution component, because the call is part of the designed process model in execution

A 3rd party system pushes a message into ADVENTURE through a Gateway (the Process Execution component manages it)

The data output of a Gateway operation will always be:

A document: EDI document, XML document, CSV document, or something else (which depends on the system linked to the Gateway implementation).

A summary information, defined as described above

Or, both.

Having that, there are three different possibilities to implements data extraction on ADVENTURE

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>177</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

An extraction service is defined as part of the designed process model. In this case, once a data document is recovered (as an output of a Gateway, or as an output of a transformation service), a call to the Extraction Service is performed to extract the summary information.

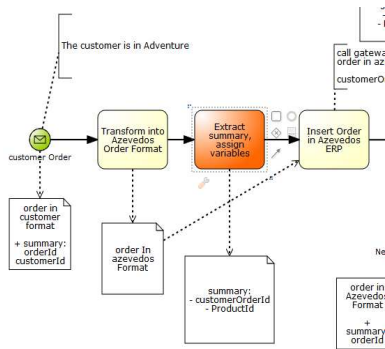


Figure 36 - An extraction service call is defined as part of the process model

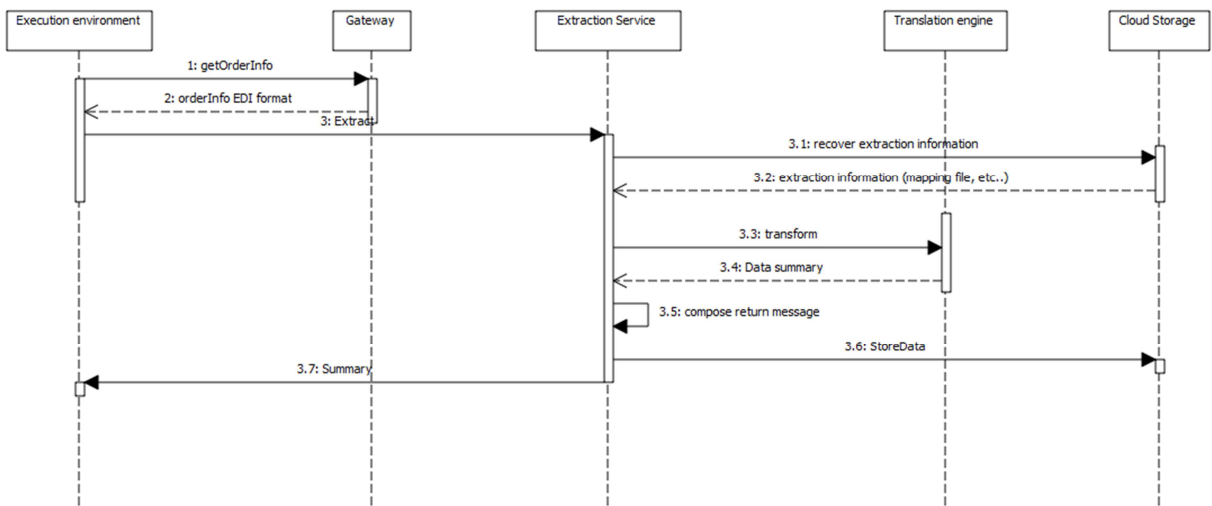


Figure 37 - Sequence Diagram - Data Extraction Service called from the execution environment after a gateway operation (as part of a designed process model)

The Gateway implements (internally) a call to a Content Extraction Service. As part of a Gateway implementation, the developed Gateway could decide to add a call to an extraction service, and then provide the summary data as part of the output of the Gateway. If a previous transformation is needed, the Transformation Service can also be called from the Gateway.

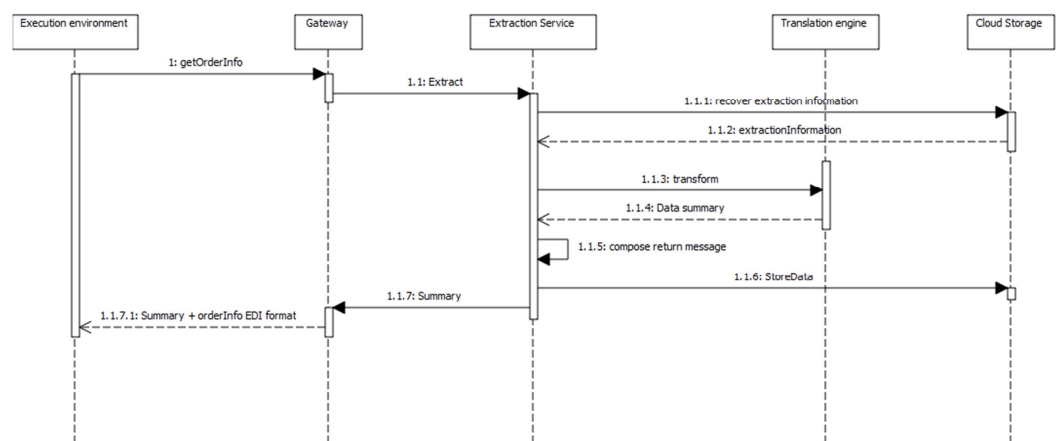


Figure 38 - Sequence Diagram - Data Extraction Service called from within a gateway operation

Summary extraction is hard coded as part of the Gateway implementation. In this case, the Gateway developer codes directly the summary information, without calling the data extraction service. This approach can be followed when the summary consists of only a small data set, and the content is easy to extract from the 3rd party system.

As the result of an extraction call, the input document (related to the Gateway or Transformation previous call) and/or the summary information extracted, will be available within the ADVENTURE system (the information will also be stored at the ADVENTURE Cloud Storage, so that it can afterwards be available for KPI calculation, or data representation at Monitoring or Dashboard). As per the predefined Execution Environment behavior, every time an Activity is executed, all the information about it is passed to the Monitoring functionality. Based on this information, this component will check the different rules against the data provided on the summary to know if some alert has to be sent (to the user or to the Dashboard). In this way the summary information will be available more dynamically for the monitoring component (alerts etc.) and others, mainly the Dashboard if the store flag is set (more static information for aggregation) so that it can offer more "business" related information to the final user, store this info into the logs of ADVENTURE, etc.

In any case, the summary description associated to a Transformation or Gateway operation doesn't mean that the operation will provide all the described parameters. In fact it will be only possible to provide these that will be available at the execution

time for this specific operation (meaning that the described parameters are not mandatory to be in the summary), e.g. if a transformation operation describes that it will return a summary of "orderNumber" and "orderStatus", but at runtime only orderNumber is available, then only this information will be available at the summary. This does not mean that the operation has failed, unless advanced functions to determine cardinality of information are required. If this information is needed afterwards, then special exception behavior need to be associated.

#### **3.5.5.1.1 Technology Comparison and Selection**

This component will be based on the functionalities of the Transformation Services, thus will use the same technologies.

##### **3.5.5.1.1.1 Selection Criteria**

See Transformation Service.

##### **3.5.5.1.1.2 Possible Technologies**

See Transformation Service.

##### **3.5.5.1.1.3 Technology Selection**

See Transformation Service.

##### **3.5.5.1.1.4 Missing Elements and Implementation Needs**

Main missing elements for the implementation of the Data Extraction Service, based on the Transformation Services are:

Development of the vocabulary that will support the characterization of the generic elements in the summary structure. This vocabulary will be integrated into the semantic facilities provided by the DPD, and will use its tools and interfaces.

Software libraries and interfaces will be developed to help the users to define, build and consume the summary information.

The summary information will be modeled as a type of Activity at the Process Designer.

#### **3.5.5.2 Technical Component Specification**

##### **3.5.5.2.1 Component Structure**

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>180</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



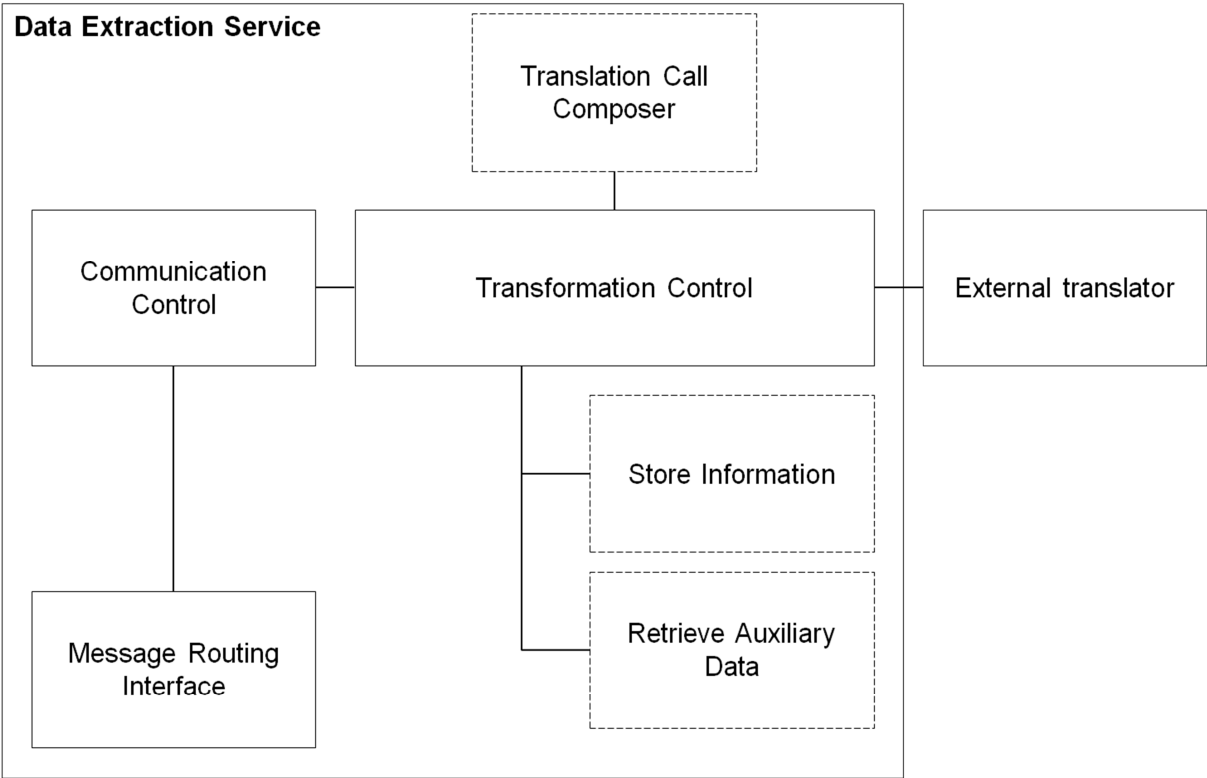


Figure 39 - Data extraction service component structure

The Data Extraction Service component structure is an extension of the Transformation Service structure and just adds the store summary information subcomponent, so only this specific part is described below. Please, see the Transformation Services component structure (3.5.3.1) for the description of the rest of the elements.

Store information

The Store Information subcomponent is used by the Transformation Control. It will save the summary information (the results of the translation action) into a non-relational data bucket at the Cloud Storage system, (if the option for storing the information has been configured by the user or service when setting up the service call).

3.5.5.2.2 Internal Sequence Diagram

This section will give an insight about the internal component communication. It is extracted from the use case scenarios of D3.2.

Extract

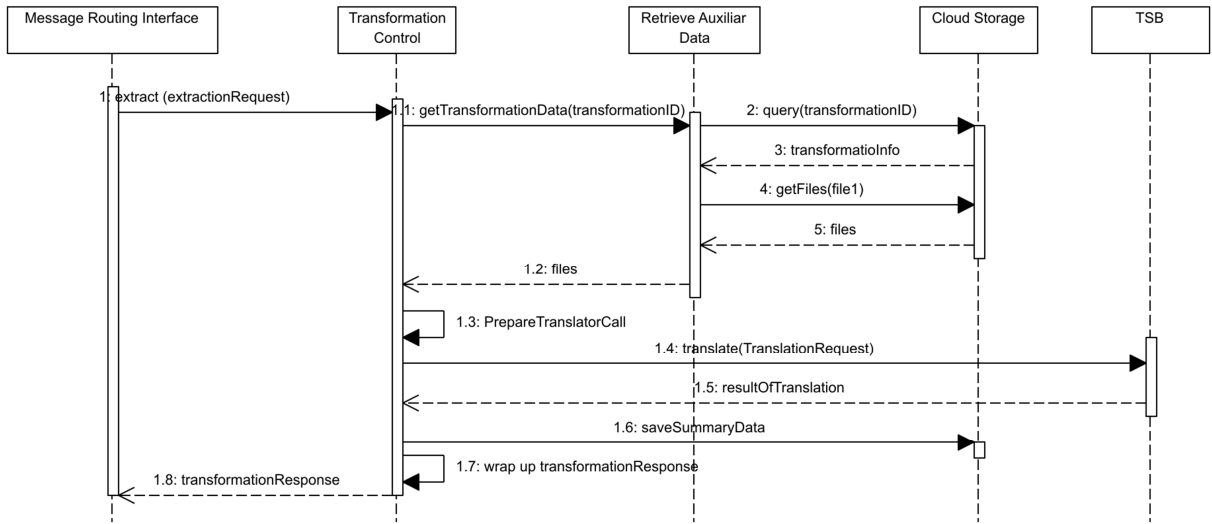


Figure 40 - Data Extraction internal sequence diagram

The content extraction sequence diagram is similar to the sequence diagram of a Translation Service (in which it is based on). The main difference, is that, once the Translation Engine (TSB) has returned the transformed data (in summary format), this data can be stored in the Cloud Storage (if the request call indicated to store the data).

3.5.5.3 Specification of Interfaces, Protocols and Formats

To communicate with other ADVENTURE components the Data Extraction Service uses the Message Routing API. The interfaces will be called using commands (XML based) within the message. The following section describes the internal API specification that will be used after the interpretation of the message.

3.5.5.3.1 API specification

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>182</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

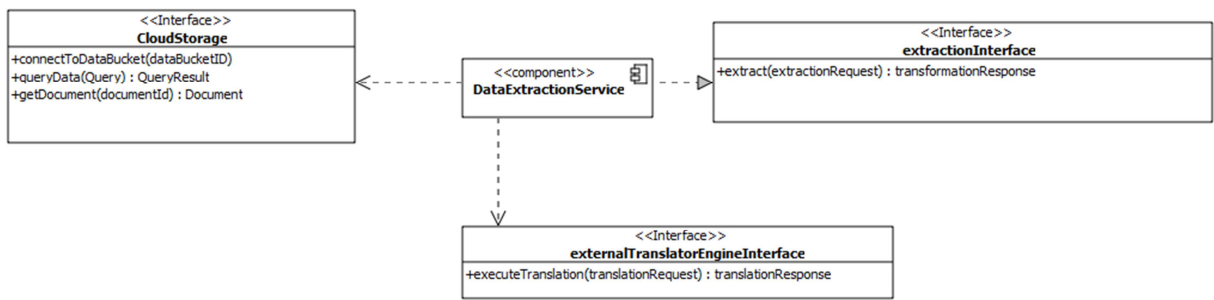


Figure 41 - Data extraction Service interfaces (provided and needed)

extract

This method will allow the different ADVENTURE components to invoke a transformation action in the Transformation Service.

```
public TransformationResponse extract(
    String extractionRequest)
```

Parameters

extractionRequest: XML structure containing the specific details of the operation to be performed. The specification of the contents of this XML structure can be found in the Content Formats and Protocol Definitions section.

Return Value

TransformationResponse: XML Structure containing the results of the operation. The specification of the contents of this XML structure can be found in the Content Formats and Protocol Definitions section.

Message-Example

Request:

<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;tns:extractionRequest xmlns:tns="http:// www.fp7-adventure.eu/xmlSchema/TransformationS/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http:// www.fp7- adventure.eu/xmlSchema/TransformationS/TransformationReq.xsd "&gt;   &lt;tns:header&gt;     &lt;tns:transformationID&gt;TransformationID&lt;/tns: transformationID &gt;     &lt;tns:save&gt;true&lt;/tns: save &gt;   &lt;/tns:header&gt;   &lt;tns:payload&gt;     &lt;tns:sourceData&gt; &lt;![CDATA[<b>CONTENTSOURCE</b>]]&gt;&lt;/tns:sourceData &gt;</pre>			
D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: 183 / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

```
</tns:payload>
</tns:transformationRequest>
```

**Response:**

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:transformationResponse xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/TransformationS/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/TransformationS/TransformationRes.xsd ">
  <tns:header>
    <tns:commandResult>OK</tns:commandResult >
  </tns:header>
  <tns:payload>
    <tns:resultData> <![CDATA[SUMMARY]]></tns:resultData >
  </tns:payload>
</tns:transformationResponse>
```

**3.5.5.3.2 Content Formats and Protocol Definitions**

The Data extraction service is an extension of the transformation service, thus, it uses the same data communication structures that the Transformation Service or extensions of it.

**Extraction Service message format XML/Schemas**

*<http://www.fp7-adventure.eu/xmlSchema/TransformationService/ExtractionReq.xsd>*

An XML document will be used to transfer data and control information from an ADVENTURE component to the Data Extraction (Transformation) Service, by means of the Message Routing functionality. XML, according to the XML-schema shown below, will be transferred as payload of the Message Routing call, and contains all needed parameters for invoking the API method specified in the sections above.

**Request-Format:**

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>184</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

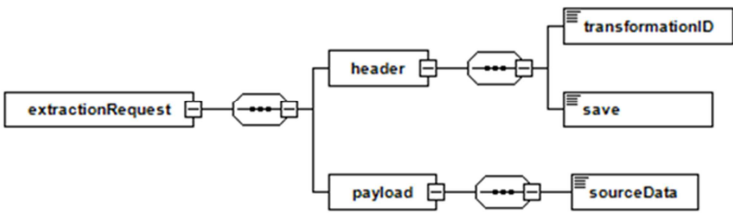


Figure 42 - Data Extraction request schema

Data elements

transformationID: Unique identifier of the transformation definition, which must be stored on Cloud Storage, to be performed. This ID will be used to retrieve the details and auxiliary files needed to perform the transformation by TSB.

save: By using this Boolean parameter during the configuration of the service, the user indicates if the data extracted has to be stored in the Data Storage System or not.

sourceData: Data to be transformed into the required format. This data structure contains a name attribute, and an encoding attribute, describing if the value data is encoded (base64) or not. In the case of large files, gateways will use the Message Routing features for large attachments, that will transparently store the file in a temporal bucket, and use a URI reference (URI - Universal Reference Identifier) into the source data parameter, so that the large file could be retrieved just in the moment of using it.

Additional information collected from the incoming Message Routing call.

componentID: Identifier of the caller component, so that the transformation result can be routed to it.

from: Identification of the sender user (or company).

to: Identification of the recipient user (or company).

Response-Format:

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>185</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The response format of the Data Extraction service is similar to the format returned by the Transformation Service. The only difference is that the result data always contains a summary data format.

### Summary data model XML/Schemas

<http://www.fp7-adventure.eu/xmlSchema/summary.xsd>

The result of a `extractionRequest` operation will always have the following generic format (characterized pair-value) and a decision is proposed to always store the content in XML. In this way, any ADVENTURE component will be able of processing this information because they know what format to expect and can, for example, amalgamate different kinds of data. The Summary is a kind of metamodel that will be able to support any kind of data extraction. The values of the semantic characterization of the elements will come from a common vocabulary (using like micro formats, RDFa structures, etc), so that afterwards the elements could be aggregated by using this common characterization (e.g. list of orders based on `orderNumber`).

The following format will be returned as part of the result of the call to the extraction services.

```
<summary>
  <info name="NAME" semRef="REF" >value</ info >
  <info name="NAME" semRef="REF" >value</ info >
</summary>
```

and a sample can be

```
<summary>
  <info name="orderId" semRef="http://adventure/summary#orderNum">234</info>
  <info name="status"
semRef="http://adventuresems/summary#orderStatusPercentage">10</info>
  <info name="temperature"
semRef="http://adventuresems/summary#temperatureCelsius">45</info>
  <info name="stock"
semRef="http://adventuresems/summary#availableStockUnits">123</info>
</summary>
```

This is a very simplistic approach that can be used for very simple data extraction, so an extension of it is proposed. It implements the same approach, but it is defined in a recursive way, so that it can support generic tree structures. As an example, a summary of an order format is shown below.

```
<summary>
<info name="string" semRef="http://adventure/summary#order">
```

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>186</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

```
<info name="orderNum" semRef="http://adventure/summary#orderNum" value="ORD-PO123-4"/>
<info name="date" semRef="http://adventure/summary#orderDate" value="2012-07-01" />
<info name="buyer" semRef="http://adventure/summary#orderBuyer" value="Buyer Inc"/>
<info name="supplier" semRef="http://adventure/summary#orderSupplier" value="Supplier Ltd"/>
<info name="line" semRef="http://adventure/summary#orderLine" value="1">
  <info name="itemDesc" semRef="http://adventure/summary#productDesc"
value="PartA" />
  <info name="itemId" semRef="http://adventure/summary#productId" value="57" />
  <info name="deliveryDate" semRef="http://adventure/summary#deliveryDate"
value="2012-07-20" />
</info>
<info name="line" semRef="http://adventure/summary#orderLine" value="2">
  <info name="itemDesc" semRef="http://adventure/summary#productDesc"
value="PartB" />
  <info name="itemId" semRef="http://adventure/summary#productId" value="5" />
  <info name="deliveryDate" semRef="http://adventure/summary#deliveryDate"
value="2012-07-21"/>
</info>
</summary>
```

And the previous "plain" example can also be transformed into

```
<summary>
  <info name="orderId" semRef="http://adventure/summary#orderNum" value="ORD-PO123-4"/>
  <info name="status" semRef="http://adventure/summary#orderStatusPercentage" value="10"/>
  <info name="temperature" semRef="http://adventure/summary#temperatureCelsius" value="45"/>
  <info name="stock" semRef="http://adventure/summary#availableStockUnits" value="123"/>
</summary>
```

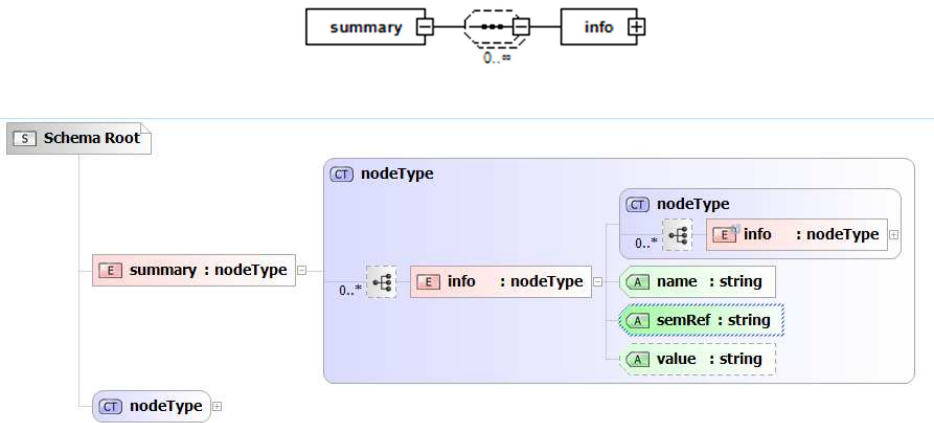


Figure 43 - Graphical representation of the XML schema

Data elements

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: 187 / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

summary: Root of the summary structure.

info: element recursively defined at the summary schema. It contains three attributes,

name: Name of the element (given by the user when defines the summary information) and that can be afterwards used for graphical representation of the data.

semRef: reference to the common summary vocabulary.

value: representing the value of the specific information element extracted.

As shown, the improved data structure can also implement the first approach. The advantage of having this structure, is that is very flexible (although it depends on the semRef definitions and the richness of the semantic resource), but also allows to keep the structure of the data represented. This structure can also be used afterwards (by e.g. the Monitoring or Dashboard components) to represent the data contained in a graphical way. On the negative part, the use of an abstract data model, referencing semantic data, can be confusing for the user until it gets used to this way of working.

Apart from that, the structure can be easily parsed and queried by using xpath notation or XQuery language. e.g. if the order number has to be recovered, then using Xpath can be something like

```
//info[@semRef="http://adventure/summary#orderNum"] or in Xquery
//info[contains(@semRef, 'http://adventure/summary#orderNum')].
```

In terms of the semRef for simplicity this can be a quickly created ontology of ADVENTURE semantic entities, but since it is not the goal (feasibility or practical reality) for ADVENTURE users to always adopt an adventure ontology this must be flexible but can quick-start things. External ontologies can be referenced through use of a suitable namespace.

### Cloud Storage data formats

The definition of an extraction operation will be similar to the definition of a transformation operation (in fact it extends the data storage model described at the Transformation Services data models section) and hence a mapping file will be needed. This mapping file will extract only the needed data and convert it into the

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>188</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



proposed data format (including the semantic reference field) and in general the main differences are that the information extracted is summary information (few fields) and the information is generally saved in Cloud Storage.

In this way the user and the ADVENTURE components can have a description of the specific data set that a summary extraction operation will return.

In order to describe and store the data extraction operations definition, the data model below will be used. NoSQL buckets will be used to store the descriptive data. This descriptive data will make reference to specific stored binary files that will describe the source and destination formats, the mapping files, and other auxiliary files if necessary.

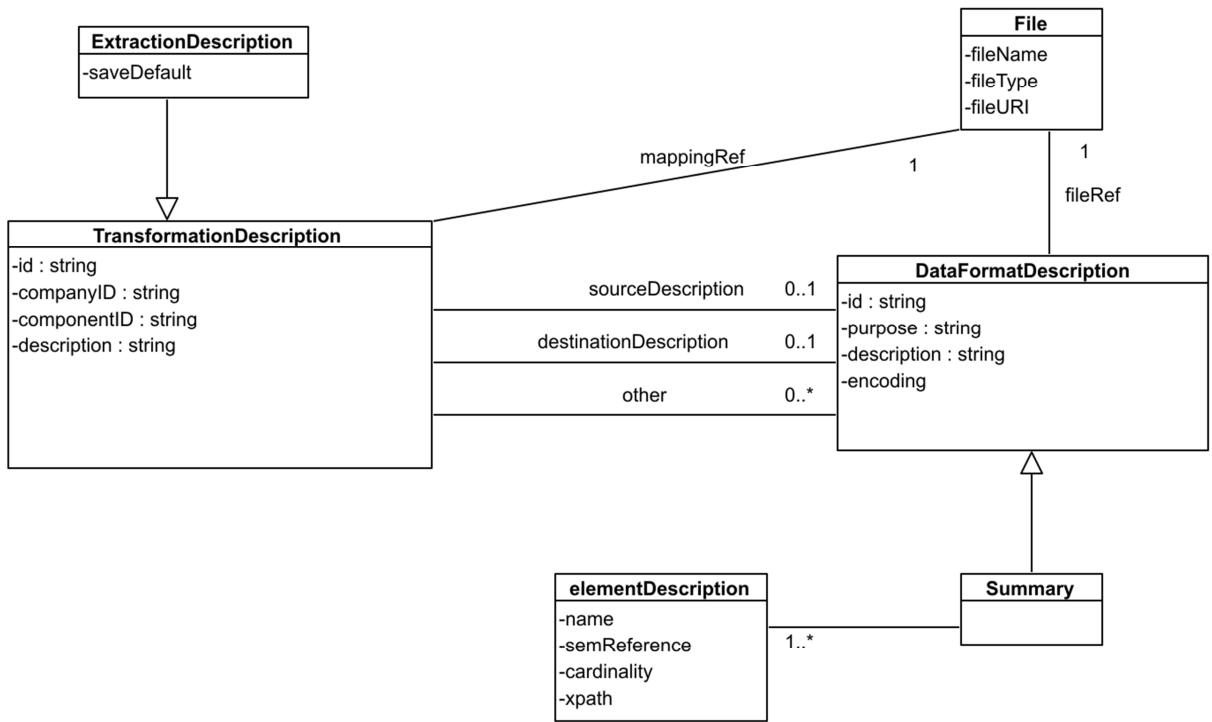


Figure 44 - Extraction operation description object model

The data extraction description data can, for example, be used by the process designer when configuring a data transformation element. *TransformationDescription*

- id: Unique Identifier for the transformation description.
- companyID: Identification of the Broker that has set up the transformation.

componentID: Id of the component (Transformation service) that will execute the transformation. Several transformation services can exist in the ADVENTURE platform.

description: Textual description of the transformation operation.

#### *DataFormatDescription*

id: Unique Identifier of the data format in the ADVENTURE Cloud Storage.

purpose: Classification of the purpose of using the data format (e.g. orderStatus retrieval).

description: Textual description of the data format.

#### File

fileName: Name of the file.

filetype: Type of the file

fileURI: URI to recover the file at AVENTURE Storage System.

#### SummaryElementDescription

name: Name of the summary element.

semReference: URI to the vocabulary element defined for summary.

cardinality: Cardinality of the element on the summary returned.

xpath: Xpath to extract the specific element from the summary document.

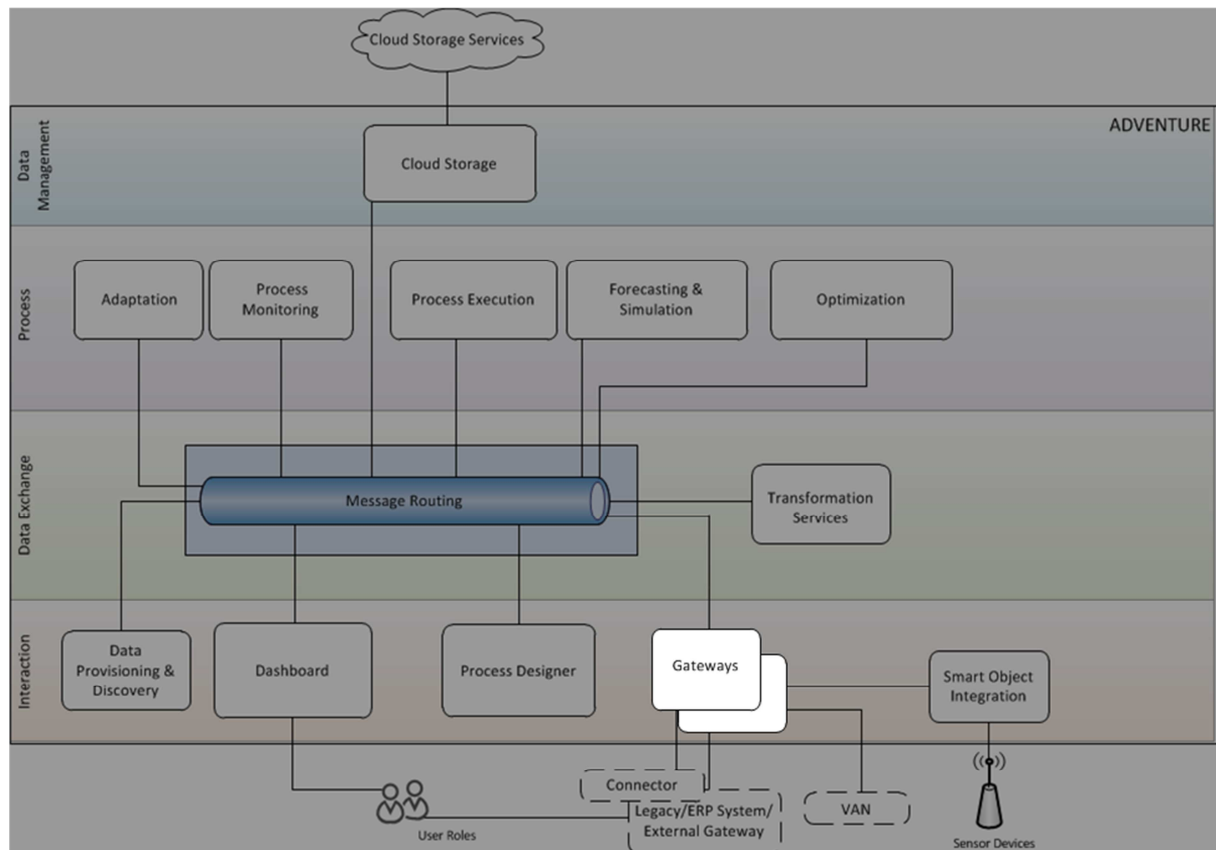
### 3.5.6 Summary

The Transformation services component (and its extended component, the Data Extraction Component), will use TIE Smart Bridge (TSB) as core technology to perform transformations between different data formats. These functionalities will

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>190</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

allow the ADVENTURE brokers to define their processes relying in the data formats currently used at their companies, and transform them (or extract data from them) afterwards, by defining/implementing the appropriate mapping files. The data format defined at the Data Extraction Service will be used internally by ADVENTURE, allowing the different modules (e.g. Dashboard, Monitoring, etc...) to work not just with process data, but also with related business data.

# Gateways



## 3.6 Gateways

Abstract:

ADVENTURE Gateways functionality provides a framework for building custom made connections to external systems, such as ERPs or Smart Objects.

A significant part of the gateway implementation is tailored to deal with a specific end system type or instance.

Gateways are agnostic with regard to the content exchanged and the format, gateways are just about communication.

Companies will be capable of registering their gateways as services on the ADVENTURE platform by defining them from a technical point of view.

### 3.6.1 Major Design Decisions

In this section, different design decisions will be exposed and justified.

The first design decision is related with the fact that ADVENTURE will provide a general framework for defining and implementing gateways. Third party system integration is a complex and long-standing problem when trying to communicate company data to or when trying to acquire data from other external systems. This usually implies different technologies and data models to be integrated of which middleware like ADVENTURE has no control over (and never will). ADVENTURE's purpose is not to add more complexity to this (e.g. by defining an additional protocol) but to reuse, as much as possible, existing integration activity. This will accelerate the return of the investment, especially for SMEs, and reflects the reality that companies will be unlikely to change their systems just because of ADVENTURE.

ADVENTURE Gateways will thus facilitate on one hand the communication with the ADVENTURE platform and on the other hand the reuse (or easy adaptation) of existing integration implementations.

Within ADVENTURE a company will be able to expose one or more gateway as access points which are specific addresses used by the ADVENTURE platform to access the data exposed by a company. A company gateway endpoint will be registered as an ADVENTURE component and accessed via the Message Routing functionality so that the Smart Process Execution (or other internal component) can call it.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>193</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

A gateway access point will expose different "services" that will be defined by the company itself by using a specific ADVENTURE DPD UI. A gateway service definition will correspond with a specific operation (i.e. getOrder(orderID)) at the company 3rd party system. The operation identifier will be sent as part of the call to the company gateway which will manage the execution (by means of the company customizations necessary to be done at the gateway and in the context of ADVENTURE via the use case partners of WP7).

ADVENTURE defines data requirements for two different kinds of operations:

Data at execution time

Data for forecasting and simulation functionalities

In order to fulfill these requirements, the definition of a gateway service can be linked to an alternative operation (in the same, or in an alternative access endpoint). The alternative operation will be used when the calling components are performing forecasting and simulation operations. It will simulate the execution of the operation, thus returning forecast data about the operation executed (e.g. maximum deliver time, possibility of providing in x days, ...) As not all the 3rd party systems will be able to provide information for Forecasting purposes, the alternative Forecasting service definition will not be mandatory.

Usually a gateway is physically installed on the company premises and has access to the 3rd party systems or Smart Objects that have to be integrated (to gather context information, access their file systems, databases, etc...). There will be also scenarios in which a custom connector will interface remote interfaces, such as SOAP Services, REST services, OData endpoints or VAN messaging systems. The implementation of the interfaces with these complex systems is not an ADVENTURE task. It depends on the custom implementation made by the company itself. Only the implementations needed to cope with the Use Case Implementation (T7.2) will be established by ADVENTURE.

For facilitating adoption, ADVENTURE gateways will define a default set of simple operations that are proposed to be present (although not mandatory) at the gateways implementations dealing with the specific scope of order and product information (note that gateway functionality does not only cover the order and product specific scenario, but provides a common framework for communicate also with other software and hardware artifacts, like Smart Objects). This default set of operations

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>194</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

will provide ADVENTURE with a common content data set, so that the different components at ADVENTURE can work successfully (e.g. getOrderStatus(orderNumber, productID), getAvailableStock(productID) and/or queries for current context parameters of Smart Objects... ).

ADVENTURE will provide a reference implementation of a gateway with this minimal set of operations (See the reference implementation description in the Specification of Interfaces, Protocols and Formats section) foreseen for order and product information at a company. This implementation will be based on the use of a tool that is easy for adoption at SMEs (e.g. Excel file) so that it can be easily updated from 3rd party systems, or manually in case these systems are not available at the company.

Finally, Gateways are the main way of obtaining data from 3rd party systems at ADVENTURE. In this respect, a gateway could provide several different types of outputs:

Documents: EDI documents, XML documents, CSV documents, etc...

Summary information (see transformation/data extraction service): XML structure containing a subset of operational data recovered from 3rd party systems, or extracted from business documents (e.g. EDI order).

Both: A gateway output can contain both, documents and its associated summary information.

The specific data format that a gateway operation will return will be specified as part of the gateway operation specification.

### 3.6.2 Technology Comparison and Selection

This section will describe key criteria to be taken into account for the development of the gateways framework and reference implementations.

Note that gateway technology depends, on the ADVENTURE side, of the technology agreed by the Message Routing Component for communication, and on the other (3rd Party System side) on the technology, adaptations, previous integration experiences, over the 3rd party systems themselves. As an example of the dependency on the 3rd party side systems, a custom gateway connector could perform queries to a Relational Database Management System RDBMS in order to retrieve information, store a file in a specific directory so that an existing process

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>195</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

could import it, or make a call to a complex SOAP interface to trigger some action in a remote ERP.

The custom connector implementation depends exclusively on the external system to be integrated. This is also applicable in the case of Smart Objects, in which, depending on the employed technology, on the location where the hardware is installed, etc..., a different strategy will be adopted to send the captured information through the gateways. Since no specific technology selection can be performed for this component as a whole, the Gateway will deliver the policies to follow, a reference implementation and recommended tools to facilitate the development of the custom parts of a gateway. In this respect, different Enterprise Application Integration frameworks exist that could be useful for the implementation. The applicability of one of these frameworks to a specific custom implementation will depend on multiple variables (complexity of the implementation, existing integration mechanisms, experience and knowledge of the developer team, etc...) and the use cases.

As such only a summary of some applicable tools will be performed but no specific technology will be selected.

### 3.6.2.1 Selection Criteria

The selection criteria for Gateways technologies were defined in the *D3.2 Functional Specification* in section 5.7.5. The description of the generic parameters can be found in the *D3.2 Functional Specification* in section 5, page 28. Now follows the description of the Gateways specific parameters:

The main sense of the above parameters are related to the ability of the gateway of providing directly or be plugged to elements providing the requested feature.

**Gateway description based on standards:** This parameter relates to the preferences about the possibility of reusing a standard data model or other previous initiative results, to describe the internal structure of gateways services, endpoints, characteristics and associated parameters (e.g. USDL).

**Allow semantic annotation:** These allow semantic annotation points to the possibilities of having gateways information characterized by means of semantic annotation like purpose, scope, etc....which allows the better finding and mashing of such services

**REST support:** This criterion is related with the ability of gateways to support REST protocol for connectivity with at the 3<sup>rd</sup> party side.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>196</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



**Web Services support:** Preferences about gateways supporting Web Services protocol at the 3<sup>rd</sup> party side.

**SMTP support:** Preferences about gateways supporting SMTP/MIME protocol at the 3<sup>rd</sup> party side.

**VAN support:** Preferences about gateways supporting EDI-VAN protocol at the 3<sup>rd</sup> party side.

**Authentication:** Allows the passing parameters such as user/password or a security token to be checked at the custom part of the gateway.

### 3.6.2.2 Possible Technologies

As explained before, due to the special characteristics of this component, no technology selection will be performed because it depends mainly on the related ADVENTURE components, the specificity of the system to be integrated or the knowledge of the person that has to implement the customized access to the system.

A State of the Art analysis of existing EAI (Enterprise Application Integration) frameworks has been performed as a recommendation to the future implementers of the custom part of the gateways. Note that EAI are integration facilitators, but usually they are very complex and have a high learning curve, so that depending on the knowledge of the developer and the objectives and complexity of the integration scope, a specific decision has to be taken whether to use them or just make a low level implementation (developer coding specific solution).

The different options and tools proposed for the implementation of gateways custom connectors are:

**Low level implementation:** It refers to the possibility of developing a specific solution, coding it from scratch at the company. This solution can be an option for very specific implementations, and will be used in companies that have not implemented an integration strategy, or whose size and resources make it more affordable.

**Apache Camel** (<http://camel.apache.org/>): Apache Camel is one of the most popular EAI frameworks at the moment. Apache Camel implements EIP (Enterprise Integration Patterns). It provides up to 100 integration components, including cloud based data services, SQL and noSQL databases, REST, Web services, and tools for dealing with different data formats (XML, Json, CSV, flat files,...). Beyond this, it provides the ability to execute it in a standalone environment, making it easier for

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>197</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

adoption by, e.g. SMEs. Camel is one of the most dynamic Apache projects, having a broad community both contributing and using it for wider developments (Camel is a core part of the Apache ESB ServiceMix, and has been also recently selected to become a core component for TalendESB).

**Spring Integration** (<http://www.springsource.org/spring-integration>): Spring Integration provides an extension of the Spring programming model to support Enterprise Integration Patterns. It supports integration with external systems via declarative adapters. Spring integrations provide enterprise orchestration and adapters for distributed applications and batch applications. It supports traditional RDBMS as well as new NoSQL solutions, map-reduce frameworks and cloud based data services. Spring Integration provides a comprehensive integration framework oriented to Spring based applications. Spring Integration is a very active project in the SpringSource Community.

**OpenAdaptor** (<https://www.openadaptor.org/>): OpenAdaptor provides many ready-built connectors that include JMS, JDBC, IBM MQ Series, TIBCO Rendezvous, TCP/IP Sockets, SOAP, HTTP and Files. OpenAdaptor provides exception management and scriptable components for data filtering, simple transformation and validation. It was originally conceived in 1997, to facilitate a large financial organization's requirement to integrate its very large application suite using Message Oriented Middleware. It was released to the Open Source Community in 2001. OpenAdaptor is providing a couple of new versions per year.

**TIE SmartBridge** (<http://businessintegration.tiekinetix.com/node/802>): TSB provides tools for seamless integration of backoffice solutions by implementing different interoperability strategies. TSB includes a B2B message exchanging broker that, in a very efficient way, is able to transport a *B2B XML or EDI message* from a source point to its destination, performing the adequate message transformations and routing. TSB is a sample of a commercial tool that can be used at the gateways in order to integrate 3rd party systems.

The following table (Table 11) compares the different solutions that can be used by the companies' developers to facilitate data integration with 3rd party systems.

*Table 11- Comparison of different technologies/strategies for Gateways*

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>198</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Parameter	Importance (--- +/- +++)	Low level implementation	Apache Camel	Spring Integration	Open Adaptor	TIE Smart Bridge
<b>Generic Parameters</b>						
Maturity & Stability	++	+/-	++	++	++	+++
Regularly Updated	+	--	+++	+++	++	+++
Technical Up-to-Dateness / Appeal	+/-	+	+++	++	+	+++
Open Source	YES/NO	NO	YES	YES	YES	NO
Non-Infecting	YES/NO	YES	YES	YES	YES	NO
Code-Quality	++	+/-	++	++	+	+++
Extensibility	+	+/-	+	+	+	+++
Community	YES	NO	YES	YES	YES	NO
Performance	+/-	+/-	++	++	++	+++
Reuse of existing developments	++	+/-	++	++	++	+++
EU project origin	NO	NO	NO	NO	NO	NO
Platform (Portability)	++	+/-	+	+	+	+
Open Standards Compliance	YES	NO	YES	YES	YES	YES
Interoperability	+	+/-	+++	+++	+++	+++
<b>Specific Parameters</b>						
Gateway description based on standard	YES/NO	NO	NO	NO	NO	NO
Allow	YES/NO	NO	NO	NO	NO	NO
D3.3-Technical-Specification				Author: UVA and Partners	Date: 2012-09-24	Page: 199 / 372

Copyright © ADVENTURE Project Consortium. All Rights Reserved.

semantic annotation						
REST support	+	+/-	+++	+++	+	+++
WS support	++	+/-	+++	+++	+++	+++
SMTP support	+/-	+/-	+++	+++	+++	+++
VAN support	+/-	+/-	+	+	+	+++
Authentication	+/-	+/-	+	+	+	+++

Note that these frameworks and tools are usually oriented to integrate systems such as ERPs.

In the case of Smart Objects using gateways for transmitting the data, the custom connector will be developed under the specific component implementation task (T4.3). So, specific means to deal with accessing different hardware devices, elaborate varying data and transmit it through the gateways will be found in the Smart Objects Integration specific section of this technical specification.

#### 3.6.2.2.1 Conclusion

The State of the Art analysis of existing EAI frameworks have been performed as a recommendation to the future implementers of the custom part of the gateways. No specific selection is proposed; due it depends on the specific characteristic of the implementation to be performed, as explained in the next section.

#### 3.6.2.3 Technology Selection

As mentioned before, gateways are very dependent on the system to be integrated, thus no universal solution can be implemented. As a framework, the gateway implementation will provide the user with:

Tools and APIs to talk to the ADVENTURE platform

Interfaces and high level reference implementation to deal with the specific (3rd party) system side, meaning that this part will be usually developed by the company itself by means of a custom implementation (based on the gateways reference implementation and provided APIs), or by using one of the tools proposed in the previous section.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>200</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

As it can be observed in the comparison tables, few differences exists between the different EAI integration tools in terms of ADVENTURE project requirements (which is normal because it is the main target of these tools).

Having this scenario in mind, any of these tools can be suitable to implement a custom ADVENTURE gateway connector. It will be then a decision of the company development team to use one of them or not. Due to the specific orientation of ADVENTURE to SMEs, this decision will depend mainly on the knowledge of the person that has to perform the custom implementation, and the overall integration scenarios expected at the company.

#### 3.6.2.4 Missing Elements and Implementation Needs

Gateways define a framework for data exchange with 3rd party systems and Smart Objects (devices) so a different implementation will be needed for specific systems at company. Gateways will provide a reusable reference implementation that can be used for testing at the company and/or for a simple and de-coupled integration at companies with insufficient knowledge to enable a tight integration.

Gateway reference implementation will provide the companies with:

All the communication tools needed to communicate with ADVENTURE (retrieve messages, respond to them, and allow the push of information).

High level (reference) implementation of a custom connector.

Systems integration is very specific for each of the systems to be integrated, so a generic solution or implementation cannot be provided, e.g. the ERP can be same or different implementation technology (java/.NET), can have or not integration facilitators (API or DB access), etc.

An important missing element for the gateways implementation is the UI that the ADVENTURE brokers will use in order to define the different gateway services, their inputs and outputs, etc. The gateways service description UI will store information about the gateways and gateways operations at the Cloud Storage System.

This UI will be accessed at the Company profile description (DPD UI) in the Dashboard.

The gateway services description will be used by the Data Provisioning and Discovery module when describing the services offered by a specific company. The

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>201</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

company gateway service description will also be used by the Process Designer, allowing associating a specific gateway call, to a specific activity.

The DPD service configuration and the gateways service description UI are complementary. Gateways service description UI is intended to describe the gateways at a low level (operations, data types, etc.),while the DPD uses the gateways operation descriptions and configure Companies services at a higher level (manufacturing scope, etc.).

3.6.3 Technical Component Specification

3.6.3.1 Component Structure

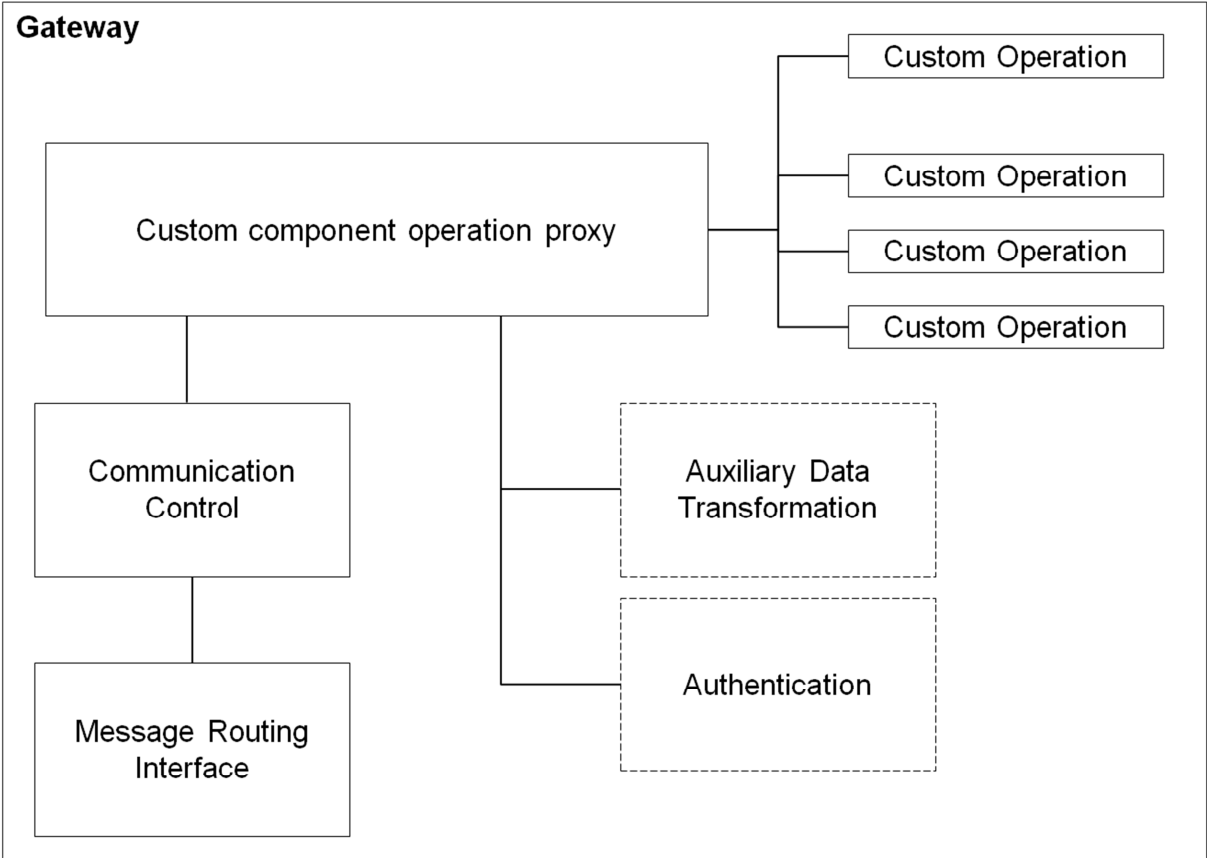


Figure 45 - Gateway component structure

The Message Routing and Communication Control parts will be provided to the custom gateways developers (as ADVENTURE code libraries) to facilitate the implementation of a specific gateway. The "Custom component operation proxy" and

the rest of the components will be a custom implementation of each specific gateway instance. They will be provided to the developers in the form of interfaces, abstract classes and partial implementations to facilitate adoption. A reference implementation will also be provided to illustrate how an implementation can work.

A gateway consists of the following parts:

### **Message Routing Interface**

This subcomponent implements the Message Routing API and provides communication with ADVENTURE. An event handler will be triggered if a message is received and Communication Control element starts to handle the request.

### **Communication Control**

The communication control handles the different requests that are sent to/from the ADVENTURE platform. It implements basic control flow for communication, such as handling the messages that are triggered from the 3rd party systems, allowing a push mechanism into the adventure platform.

### **Custom component operation proxy**

A gateway endpoint can expose different operations to be performed on the 3rd party system. These operations and their parameters are described (by the ADVENTURE broker) as part of the company gateway description on the ADVENTURE platform. When a call to a gateway is performed, the corresponding operation to be executed has to be included as part of the call.. The "Custom component operation proxy" recovers from the message, the information about which specific operation has to be performed and starts the execution of this operation by calling the Custom Operation Protocol (e.g. store a file in directory X, perform an SQL query on Database Y, Call a VAN gateway to deliver the message to Z, query a Smart Object for current context data, etc.). The Custom component operation proxy can use (if specified) the Authentication and Auxiliary Data transformation modules.

### **Authentication**

As part of the custom part of the gateway an authentication mechanism can be implemented. This will ensure that only authorized users can have access to the calling mechanisms linked to the in-house systems. It is recommended for this to be based on a user/password mechanism or in passing a security token. The checking mechanism depends entirely on the custom application. The usage of the

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>203</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

authentication mechanism is defined when the company publishes the different gateway operations exposed.

### Auxiliary Data transformation

Depending on the scenario and on the custom connector implementation, it may be necessary to perform some simple data transformation operations. This customized module will perform these operations utilizing the Transformation Service directly if the transformation is complex enough. Typically this would be relevant where always data is transformed regardless of the process or the involved partners. In this case, the Content Extraction Service can also be called from the Auxiliary Data Transformation module in order to allow the Gateway to return or parse XML Summary content. For the summary data extraction, the developer could also choose the option of coding the XML itself (based in the information retrieved by the custom connector) although this possibility is probably just suitable where there is not a lot of information to add to the summary, or if this information is easily accessible.

#### 3.6.3.2 Internal Sequence Diagrams

This section will give an insight about the internal component communication sequence based on a list of example sequences. It's extracted from the use case scenarios of D3.2.

executeAction

The following sequence diagram depicts the internal sequence that a gateway component will follow when a component (e.g. Smart Process Execution) sends request to a specific gateway.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>204</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



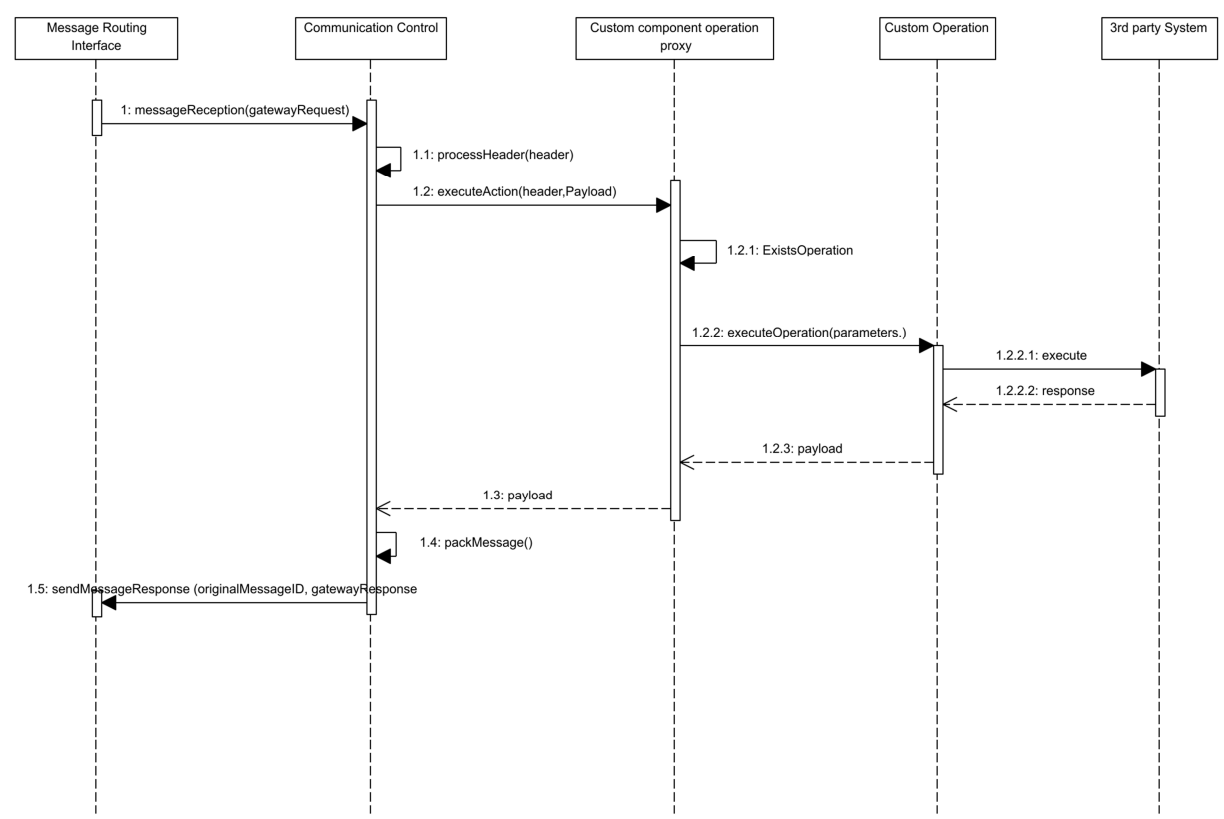


Figure 46 – Gateways, call/ response scenario

A gateway is a software component linked to the Message Routing component, so that all the calls coming from the ADVENTURE framework will be delivered through it. Once the message has been delivered to the gateway, the Communication control component will take care of it, extracting the information from the request call, and coordinating the different actions to be performed. Once this component has the information about which specific operation has to be executed, it is passed to the operations proxy that will give it to the specific custom operation implementation. The custom operation implementation will contact the 3rd party system (e.g. it will perform a query on a DB, using the parameters extracted from the original message) and retrieve the information requested. This information will be sent back to the Communication control (after some data transformation if needed) that will compose the return message, and deliver it to the Message Routing (that will send it to the requester component).

pushMessage

The push Message communication schema is not started by a component at the ADVENTURE infrastructure, but by a software artifact directly connected to the 3rd party system (or by the 3rd party system itself).

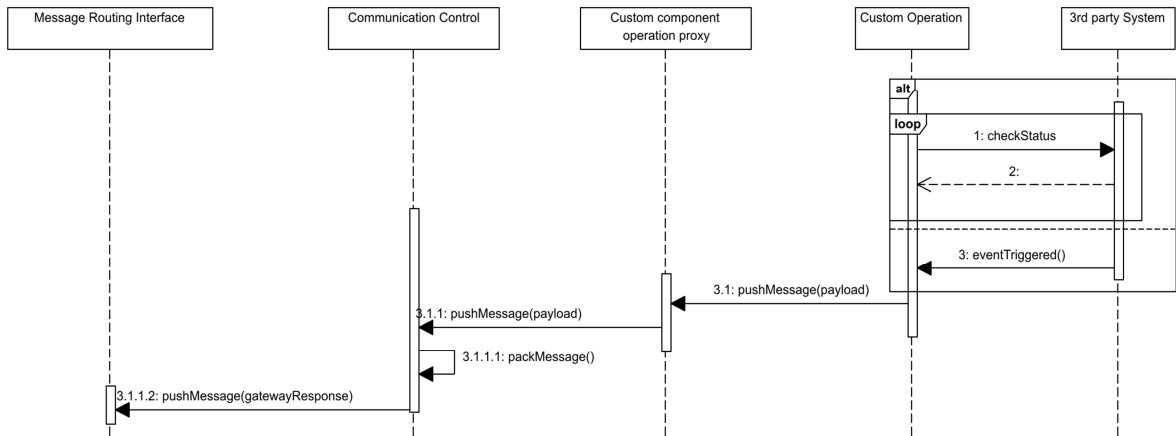


Figure 47 - Gateways, pushing message scenario

Usually it will consist in a daemon function checking for changes, and will send a message to the ADVENTURE Smart Process Execution module with the contents of the change occurred at the 3rd party system (in the data format specified in the gateway operation description). It can also consist on a listener waiting for specific message sent by the 3rd Party System.

Internal extraction service call

The following diagram describes an example scenario, in which the developer of the custom operation has implemented an internal call to a Data Extraction service, so that the result of the gateway call could contain, both, a document (EDI order) and a summary of the information contained on it (e.g. order number, delivery date and total amount).

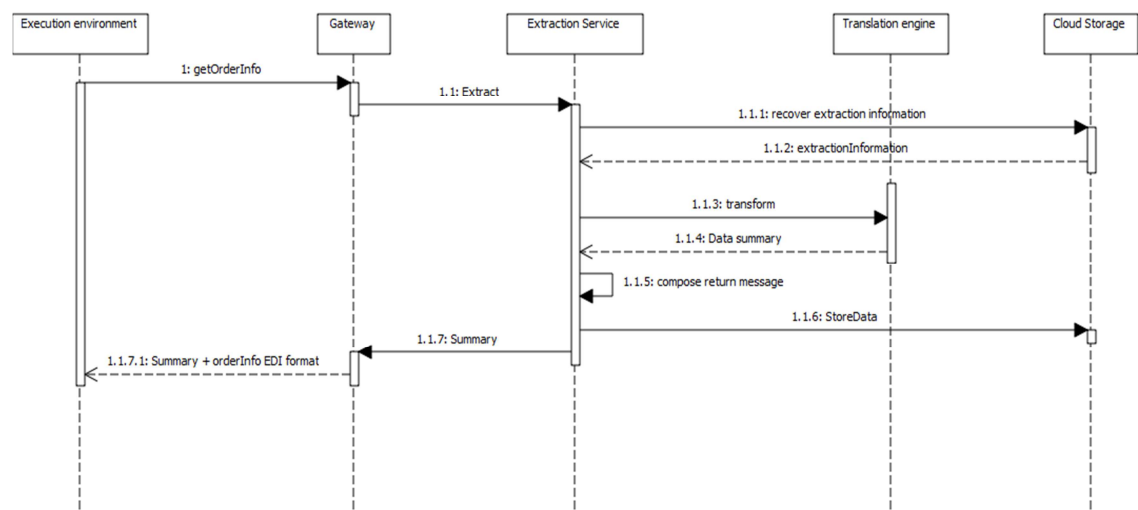


Figure 48 - Sequence Diagram - Data Extraction Service called from within a gateway operation

3.6.4 Specification of Interfaces, Protocols and Formats

Gateways requests and responses are provided to the rest of components of ADVENTURE, through the API exposed at the component. In the following sections these interfaces are explained.

3.6.4.1 API specification

The Gateways component will provide a list of API methods that can be indirectly used by the other ADVENTURE components via the message routing component or that are used internally by gateway component in order to interact with the rest of the ADVENTURE platform.

Those API methods are defined as follows:

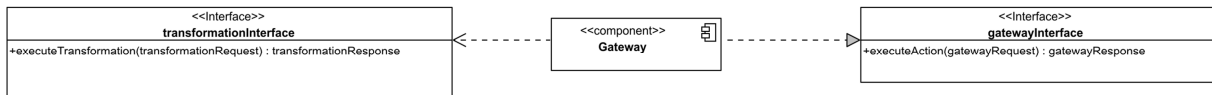


Figure 49 - Gateway interfaces (required and provided)

executeAction

This method will receive a message from an ADVENTURE component in to the gateway in order execute it at the 3rd party system side.

```
public GatewayResponse executeAction( String gatewayRequest )
```

### Parameters

gatewayRequest: XML structure containing the specific details of the operation to be performed at 3rd party system. The specification of the contents of this XML structure can be found in the Content Formats and Protocol Definitions section.

### Return Value

gatewayResponse: XML structure containing the response given by a specific operation performed at a 3rd party system. The specification of the contents of this XML structure can be found in the Content Formats and Protocol Definitions section.

### Message-Example

#### Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:transformationRequest xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/TransformationS/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/Gateways/GatewayRequest.xsd ">
  <tns:header>
    <tns:operationName>getOrderStatus</tns:operationName>
    <tns:transformationID>TransformationID</tns:transformationID>
    <tns:communicationType>continuous</tns:communicationType>
    <tns:authenticationInfo>SecToken284576398</tns:authenticationInfo>
  </tns:header>
  <tns:payload>
    <tns:inputParameter paramName="orderNumber" encoding="false">
      23</tns:inputParameter >
  </tns:payload>
</tns:transformationRequest>
```

#### Response:

```
<?xml version="1.0" encoding="UTF-8"?>

<tns:transformationRequest          xmlns:tns="http://          www.fp7-
adventure.eu/xmlSchema/TransformationS/"          xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"          xsi:schemaLocation="http://          www.fp7-
adventure.eu/xmlSchema/Gateways/GatewayRequest.xsd ">
```

```
<tns:header>
  <tns:operationName>getOrderStatus</tns:operationName>
  <tns: commandResult>OK</tns: commandResultDescription>
</tns:header>
<tns:payload>
  <tns:outputInformation paramName="orderNumber">23</tns:outputInformation>
  <tns:outputInformation paramName="orderCompletion">60</tns: outputInformation>
  <tns:outputData><![CDATA[XML_ORDERSTATUSREPORT_DOCUMENT]]></tns:outputData>
</tns:payload>
</tns:transformationRequest>
```

3.6.4.2 Content Formats and Protocol Definitions

An XML document will be used to transfer data to and from the Gateways via the message routing component. The XML-schema, which will be shown below, contains all needed parameters for invoking the API method specified in the sections above. The XML contains a “header” element to include the gateway operations and parameters. It also contains a content part (payload) where the data needed for the operations is contained.

Gateway message format XML/Schemas

<http://www.fp7-adventure.eu/xmlSchema/TransformationService/GatewayReq.xsd>

<http://www.fp7-adventure.eu/xmlSchema/TransformationService/GatewayResp.xsd>

Request-Format:

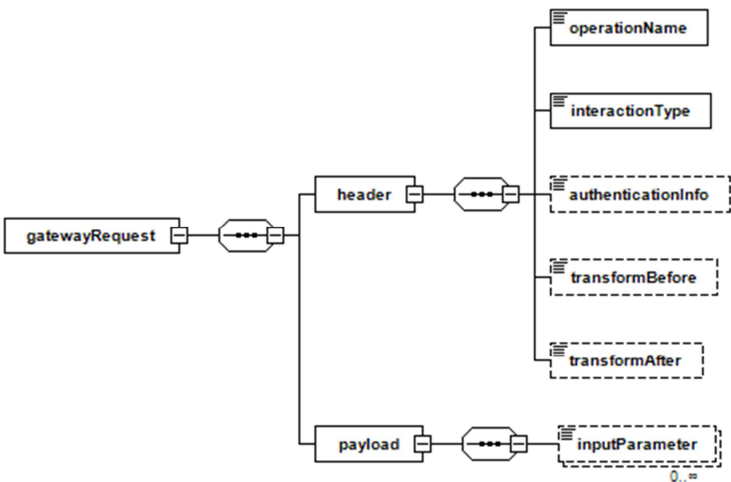


Figure 50 - Gateway request schema

*Data elements*

operationName: Specific operation to be executed by the custom connector at the 3rd party system (e.g getOrderStatus).

interactionType: Kind of interaction schema applicable for the current operation (NORMAL or CONTINUOUS). E.g. "continuous" if the operation has to be repeated every x minutes and deliver results by means of a push action until an end condition is reached (the specific behavior will be hardcoded at the custom operation implementation). Normal meaning is for a usual synchronous call.

authenticationInfo: Security token to be checked at the third party system in order to constrain access to the gateways operations.

transformBefore: Field that triggers the execution a specific call to the Transformation Services component from the gateway to perform a transformation on the input data before calling the custom connector functionalities. This field contains the ADVENTURE transformation identifier, and the inputParameters will be used as source data for the transformation.

transformAfter: Field that makes possible to execute a specific call to the Transformation Services component from the gateway to perform a transformation on the output data before returning it to the caller component. This field contains the ADVENTURE transformation identifier, and the internal output data will be used as source data for the transformation.

inputParameter: input parameters containing the name, if it is encoded or not, or if it is an external reference (false, base64 or REF) and the value. In the case of large files, gateways will use the Message Routing features for big attachments that will allow to temporary store the file, and use a URI reference (URI - Universal Reference Identifier) into the input parameter, so that the big file could be retrieved just in the moment of using it.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>210</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Additional information collected from the incoming Message Routing call.

- MessageId: Identifier of the message that is calling the gateway service from the execution environment.
- from: Identification of the sender user (or company).
- to: Identification of the recipient user (or company).

Response-Format:

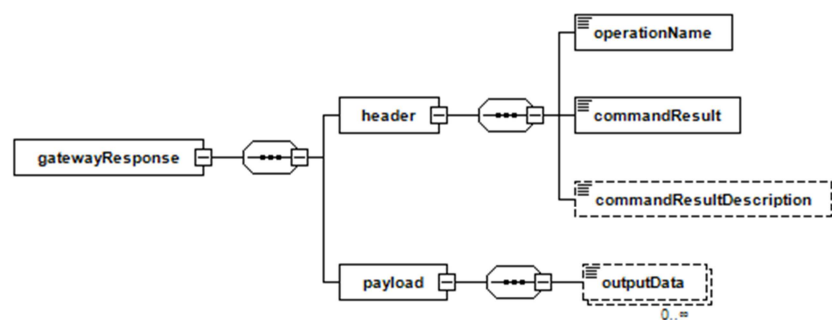


Figure 51 – Gateway response schema.

Data elements

- operationName: Specific operation executed by the custom connector at the 3rd party system (e.g getOrderStatus).
- commandResult: Result of the command executed (OK, ERROR)
- commandResultDescription: If an error has occurred, this field describes the cause of the error so that the upper process can be informed.
- outputData: Document (if any) resultant on the call to the 3rd party system (e.g. deliveryNote document, orderStatus document, etc..). The output document characterization is part of the specific gateway service description. It will contain a specific attribute describing if it is encoded or not, or if it is a reference (false, base64, REF). In the case of large files, gateways will use the Message Routing features for big attachments that will allow to temporary store the file, and use a URI reference (URI - Universal Reference Identifier) into the outputData parameter, so that the large file could be retrieved just in the moment of using it.

Additional information embedded in the Message Routing message response.

MessageId: Identifier of the original message that started the interaction with the gateway.

from: Identification of the sender user (or company).

to: Identification of the recipient user (or company).

Cloud Storage data format

A definition of the data model for storing the description of the different gateways operations, and made them available to other ADVENTURE components (e.g. Data Provisioning and Discovery, Process Designer, etc...) is needed.

This data model will describe the different operations available at a specific gateway instance, its inputs and outputs, and will also contain descriptive data about them (that can also be contained in attached files).

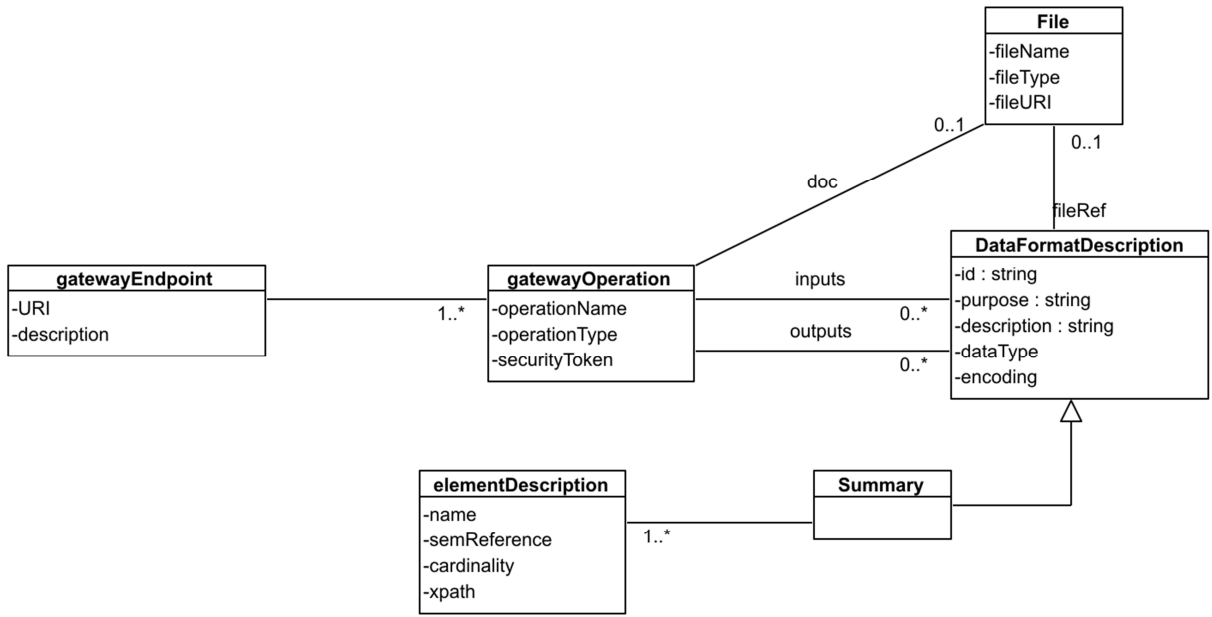


Figure 52 - Gateway services description data model



*Data elements*

gatewayEndpoint: Information about how to reach the specific instance of the gateway. It will be usually a URL or a URI.

gatewayOperation: Description of a gateway specific operation, Including the name of the operation and the behavior Type that implements (i.e. NORMAL / CONTINUOUS).

File: The reference to a file stored in the ADVENTURE system.

DataFormatDescription: Description of the formats of the input and output parameters of the gateway.

SummaryElementDescription: Description of the content elements that the specific summary offered by the gateway provides. It includes a reference to a vocabulary and the xpath to use to extract a specific kind element from the summary.

A document based repository is appropriate to store this specific information, so a noSQL data bucket will be used to store this in the ADVENTURE Cloud Storage System.

The adoption of a noSQL data bucket implies different consequences:

Each company endpoint will be stored as a complete document (facilitating the possibility of importing/exporting/sharing them if necessary).

noSQL repositories have the ability of a non-fixed structure (unlike SQL) in this way different versions of the gateway definition can coexist at the same repository (this ability is inherent to noSQL databases) without problems.

#### **3.6.4.2.1 3rd party Interface (example based on TIE SmartBridge acting as external gateway for sending e.g. EDI payload)**

When dealing with already implemented ways of exchanging information with 3rd party systems, external gateways have to be used linked to the ADVENTURE gateways. This is the case of sending a EDI message to an ERP. EDI works over specific protocols and specific networks (VAN), using specific formats, etc. In this case, the custom connector of the ADVENTURE gateway will consist in an adaptor to call an external mechanism (e.g. middleware such as TIE SmartBridge - TSB) able to deliver EDI messages.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>213</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

As an example, the definition of the interfaces with TSB (.NET based) is provided in this document. This specific implementation will consists on the implementation of SOAP Web Services interface, where two different operations will be exposed.

These interfaces will implement the following operations:

```
TSBResponse sendMessageRequest(TSBRequest)
TSBResponse getStatusRequest(commandID).
```

The Custom Operation Control will initiate the communication, by sending TSB a sendMessageRequest(), including as a parameter an XML document containing all the information needed by TSB to perform the delivery. After that, the custom operation control will ask the TSB for the status until the message will be confirmed, or an error happens. Once confirmed that the message has been delivered, the appropriate information will be returned to the ADVENTURE platform (if an error happens, then it will also be returned with as much information as TSB provides).

Request-Format:

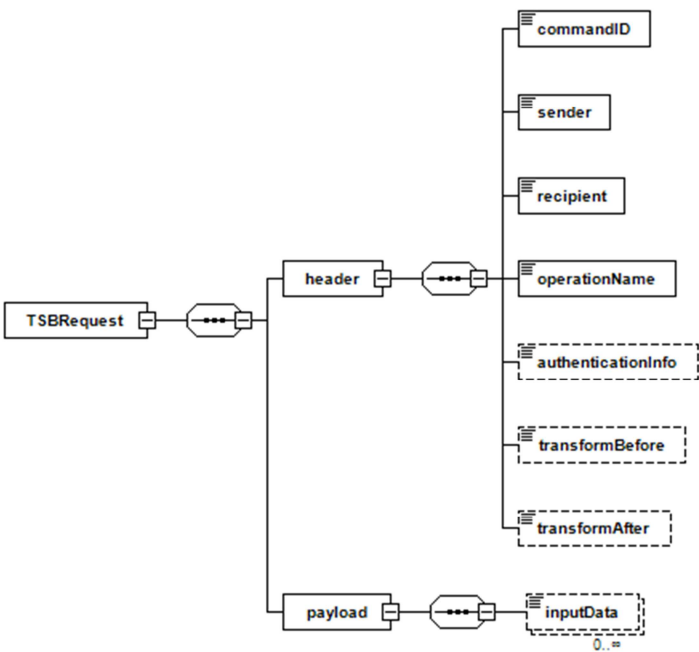


Figure 53 - TSB request schema for document sending.

The data structure is similar to the gatewayRequest one (including the parameters implicitly sent by the Message Routing functionality). The main difference is that it contains an additional identifier (commandID) to assure that the operation is executed at TSB only once, and that the subsequent status queries can be successfully being executed (based on this ID). The ID will be based on UUID so that it can be generated univocally (by the custom components) in a distributed way. It additionally contains the sender and receiver information.

Response-Format:

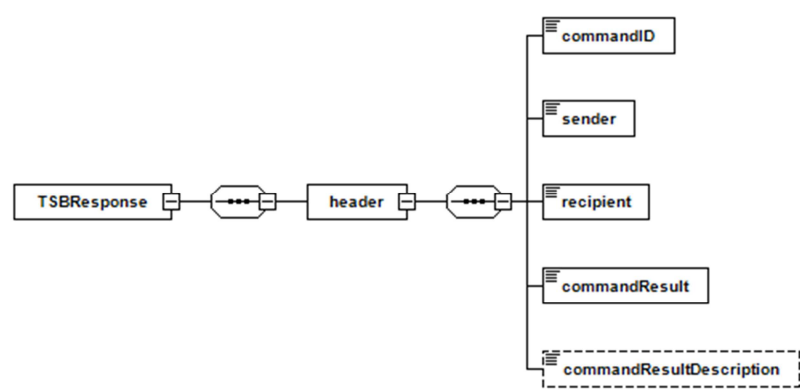


Figure 54 - TSB Response Schema

Data elements

- commandID: UUID of the original message to which the current message is the response. In case it is a push message, the UUID will be generated at the triggering system.
- sender: Identification of the sender user (or company).
- recipient: Identification of the recipient user (or company).
- commandResult: Result of the command executed (OK, ERROR)
- commandResultDescription: If error, this field describes the cause of the error, so that the upper process can be informed.

### 3.6.4.2.2 3rd party Interface (Smart Objects)

The same approach will be used for the communication with a Smart Object via the Smart Object Integration component. Smart Objects are realized on the basis of different, heterogeneous technologies and thus employing different hardware, software, and communication standards and protocols. So, in this context, the custom connector of the ADVENTURE gateway will realize an adaptor to call an external mechanism (here, based on the Smart Object Integration component) possessing the ability to interface with different Smart Objects.

### 3.6.4.3 Reference implementation

Below is presented the method descriptions of a reference implementation of a gateway. Thus represents basic information (on the scope of order exchange and production monitoring) and which is implemented over an Excel file representing a minimal data set for an ordering exchange and manufacturing scenario. The purpose of the reference implementation is to provide a tangible implementation in which the user can test the client side against, and if necessary base their custom implementation on. The scope of this implementation is limited to a few basic and fundamental methods that allow order submission and monitoring, but on the other side, it can provide very small companies with the entry functionality to work as providers of Virtual Factories defined by its customers.

submitOrder

Submits an order to be manufactured by the factory.

```
OrderId submitOrder(Order order)
```

#### Parameters

order: XML structure describing an order. It contains the following information:

- buyerId: id of the order originator
- supplierId: id of the order manufacturer (factory)
- description: textual description of the order
- numberOfUnits: how many items to produce
- List of productIds: id of products that constitute the order

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>216</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- expectedUnitPrice: expected price of one item in the order
- expectedDeliveryDate: expected delivery time of the order

#### *Return Value*

OrderId: string uniquely identifying submitted order

#### *getOrderInfo*

Retrieves accounting and logistics information related to a submitted order.

```
OrderInfo getOrderInfo(OrderId orderId)
```

#### *Parameters*

orderId: string containing the unique id of the order in question

#### *Return Value*

OrderInfo: XML structure describing supplementary order information. It contains the following information:

- acknowledgeReference: string reference of the factory order acknowledge
- dateAcknowledged: date when the submitted order was acknowledged by the factory
- invoiceReference: string reference of the factory invoice
- despatchAdviceReference: string reference of the dispatch advice
- getOrderExecutionInfo

Returns a description of the current state of the execution of a particular order, e.g. it contains data like completion percentage, time to finish the order and time to ship it. This version works on the order level only. To get the execution status of a particular product that is a part of an order, see the corresponding method below.

```
OrderExecutionInfo getOrderExecutionInfo(String orderId)
```

#### *Parameters*

orderId: string containing the unique id of the order in question

*Return Value*

OrderExecutionInfo: XML structure describing the current state of the order execution (or additionally the status of the order related to a specific product). It contains the following information:

- completionPercentage – Current position in the time interval required to finish an order. For calculation it uses time passed since submission of the order and its estimated completion time. For completion percentages of a particular product see corresponding product-based method below.
- timeToFinish – Time by when the order will be finished (in absence of incidents)
- timeToShip – Time when the order will be shipped (in absence of incidents)
- currentIncidents – list of current incidents delaying execution of the order. Every element in the list has the following attributes:
  - startTime – time when incident occurred.
  - estimatedTimeToFix - estimation of when the incident will be fixed.
  - description – textual description of the incident, its cause and what is being done to overcome it
- productIds – list of product ids incident directly impairs
- delay – how long the order will be delayed because of this incident. This delay is on the order level. For instance, consider the following situation: there are two products being manufactured in parallel: product 1 and product 2. Product 2 requires more time to finish than product 1 and order manufacturing can continue only if both products are done. There is an accident that prevents product 1 from being finished. As long as this delay is no longer than time difference in producing product 2 and product 1, this “delay” returned will be zero, as product 1 still has some time to be finished before product 2.

### getProductExecutionInfo

Returns a description of the current state of the execution of a particular product within an order, contains completion percentage and list of incidents.

```
ProductExecutionInfo getProductExecutionInfo(String orderId, String productId)
```

#### Parameters

orderId: string containing the unique id of the order in question.

productId: string containing id of a product which is a part of the order.

#### Return Value

ProductExecutionInfo: XML structure describing the current state of the product execution. It contains the following information:

- completionPercentage – current position in the time interval required to finish the product. For calculation it uses time passed since product manufacturing started and its estimated completion time. If product manufacturing is not started yet, the attribute is zero.
- currentIncidents – list of current incidents currently delaying execution of the product. Every element in the list has the following attributes:
  - startTime – time when incident occurred.
  - estimatedTimeToFix - estimation of when the incident will be fixed.
  - description – textual description of the incident, its cause and what is being done to overcome it
- getProductInfo

This method returns factory capacity parameters related to the given product. Properties like maximum capacity, unit price, carbon footprint, etc. can be obtained from the result.

```
ProductInfo getProductInfo(String supplierId, String productId)
```

*Parameters*

supplierId: string containing unique id of the factory users are querying

productId: string containing unique id of the product in question

*Return Value*

ProductInfo: XML structure describing capacity properties related to this product. It contains the following information:

- maximumCapacity – How many units of the product can be manufactured by this factory within 24 hours (ignoring the current work load and ongoing tasks in its pipeline)
- estimatedCapacity1Week – how many units of the product the factory will be able to produce within 24 hours one week from now (taking into account orders already submitted and production planning, which may use resources required to produce this particular product). E.g. on the 22nd of June.
- estimatedCapacity1Month – how many units of the product the factory will be able to produce within 24 hours one month from now (taking into account orders already submitted and production planning, which may use resources required to produce this particular product). E.g. on the 15nd of July.
- stock – how many units of the product the factory already has in stock
- unitPrice – price of one unit of the product
- carbonFootprint – carbon footprint of producing one unit of product

*informAdventureOnOrderEvent*

This method is implemented by the client and invoked on the client side once a change in the order production occurs. For example, it will be called to notify about completion of an order, new incident, etc. It lets the user know that something has changed in the order production. The functionality will be based on a cron like approach (will monitor the local database every x time) and will push a message on the ADVENTURE system upon a change in the status of the order (e.g. some incident in manufacturing process).

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>220</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



```
void informAdventureOnOrderEvent(OrderId orderId, OrderExecutionInfo orderExecutionInfo)
```

### Parameters

**orderId:** string containing the unique id of the order, execution state of which has changed

**orderExecutionInfo:** XML structure described above and containing information about current state of the order (see the method `getOrderExecutionInfo()`).

### Payload-Example

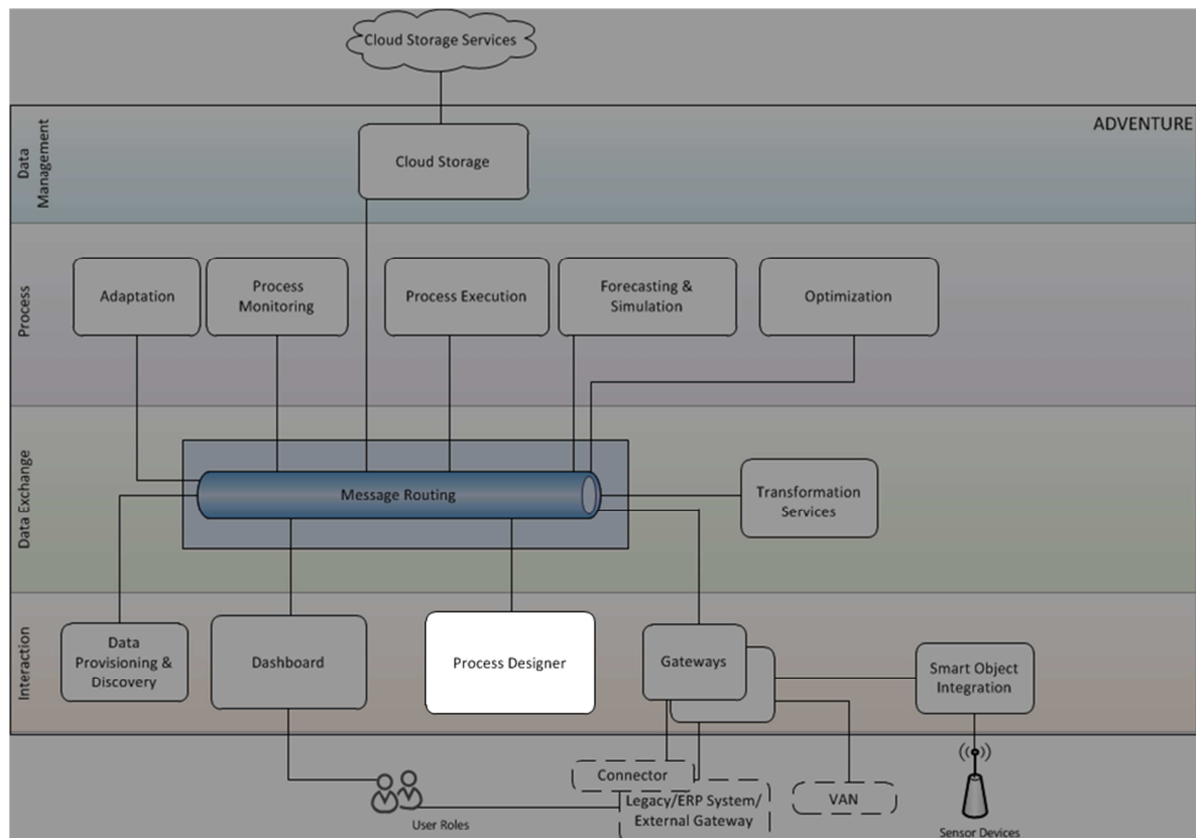
```
<summary>
<info name="string" semRef="http://adventure/summary#orderStatus">
  <info name="Order Number" semRef="http://adventure/summary#orderNum" value="ORD-
PO123-4"/>
  <info name="Completion Percentage"
semRef="http://adventure/summary#completionPercentage" value="40"/>
  <info name="Days to Finish" semRef="http://adventure/summary#timeToFinishDays" value="3"/>
  <info name="Days to Ship" semRef="http://adventure/summary#timeToShipDays" value="4"/>
  <info name="Incidence" semRef="http://adventure/summary#Incidence" value="1">
    <info name="sTime" semRef="http://adventure/summary#incidenceStartTime" value="2012-
07-26 10:00"/>
    <info name="fixTime" semRef="http://adventure/summary#incidenceFixTime" value="2012-
07-27 10:00"/>
    <info name="delay" semRef="http://adventure/summary#incidenceDelayDays" value="1" />
  </info>
</info>
</summary>
```

## 3.6.5 Summary

The Gateways services provide the ADVENTURE platform with the data coming from external systems (3rd party systems and Smart Objects). The Gateway services are presented as a framework, supporting common features, but a tailored implementation for accessing in-house systems will be necessary by the company developers, in order to implement the company gateway. A reference implementation as well as implementation libraries will be delivered by the ADVENTURE project so that they could be used by the developers for the implementation of the specific gateways.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>221</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

# Process Designer



## 3.7 Process Designer

Abstract:

The Process Designer module is a process model graphical editor, able to design manufacturing processes. It is the key tool in the stage of designing Virtual Factories. Companies will define their processes based on skill and technical requirements.

The process designer allows the assignment of partners or partner services to a specific task, as the results of searching for the appropriate partner profile or partner service.

It collaborates with assistive tools like simulation and optimization in order to provide ADVENTURE brokers with the core functionality to accomplish a successful Process Model design.

### 3.7.1 Major Design Decisions

The ADVENTURE Process Designer provides companies with the functionalities needed in order to design manufacturing processes by combining production steps, partner services and manufacturing resources. The Process Designer is composed of the Process Model Management (PMM) and Process Model Editor (PME) sub-components:

#### Process Model Management (PMM)

In order to facilitate the management of the different process models that ADVENTURE broker can design, the Process Designer will expose a Process Model Management (PMM) feature. The PMM will allow the ADVENTURE brokers to create/edit/find and share their own process designs, so that they can be put into execution, simulation, optimization and/or reused by other ADVENTURE brokers.

The PMM will also feature process templates management that will allow ADVENTURE Process Model Editor's users to share best practices in modeling processes for example for a specific manufacturing domain, so that brokers can reuse these templates to accelerate or , improve or make their designs compliant.

#### Process Model Editor (PME)

The Process Model Editor (PME) will generate process models compliant with the BPMN (Business Process Model and Notation) 2.0 notation and semantics. BPMN 2.0 (<http://www.omg.org/spec/BPMN/2.0/>) is an OMG specification for designing and

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>223</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

executing process models. It defines different conformance levels (Process Modeling Conformance, Process Execution Conformance, BPEL Process Execution Conformance, etc.) depending on the usage.

The process model data format is the data exchanging format between the Process Model Editor and the Process Execution module. Therefore, a specific conformance level is needed in ADVENTURE so that the communication between the editor and the execution environment will be successful and all the designed activities could be executed at the run time. The choice therefore is for BPMN 2.0 Process Execution Conformance compliance level, which is the executable BPMN subset (as defined in the OMG BPMN 2.0. standard). Process models created by 3rd party tools and imported in the Process Designer also need to be compliant with this conformance level of BPMN 2.0 in order to be managed and processed by ADVENTURE toolset.

In order to improve the usability of the editor and to facilitate the modeling by non-expert users, specific ADVENTURE elements will be introduced as part of the graphical notation, simplifying the design of complex common interactions. Based on that, specific extensions of BPMN2.0 will be developed (e.g. in order to be able to associate Carbon footprint maximum value per task, facilitate frequent ADVENTURE modeling patterns, etc.). These extensions will be specific to ADVENTURE, and will be easily located in the BPMN 2.0 XML formats, because all of them will pertain to a specific ADVENTURE namespace (e.g. <http://adventure.org/ns/#carbonPrintMax>) and visual representation. The extensions may also require specific capabilities of the execution environment (if they need some specific behavior for execution). As a prerequisite for these custom extensions there should always be a simple way back to the standard model, and extensions must comply with the BPMN standard extension protocols.

The PMM will act as a launching platform for the Forecasting, Simulation and Optimization functionalities. Based on the selection of an existing Process model, these ADVENTURE modules can be called from within the PME (in case of simulation or optimization of individual process activities or the whole process model) or the PMM (when the complete process model is simulated or optimized). In order to provide a graphical representation of the process models at these functionalities, the Process Designer will save the graphical representation of the model into SVG format (Scalable Vector Graphics (SVG) is a family of specifications of a standard, open XML-based file format for two-dimensional vector graphics developed and maintained

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>224</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

by the W3C SVG Working Group). In this way these modules will be able to manipulate and enrich the graphical representation of the model, adding the necessary information to it. The process editor will also expose a read only behavior mode, and calling interfaces, so that other modules can decide to use this API (or the SVG option) to layer and represent information on top of a specific data model for visualization (e.g. Monitoring showing the current execution status of a process model instance).

The PME will also allow the users to define script based tasks for advanced data management and logic. The scripting language choice is JavaScript, since it fulfills the following requirements:

- It is a common scripting language shared by the script editor in the PD and the SPE
- It will be used for simple manipulation of data elements, so it should be compliant with JSON and XML (as PME will store internally the data in these formats).
- It should be easy to be used and should easily support debug in editor.
- Advanced (3<sup>d</sup> party) editors will be used (e.g. ACE or CodeMirror) to help the user in the script elaboration.
- Script tasks should not implement connectivity (via soap, rest, to databases) themselves and should not use other JavaScript libraries.

Taking into account the previous constraints, JavaScript will be used as default scripting language for ADVENTURE.

The PME will save the process models into the ADVENTURE Cloud Storage. A process model definition will contain metadata describing the model and its purpose, as well as a link to the XML file containing the BPMN 2.0 description, and an optional graphical representation (SVG) of the model (so that it could be used by other ADVENTURE modules).

### 3.7.2 Technology Comparison and Selection

In order to select a technology that can provide a solid base for the implementation of the Process Designer, a SOTA analysis regarding Process Model editors has been performed. In order to consider all the different aspects that can be important for the editor, the analysis has been made based on the criteria defined in the Functional Specification (D3.2).

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>225</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

In the next section, the results of the SOTA analysis are introduced, as well as a short description of each of the options taken into consideration.

### 3.7.2.1 Selection Criteria

The selection criteria for Process Designer technologies were defined in the D3.2 Functional Specification, section 5.2.5, page 70.

The specific parameters are related to main requirements of the Process Designer in terms of relationship with other components (e.g. the Data Provisioning and Discovery component or Process Execution component), but also specific parameters from the point of view of usability and other functional and non-functional requirements.

These parameters are defined as follows:

**Generation of executable models:** This criterion refers to the ability of the editor in exporting the designs with all the necessary operational details, so that it can be executed by an external process execution engine. Some options are saving the model into executable formats like BPMN 2.0, jPDL , etc...)

**Parameter mapping:** When connecting several process steps, it is not enough to simply draw a line between 2 boxes. Instead, the process editor will need to be able to specify the parameters, resulting values and scope for each process step.

**Support for Multi-Binding:** At design time, it has to be possible to assign a process step to a group (0..\*) of company services (e.g. selecting these that match a specific profile, capacity, etc.). Before (or maybe during) execution, it will be necessary to select a specific service provider and bind it to the process step as the selected and executable one.

**BPMN elements support:** The process editor will provide basic modeling element (tasks, gateways, events) so that it will be flexible enough to cope with complex process model. At the same time the editor has to contain the elements that exist in the most popular process execution languages, in order to facilitate the execution of the modeled processes.

**Decision criteria definition:** In order to have better control for exceptions or errors, it will be necessary to capture events during the execution of a specific task. Further, based on the decision criteria specified, a concrete action will be performed. This will facilitate the definition of exception handling, for example what will happen if a factory is unavailable, or what to do if any other condition is not fulfilled.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>226</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Calling parameter definition:** During the execution of a process, it might be necessary to pass some static parameters to a specific activity (e.g. the unique ID of the process) as well as some dynamic parameters (e.g. the current date).

**Usability:** The selected process editor has to be easy to use from the user's point of view, as a common case is supposed to be a non-technical person.

**Script/Code editor:** The selected technology should be able to provide an advanced script editor, so that the user can define specific behaviors of activities, beyond what might be limited with visual UI approach.

**Model validator:** The selected technology should be able to provide a syntax model validator in order to facilitate the design of proper and coherent process models.

**Basic event support:** The process model editor needs to support the basic elements to define different kinds of events.

**Support rules definition:** A Logic rules mechanism will allow performing logic inference and evaluation of conditions. This feature will provide flexibility and power in the evaluation of conditions. The process editor will provide a mechanism to allow the user to define these rules.

**Human task support:** This parameter is referring to the ability of the process designer to define and model tasks that need some kind of human interaction (e.g. approval of documents).

**Strictly block structured process logic:** This criterion is related to the way (language) that Process Designer uses to export the process model in a suitable for execution format. Block structured execution logic can be very easily converted to graphs and very easily checked for correctness and consistency.

**Export to image/vector formats (e.g. SVG png, etc.):** This specific criterion is related to the ability of the selected process designer tool to export the process model in a graphical format, so that it can be used by other ADVENTURE modules for enriching their representation.

### 3.7.2.2 Possible Technologies

During the SOTA analysis, different kinds of process designer tools have been evaluated considering the set of parameters coming from the requirements and Functional Specification. As part of the Functional Specification, a set of evaluation conditions were defined, so that they can be considered for tool selection. Apart from these criteria (described in the previous section), other technical parameters have

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>227</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

also been taken into consideration, such as the alignment to the overall envisaged technology for ADVENTURE and the modules to which it has dependencies.

The SOTA analysis for ADVENTURE was performed in two phases. In the first one, different technologies have been analyzed (including e.g. desktop process modelers). The objective of this first phase was to identify the different characteristics that the different process editors offer. In that way, the team could identify and specify technical criteria not introduced as part of D3.2 evaluation. In the second phase, a more in-depth evaluation of the Web based editors has been performed taking into account these enhanced criteria. The following Process designer tools have been evaluated as part of the second phase (Web designers) of the technology selection procedure:

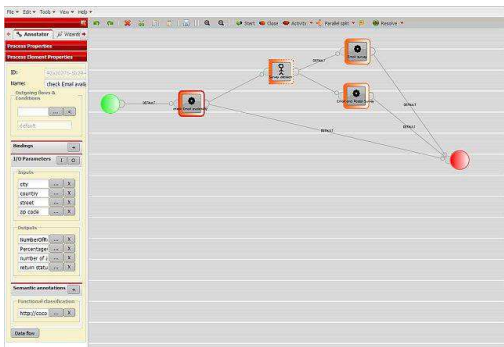
**Orchestra** (<http://orchestra.ow2.org> )

Orchestra is a solution to handle long-running, service oriented processes and providing orchestration functionalities to handle complex business processes. Its execution environment is based on the OASIS standard BPEL (Business Process Execution Language). The process designer is based on PHP and Flash, and support BPML 2.0 notation elements.



**SOA4ALL Composer** (<http://sourceforge.net/projects/soa4allcomposer/> )

The SOA4All Composer is a RIA (Rich Internet Application) for process modeling, built for the lightweight process modeling language of SOA4All (LPML). The Composer allows non-technical users to compose executable processes from semantically annotated Web services. It was implemented in the SOA4ALL EU project (<http://www.soa4all.eu>).



**CPEE Process Editor**

D3.3-Technical-Specification
Copyright © ADVENTURE Project Consortium. All Rights Reserved.

Author:  
Project:

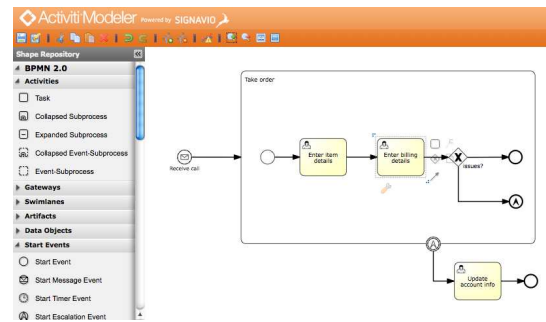
2



CPEE is a modular, service oriented workflow execution engine targeted at researchers, developers and system administrators. Its goal is to provide a simple and lightweight, but also more powerful alternative to existing execution engines. It provides a lightweight execution oriented Web designer in order to allow the user to model their specific workflows.

### Signavio Core Components / Activiti modeler

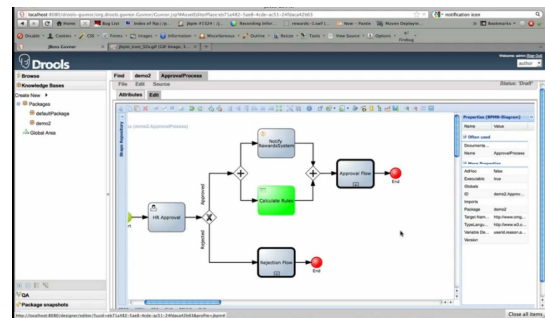
(<http://code.google.com/p/signavio-core-components/>). The Signavio Core Components enable the basic creation and interchange of BPMN 2.0 diagrams. It includes a lightweight file-based backend repository to save and manage the designed diagrams.



Signavio Core Components inherits its development from a previous Open Source project called Oryx (<http://bpt.hpi.uni-potsdam.de/Oryx>). At this moment, there are several initiatives using Signavio Core Components as Process Editor, one of them is Activiti (<http://activiti.org>). The Activiti workflow engine uses a Signavio based process editor to allow the users design their specific process models that are afterwards executed at the Activity Execution engine. The Signavio Core Components project contains the technical foundations for editing process diagrams.

### jBPM Web Designer 2.0 (<http://www.jboss.org/jbpm/>)

The jBPM Web Designer is an open-source BPMN2 editor to allow the creation of business processes. The jBPM Designer focuses not only on modeling, but also on creating executable business processes. jBPMN is integrated with Drools Guvnor as a repository for business asset creation (business processes, business rules, task forms, etc). jBPM Web designer is also based on Signavio and Oryx projects. It extends the features of Signavio to make it closer to the jBPM execution environment, improving it with extended functionalities like import BPMN2 models, exporting models to graphical formats, etc.



*Table 12- Technology selection criteria and comparison of technologies for Process Designer*

Parameter	Importance (--- +/- +++)	Orchestra	SOA4ALL Process Editor	CPEE Process Editor	Signavio Core Components / Activity modeler	jBPM Web Designer 2.0
Generic Parameters						
Maturity & Stability	++	++	+/-	+/-	++	++
Regularly Updated	+	+	-	+/-	++	++
Technical Up-to-Datedness / Appeal	+	+	++	++	++	++
Open Source	+++	YES	YES	YES	YES	YES
Non-Infecting	+++	YES	YES	YES	YES	YES
Code-Quality	++	++	+/-	+/-	++	++
Extensibility	++	+	+	+	++	+++
Community	+++	+	-	-	++	+++
Performance	+/-	++	++	++	++	++
Reuse of existing developments	+/-	+/-	+/-	+/-	++	++
EU project origin	+/-	NO	YES	NO	NO	NO
Platform (Portability)	+	++	++	++	++	++
Open Standards Compliance	++	+	+	+	+	+
Interoperability	++	++	-	-	+	++
Specific Parameters						

Generation of executable models	+++	++	++	++	+	++
Web based editor	+++	++	++	++	++	++
Parameter mapping	+	+	+	+	+	+
Support for Multi-Binding	++	-	-	-	-	-
BPMN elements support	++	++	+	+	+++	+++
Decision criteria definition	+	+	+	++	++	++
Calling parameter definition	+	+	+	+	+	+
Usability	+++	+	++	++	+++	+++
Model validator	++	+	+	++	++	++
Script/code editor	+	++	++	++	++	++
Basic events support.	++	++	++	++	++	++
Support rules definition	+/-	+	+	+	+	+++
Human task support	+/-	+	+	+	+	+++
Strictly block structured process logic	+/-	+	+	+	+	+
Export to image/vector formats (SVG, png, ...)	+	++	+	+	+	+++

### 3.7.2.3 Technology Selection

From the result of the SOTA analysis, jBPM Web Designer shows to be the most complete tool to base the ADVENTURE developments on. It has a lot of advantages and functionalities needed, but on the other side, it has a very strong dependency on

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>231</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

other JBOSS technologies (JBossAS, Guvnor, Drools, etc.). It can be complex to adapt or remove these technologies, because their functionality exceeds the specific purposes of ADVENTURE. Nevertheless it is a very dynamic community in which collaboration seems quite easy, and innovative features (like Web voice capabilities) are added constantly to the Web process editor that makes the effort worth it. Second in the list is Signavio Core Components. It provides almost all the features needed for the ADVENTURE project, with no strong dependences in terms of technology, but the company maintaining the source code has informed about a new commercial strategy that will affect the Open Source distribution in short term. Both Signavio Core Elements and jBPM Web Design, are branches of the same source - Open Source Projects (Oryx) and maintain the core features from it. So the decision will follow this special relationship between these two tools: The implementation will be based on jBPM Web designer, complemented with source code coming from Signavio Core Components when needed (the licenses of both developments are compliant and allow this).

#### 3.7.2.4 Missing Elements and Implementation Needs

The technology of choice for the implementation of Process Designer provides a good basis for realizing the defined requirements. It offers a number of out-of-the-box features for business process modeling. However, the base software needs to be adapted, extended or improved to cover also the specific scenarios and requirements in ADVENTURE.

The main enhancement needs are in the following areas:

**Process model management:** The selected technology provides a basic process management facility that needs to be extended to meet ADVENTURE requirements. For example:

- The UI will be extended with search, sort and filter functions for process models, specific layout of managed process views (e.g. additional attributes like process type and status, authoring info).
- The whole component will be made privacy-aware.
- The load/store functionality will be enhanced to load/store process-related artifacts and metadata from/in the respective ADVENTURE modules. (JBoss technologies attached to jBPM designer for storage has to substituted and/or bridged with ADVENTURE components).

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>232</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- Export/import functionality will be added.
- The component will be integrated with other modules like simulation and optimization as a starting point triggering them.
- New functionality for managing the executable processes lifecycle (start, pause, stop, edit, delete), monitoring the running instances and sharing will be added.
- Process model attributes shown in the list will be customized.
- Starting new process model designs from templates or from scratch will be implemented.

**Modeling interaction enhancement:** The process design with jBPM designer does not provide for interactive drill down into possibly hierarchical elements like BPMN2.0 call activities (sub-processes). This will be enhanced by the development of the required interactive experience.

**Editor stencil improvements:** In order to make the usage of ADVENTURE easier, a set of new graphical ADVENTURE related elements will be introduced at the process editor. These elements will be like shortcuts for common and/or complex specific ADVENTURE tasks, so that the user just can use them instead of building complex sub-processes. Attached to these elements, it will be necessary to create specific data extensions so that they could be transmitted and interpreted correctly by the execution environment. Some of these improvements are related to:

- Associated Activity and transformation.
- Transformation Activities.
- Data extraction activities.
- Correlation Activities.

**Process model and activity modeling element attributes enhancement:** The standard set of attributes has to be extended and the extended set will be modeled as BPMN 2.0 extensions. Process models will include attributes defining the domain of the process, non-functional properties like total cost cap, maximum carbon footprint constraint, maximum duration or others such as main process stamp (multiple times instantiable process), all of which can be selected from a predefined, extensible set. Activity attributes will be extended to contain a compliance specification for the task, based on manufacturing business properties like maximum duration, maximum total cost, maximum carbon footprint, quality metrics as well as attachments that can complete the specification. Those terms are shared with the descriptions of partner services in the Data provisioning and Discovery information

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>233</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

space for realizing matchmaking for compliant partner services in order to use the same, shared information model of terms. Annotation editors that provide access to that shared model will be developed and integrated with the attributes and process UI.

Further, the input and output specification-related attributes of service activities will be enhanced to meet the ADVENTURE-specific options. The great number of attributes in the basis editor version is too large for good end-user experience, so a suitable categorization or live search/filtering to help designers to easily make their way through it will be developed.

**Partner recommendations:** Based on the metadata provided by the broker designing the process, in the form of requirements, constraints and specifications, and the metadata provided in the member profiles, the Data Provisioning and Discovery module will provide a list of recommended partners matching the modeled activity specification. This integration is realized by the corresponding editors for annotating the process models and the recommendations and search dialogs from which designers can select and assign services to activities.

**Enhanced BPMN 2.0 model definition parsing and saving:** Although the selected editor is able to save BPMN 2.0 XML models, due to the enhancements of the supported model elements and attributes, the standard model has to be extended as well, in order to contain the ADVENTURE specific characteristics and components.

**Enhanced graphics saving:** In order to allow external components to be able to easily share the graphical representation of a process model, saving into a vector graphical format will be used. This will be provided by the selected tool, but needs to be improved in a way that the saved graphical elements have a relationship with the BPMN elements. None of the editors studied supports this, so the saving process of the graphical file needs to be improved.

**Read-only interface:** This will be an alternative to the previous point. In case high levels of interaction are needed with a process model for visualizing it and adding some other information in the model (like status, etc...), a read-only mode will be implemented for the Process Designer. In this mode the Process Designer will expose a JavaScript API, so that the external modules can use it for drawing new graphical elements on top of the loaded (in view only mode) process model. The read only view will be configured to either show only the process model canvas or also the attributes palette.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>234</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**JavaScript editor:** In order to provide the ADVENTURE broker with a scripting functionality (e.g. to support script tasks), a JavaScript editor will be integrated within the Process Editor. This editor will provide advanced features (such as auto-completion, syntax coloring, process context variables, etc.) to facilitate the definition of scripts in script tasks or guard condition expressions.

3.7.3 Technical Component Specification

3.7.3.1 Component Structure

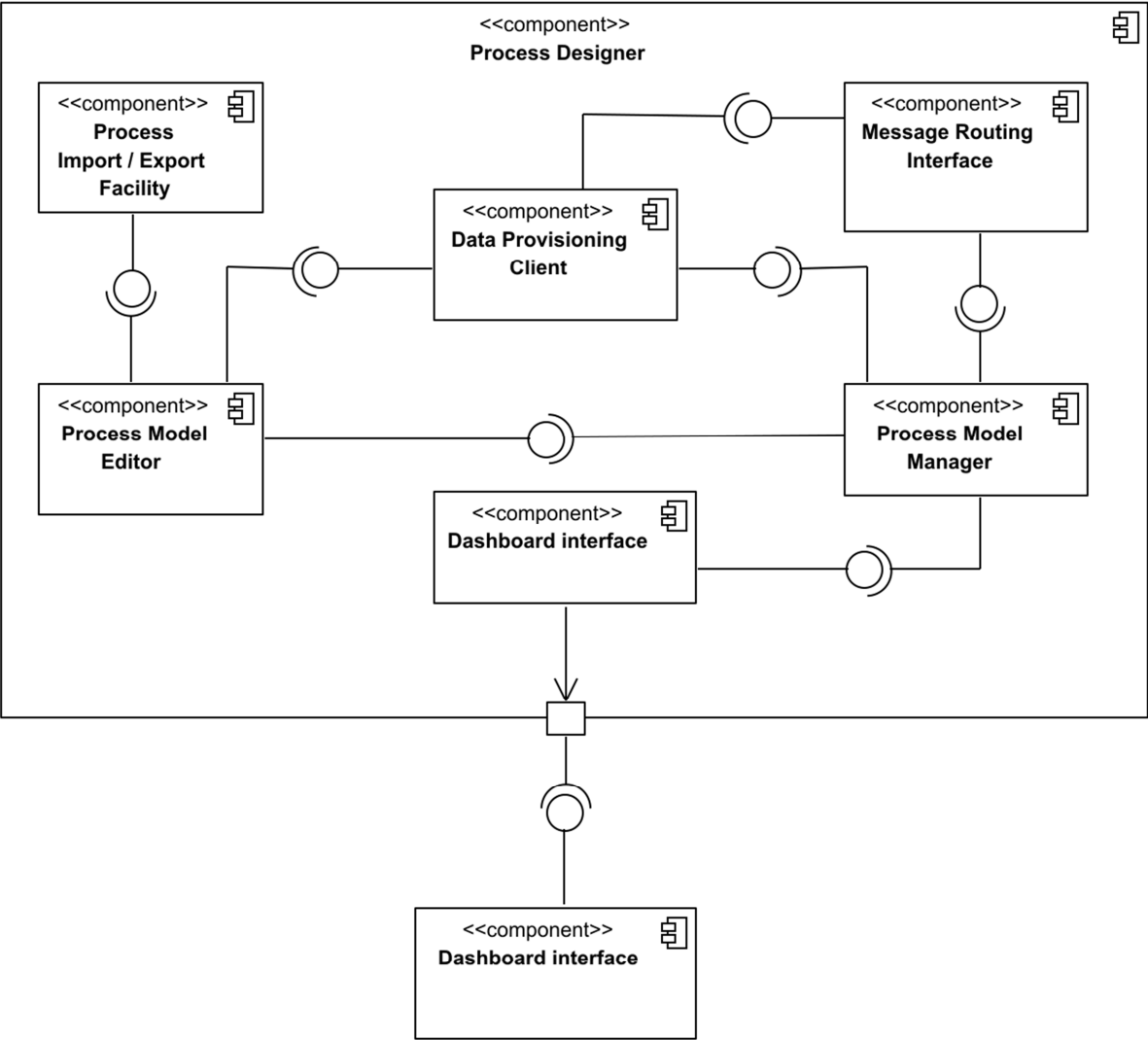


Figure 55 - Process Designer structure

The ADVENTURE Process Editor consists of the following components:

### **Process Model Manager**

The Process Model Manager (PMM) is a UI component that allows the ADVENTURE broker to manage its process models. It acts as a model repository, allowing also the brokers to share their process models in such a way that they can be included by other brokers into their Virtual Factories design. The PMM is the entry point to the Process Editor. It can also act as launcher for specific ADVENTURE modules related to the improvement of the process model features, like simulation or optimization.

### **Process Editor**

The Process Editor is a client based graphical editor based on the selected tool (see previous section). It contains all the core graphical and data elements needed by the user to design a process model. The Process Editor is linked to the rest of the components in the technical design to provide data to/from the UI, so that the ADVENTURE brokers can design their processes successfully. For example, the Process Editor uses the DPD (Data Provisioning and Discovery module) in order to be able to configure specific activities in the process model, with specific partners' services and gateways connected to them, so that the execution of the specific activity will consist and will be monitored through the interaction of the system with the specific gateway service.

### **Dashboard Interface**

The Dashboard Interface is an integration utility that allows the Process Designer to interact with the UIs of the rest of the ADVENTURE modules. By means of this interface, the Process Editor will be aware of the ADVENTURE broker that is authenticated and is currently using the editor. It will also allow the Process Editor to interact with other modules offering a graphical interface (e.g. redirect the end user to the Monitoring graphical interface to check out the status of the running processes).

### **Process Input/Output Facility**

This subcomponent will deal with the import, export and saving features of the Process Editor. It will implement both the standard elements and custom ADVENTURE elements (BPMN2.0 extensions) and manage the input/output operations in this format.

### **Data Provisioning Client**

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>236</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



The Data Provisioning and Discovery component will be accessed from the Process Designer in order to associate activities with partner services offered by companies. The dependency to DPD is realized by the Data provisioning client. The Data provisioning client is a set of subcomponents that will provide the Process Editor with the graphical interfaces and functionality that will allow services or company selection, gateways selection and configuration. It will also contribute with annotation editors that can be used to add classification terms or other concepts to the process model to enrich the metadata in it and to facilitate the service-for-activity matchmaking process. The client will access DPD module, as well as the Cloud Storage system to retrieve this information.

### Message Routing Interface

The Message Routing interface is the main communication interface to the other modules in the ADVENTURE platform. All the internal components will make use of it when they need to communicate with a different ADVENTURE module (via the Message Routing component). Process Designer will make use of it to communicate with the modules that it depends on.

#### 3.7.3.2 Internal Sequence Diagrams

This section will provide an insight of the internal component communication sequence based on several example scenarios. It's extracted from the use cases of the Process Designer component described in D3.2. These internal sequence diagrams are showing the main representative set of the operations of the component.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>237</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Processes Management

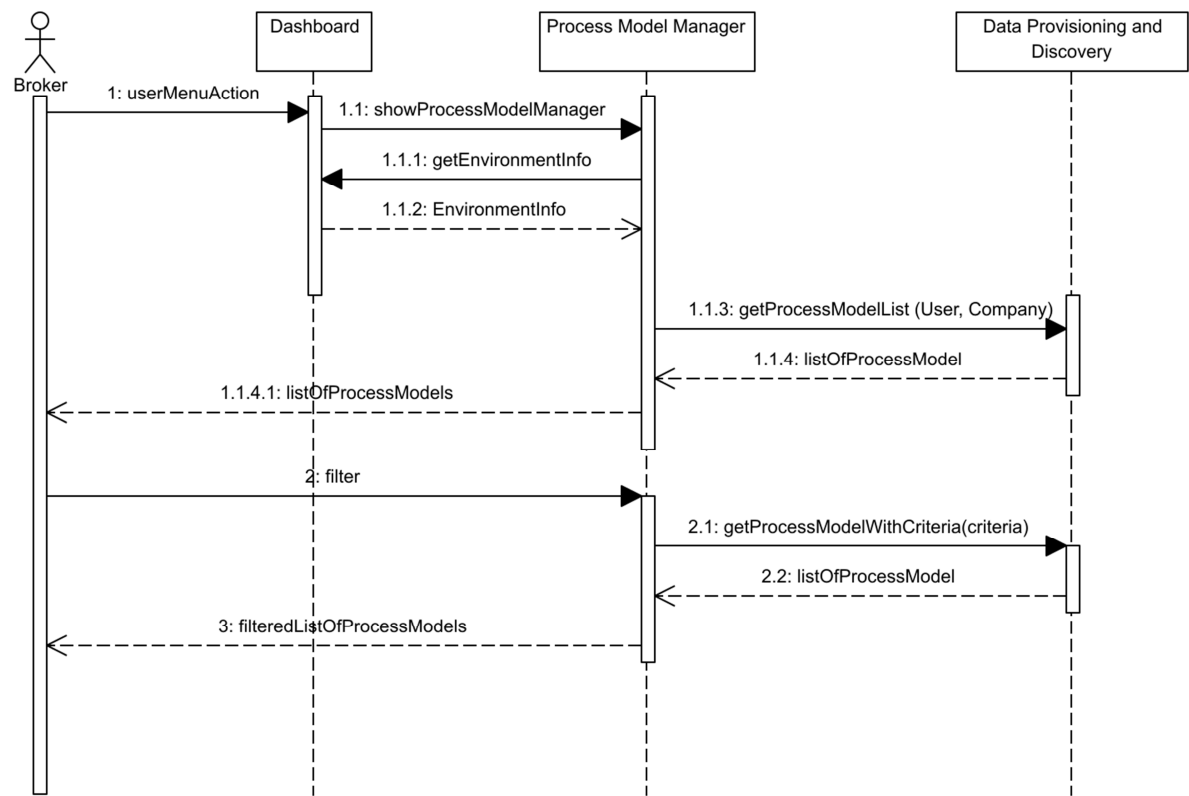


Figure 56 - Show Process Model Manager, list Process Models and filter sequence diagram

This sequence diagram describes the steps that are followed to open the PMM and represent a list of the available process models for the ADVENTURE broker using the Dashboard interface.

The Dashboard shows the PMM, as a response to a broker action, and then all the environment information is requested from the Dashboard component. As PMM will be implemented as a Portlet, it will use this interface to retrieve the information from the Portlet container. Once this environment information is retrieved, the PMM will use it (e.g. the identification of the authenticated broker) to query for the list of process models that this user can access (own models, models shared by other users, etc.). The list will be represented as table with different information about the process model itself. The information represented in this table has to be retrieved from the DPD module which extracts the processes and their respective metadata from the Cloud Storage.

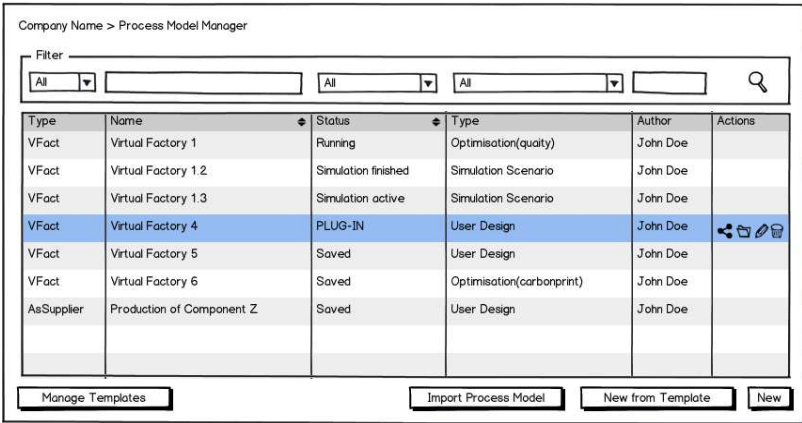


Figure 57 - Mock-up of the PMM main page

At this stage the user can provide filter information (on the already retrieved result set) in order to search for a specific process model or a group of process models, and perform further actions on it/them. Filtering actions will also use the DPD and the Cloud Storage System, respectively to perform the queries, in case of paging of the result set).

Create New Process Model

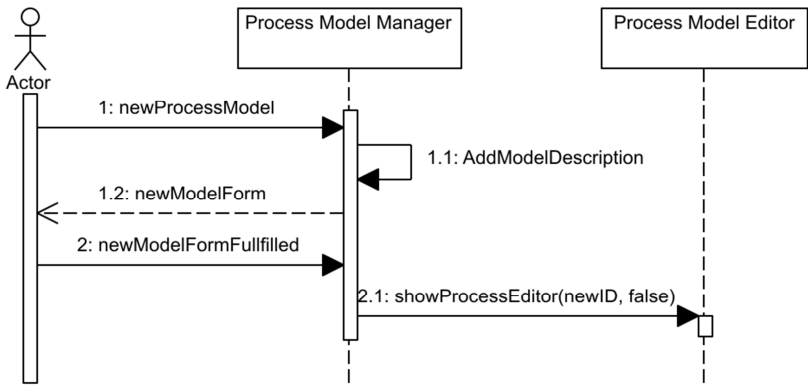


Figure 58 - New process model sequence diagram

The diagram above describes the steps of opening the Process Model Designer (editor) to start designing a new model from scratch. To perform this action, the user (ADVENTURE broker) will select the "new" option at the PMM. This will show a dialog, in which information about the new process model to be designed can be described (name, purpose, domain, as well as non-functional attributes like CO<sub>2</sub> maximum amount, maximum manufacturing price, maximum delivery time, etc.).

This information can also be edited within the PME.

Once the user presses the "OK" button of the dialog, the PME is opened with a blank canvas, so that the broker can start dragging graphical elements on to it and around it.

Create New Process Model From Template

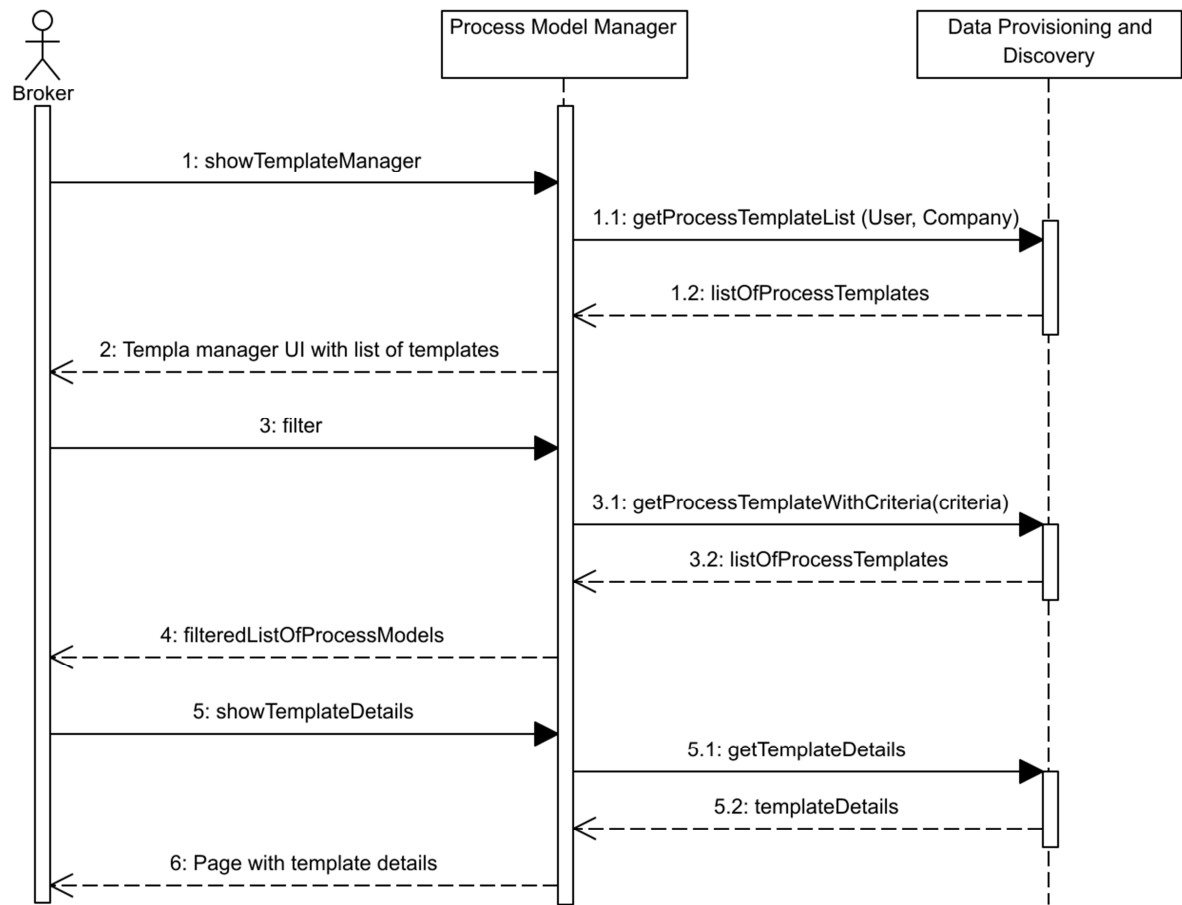


Figure 59 - Show Process Template Manager and related actions sequence diagram

As an extension of the previous sequence diagram, the user can decide to select one of the process templates from the list the user is viewing in the PMM to serve as a base for the design of a new process model. A process model that uses a template as a base, becomes independent of the template, in such a way, that if the template changes, the process model is never affected.

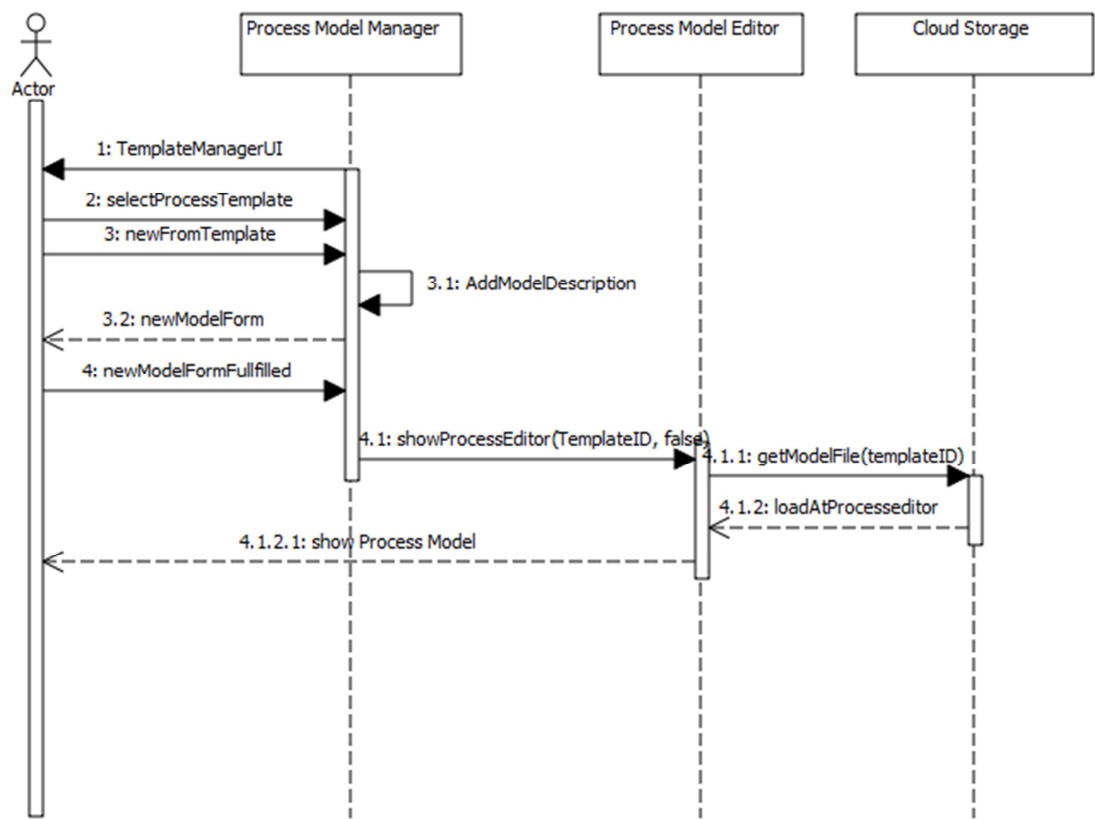


Figure 60 - New process model from template sequence diagram

Once the user selects a template from the list, it initiates the newModelFromTemplate action that will show a form dialogue to let the user to describe the new model. When the form is filled in and the user presses the "OK" button, the process model editor will be displayed in edit mode will retrieve the process template design from the Cloud Storage system (through the DPD) and will load it so that the user can start editing and configuring it.

Edit Process Model

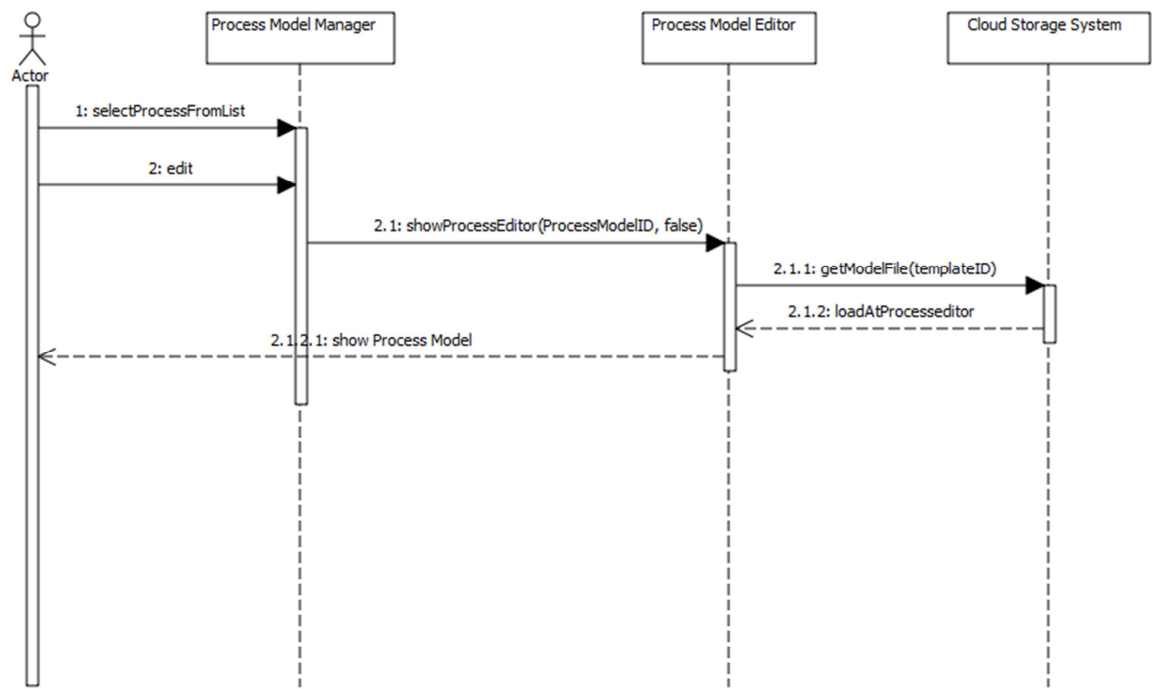


Figure 61 - Edit process model sequence diagram

The edit process model sequence diagram describes the interactions between the PMM, the PME and the CSS (Cloud Storage System) when the ADVENTURE broker selects a specific process model at the PMM in order to edit it.

In this case, the PME is called, and as parameters it has the reference to the BPMN2.0 file of the process to be loaded, and a Boolean parameter showing whether the editor is loaded in editing or read-only mode. The editor loads the file and shows the process model graphical representation to the user.

Save Process Model

The saveProcessModel action is triggered from the process editor, and interacts with the Cloud Storage to persist the information regarding the edited process model. It will save all the information about the model (BPMN2.0 XML file, SVG representation, metadata fetched from the edited model, etc...). The process includes also interaction with the DPD module (not shown on the diagram for simplicity) which extracts the activity assignments and other metadata from the process model to keep track of the organizations activities.

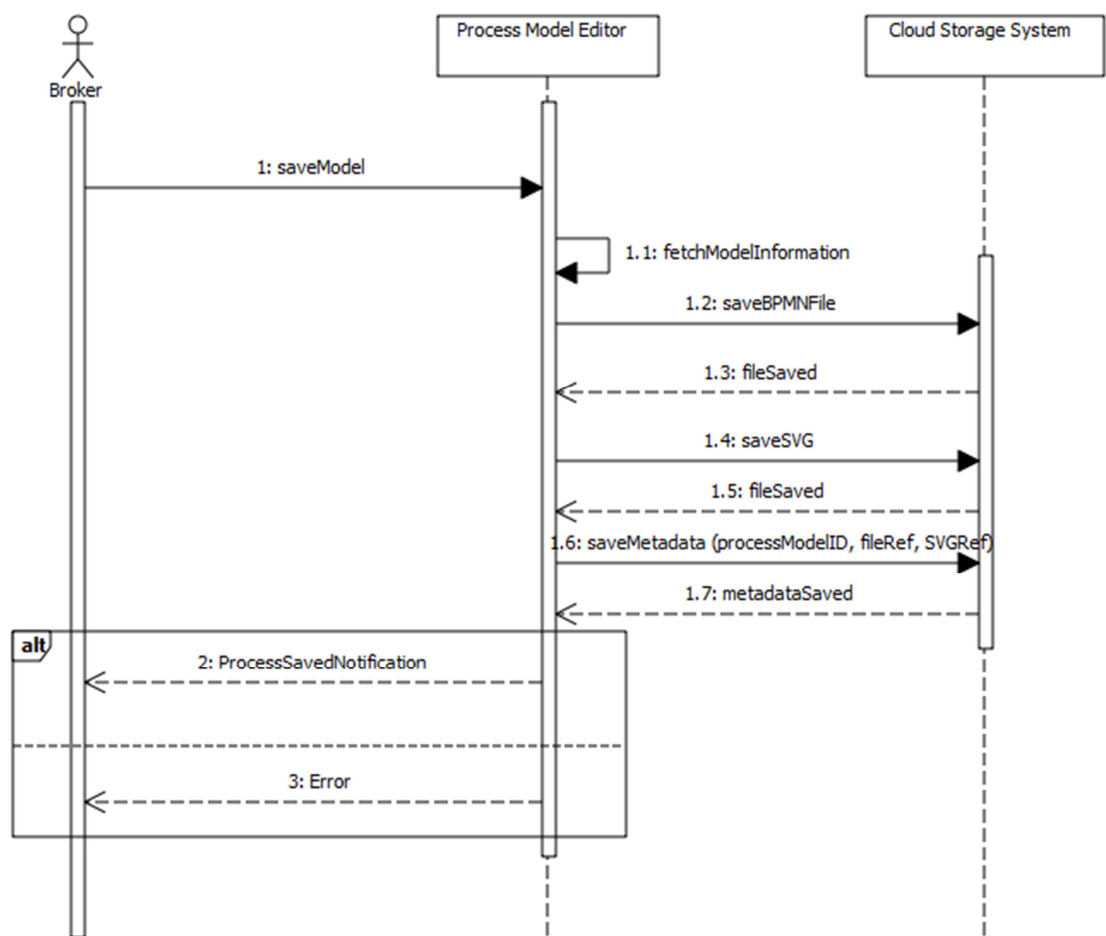


Figure 62 - Save process model sequence diagram

Share Process Model

The shareProcessModel is an action triggered by the user in order to allow other users/brokers to exploit the designed process model (including all the configurations of services) and include it as a sub-process in new designed models. This will allow the rapid design of complex processes based on the "sub-processes" designed by other ADVENTURE members.

When sharing a process model, the ADVENTURE broker will decide with whom this process model will be shared, so different options will be offered to the user regarding that (individual companies, companies in a certain domain, partner companies, everybody, etc.).

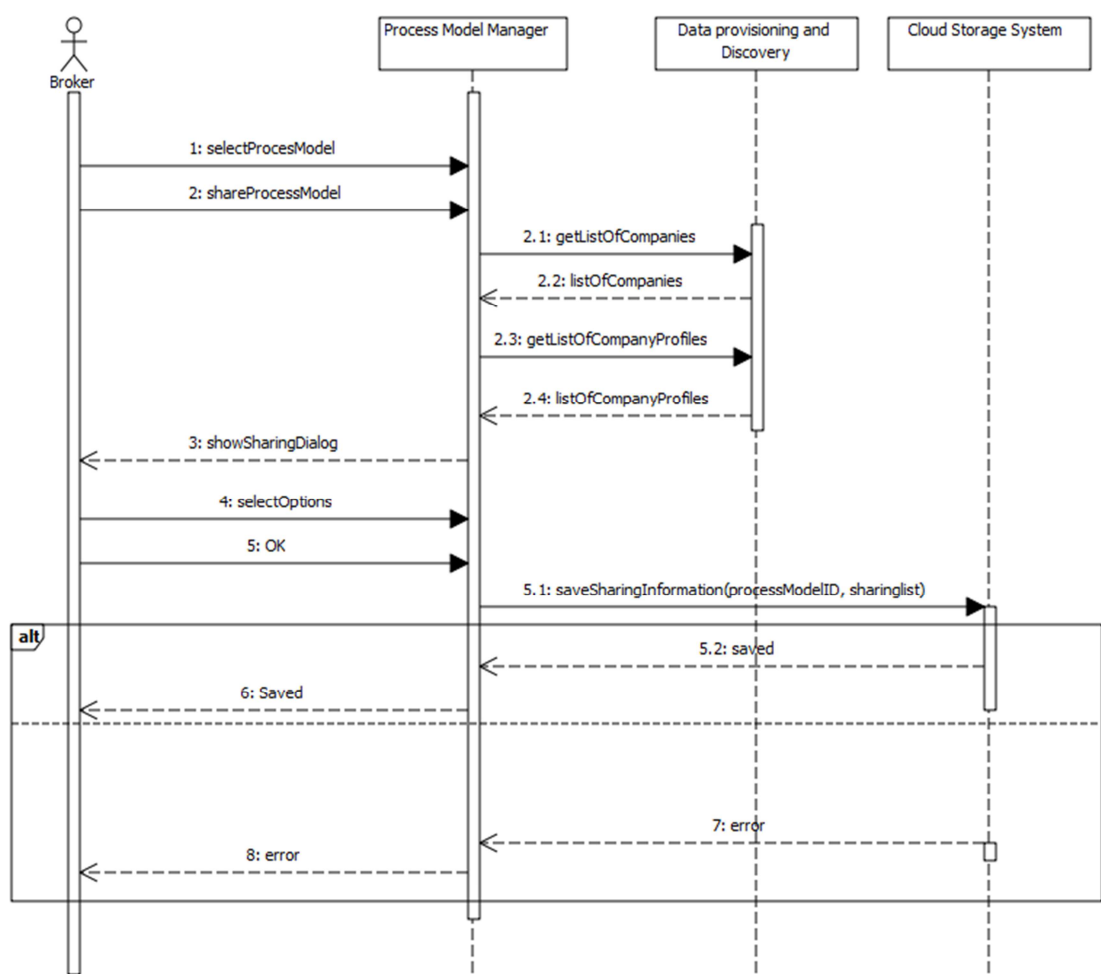


Figure 63 - Share process model sequence diagram

Add New Activity, Assign Company Service To Activity

Once the Broker is in the PME (in editing mode) they will have different tools to design the Process Model. An important step in the design of a Process Model is to add a new activity to it, and complete its configuration, so that it can be executed afterwards. In ADVENTURE, an activity will usually be associated to a partner organization's service operation.

In addition to configuring the standard BPMN attributes, the configuration of a task generally implies:

- The configuration of manufacturing-related, non-functional properties (type of task, CO<sub>2</sub> limits, time limits, price limits, etc.) that have to be taken into account for validation (and in optimization) processes. The ADVENTURE-specific non-



functional properties are defined by a shared vocabulary with the optimization component and DPD.

- Configuration of functional properties that describes the specific company services that will be executed from operational perspective,
- The characterization of the service and resulting product that should be developed, or what service (business service) should be implemented.
- There might be also other operational configurations, such as activity status which can be used to specify what kind of status of this task the system will tack at runtime and how. For example, the broker might consider as important different statuses such as order status, stock levels, smart objects statuses and other and configure the schedule for pulling status information (e.g. every 10 seconds, every hour, every week).

The above configurations are made with the help of annotation editors through which the configuration information about the task, be it functional or non-functional, is annotated to commonly used terms, using ADVENTURE ontologies managed by DPD, and thus the further searches for the best fitting partners are facilitated. Also the Broker is not limited to use only a fixed set of predefined configuration attributes but has the flexibility to add new as appropriate for the detail and nature of the task in design.

The broker may either:

- Design the whole process including all its activities specification and configuration and then invoke search for partners globally for the whole process
- Invoke search at each step (after each activity configuration).

When a search is triggered, the DPD component retrieves the services from partners' profiles that best fit the activity and its concrete attributes (see Sequence diagram for Find services for activity in the DPD component section) through a semantic search and provides a list of results. Further in the process of selecting a service from the recommended list, the user may browse the partners' profiles. When the concrete service is selected, the different service properties (such as BPMN-specific, ADVENTURE non-functional and operational, etc.) are automatically populated in the PME attributes pad for the selected activity.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>245</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The operational service properties are these properties that allow a task to be executed by the SPE (Smart Process Execution) component, and that usually are defined by means of calling a specific company gateway operation. This configuration implies that all the parameters of the gateway operations have to be imported from the gateway definition (inputs, outputs, securityTokens, etc.) that is specified during service data provisioning, to compose the activity configuration. The PME will interact with the DPD in order to retrieve this information and further configure it.

The following diagram shows the interaction between the ADVENTURE broker, the PME and the DPD in order to configure a task.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>246</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

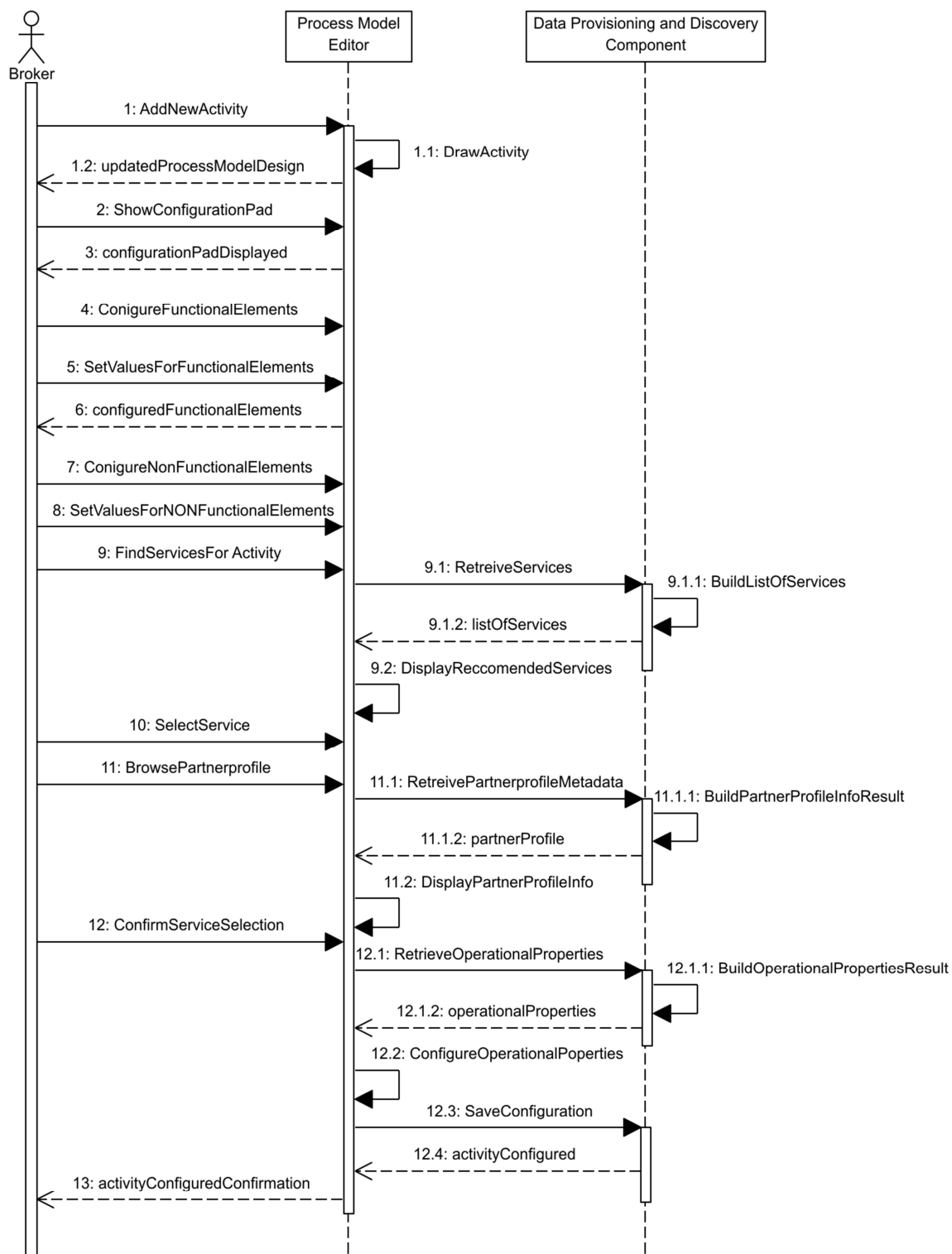


Figure 64 - New activity & activity configuration sequence diagram

The diagram starts with the Broker dragging an Activity from the editor’s Shapes Repository to the editor canvas. As a result of this action, the activity is displayed in the editor. Once this is done, the Broker can open the configuration pad in order to start configuring the different functional and non-functional attributes (standard and ADVENTURE-specific) of the activity. Using this interface the Broker designs the task specification and then, using the DPD-based interfaces, can request recommendations for feasible partners that can execute the assignment.

Assign Company Classification Description to Activity

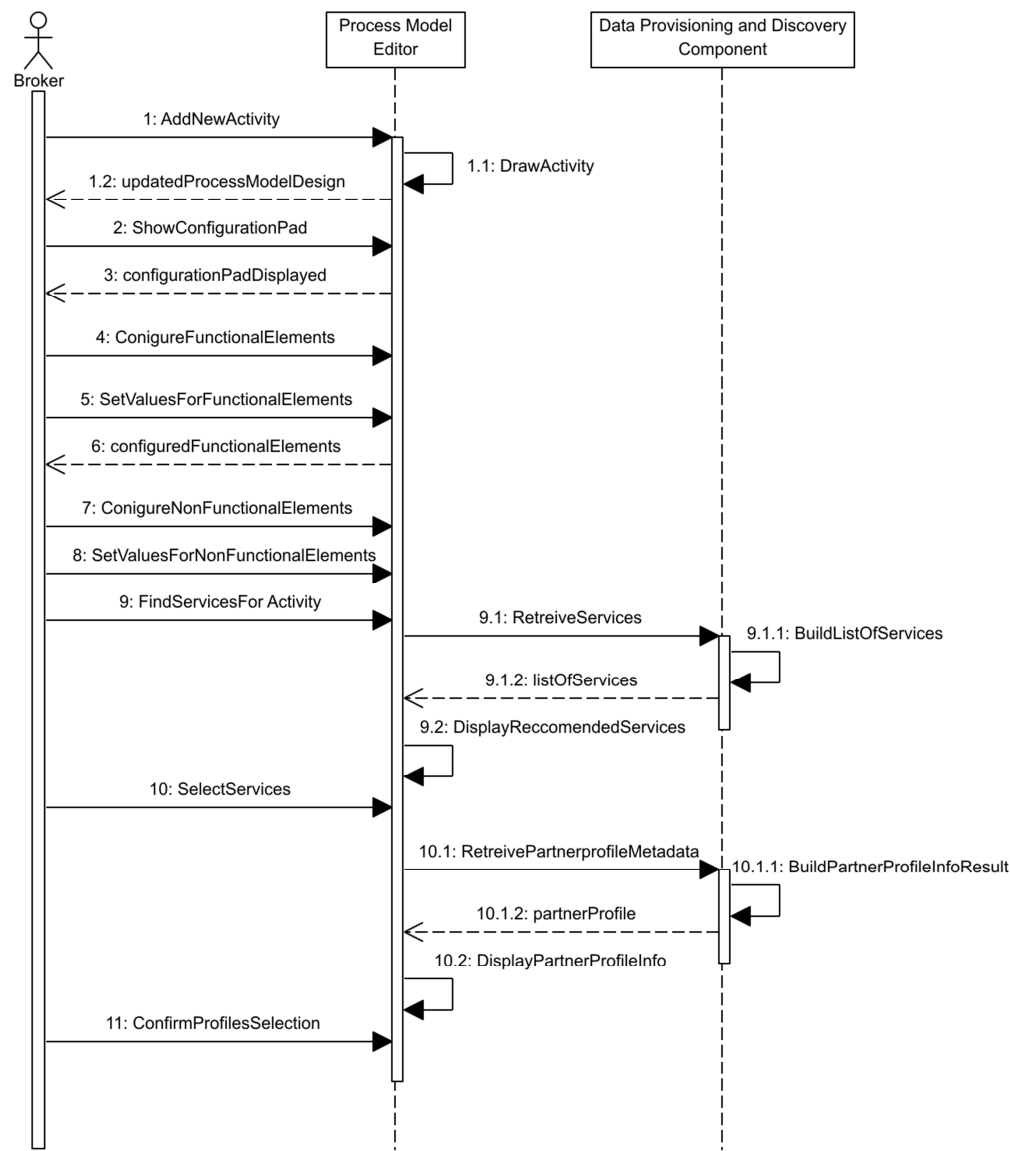


Figure 65 - Assign profile classification sequence diagram

This is a sub-scenario of the previous one, in which the user ends the configuration of the activity by just assigning a company profile classification (list of possible service/partners candidates to be assigned to the activity). This will afterwards allow the DPD component to recommend the most suitable company service to deal with the task. The options of multi-binding might be also supported through this scenario e.g. when a task should be performed by two partners at the same time, or when an unexpected event occurs and the partner assigned should be automatically changed during process run time adaptation.

Configure Activity

This diagram is an example of the configuration of an specific ADVENTURE customized (technical) activity. In order to facilitate the user with the process model building, specific ADVENTURE technical elements might be added to the BPMN editor. These elements will have specific purposes in the ADVENTURE framework (in this case the configuration of an ADVENTURE transformation service, allowing the transformation of an activity output document, into a different format).

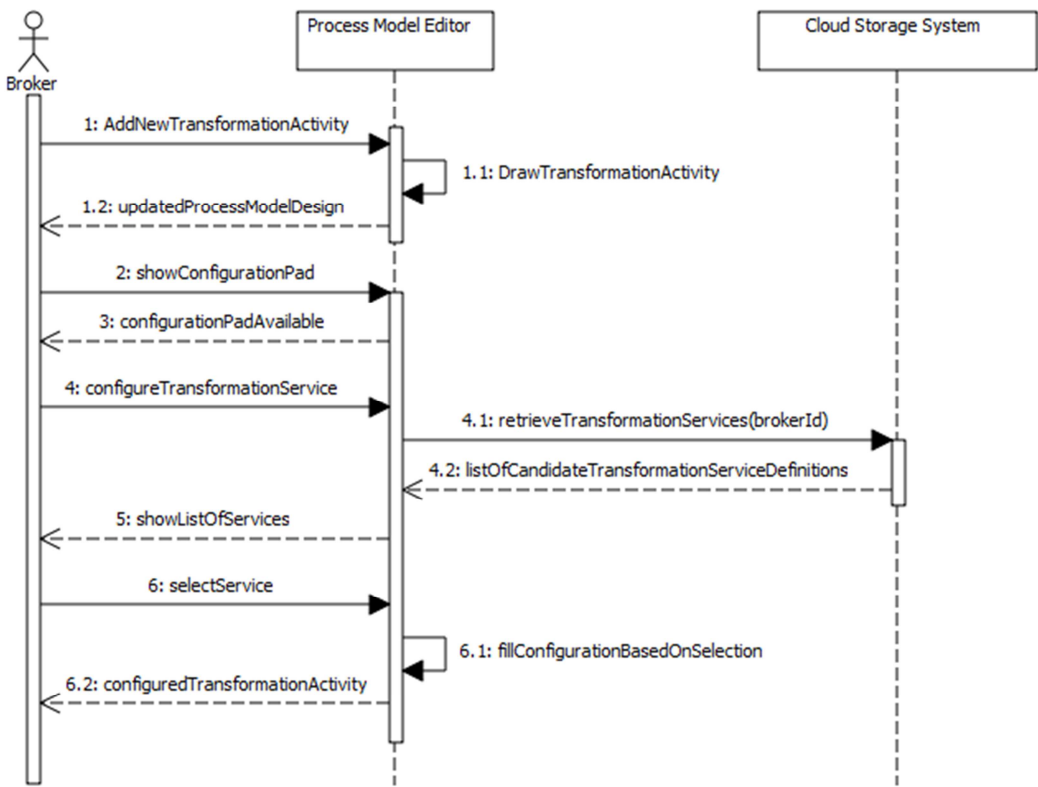


Figure 66 - New transformation activity sequence diagram

Once the Broker has dragged the transformation activity element (Transformation Service element) from the shape repository of the PME to the drawing canvas, the PME draws it. In this moment, the Broker can select this element, and display the Configuration pad, where the specific attributes for this element will be displayed. The user configures the transformation task and finally selects a specific transformation. Based on this selection, the PME completes the setup of the selected activity attributes (e.g. inputs, outputs, etc.).

Another option provisioned for configuring ADVENTURE customized activities (transformation in this case) is that they can be set up as part of the business activities configuration (as described in the previous scenario), in the operational attributes configuration part. This option spares the designer the need to use custom modeling elements in the design and keep it simpler (with the tradeoff for richer activity attributes configuration)

#### New Sub Process from Existing Process Model

The interaction between two companies might be more complex than just the execution of a specific (single) activity and embedding a sub-process in a process model will be needed. ADVENTURE will allow the Brokers, to select and configure an already existing process model saved by other companies. The Broker will be able to find and select these process models based on the non-functional information that describes each process model (e.g. information like product or sector classifications, etc.). The retrieval of information will be performed by the DPD client component. The selected process is embedded into the Virtual Factory, and can be further configured by the ADVENTURE broker.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>250</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

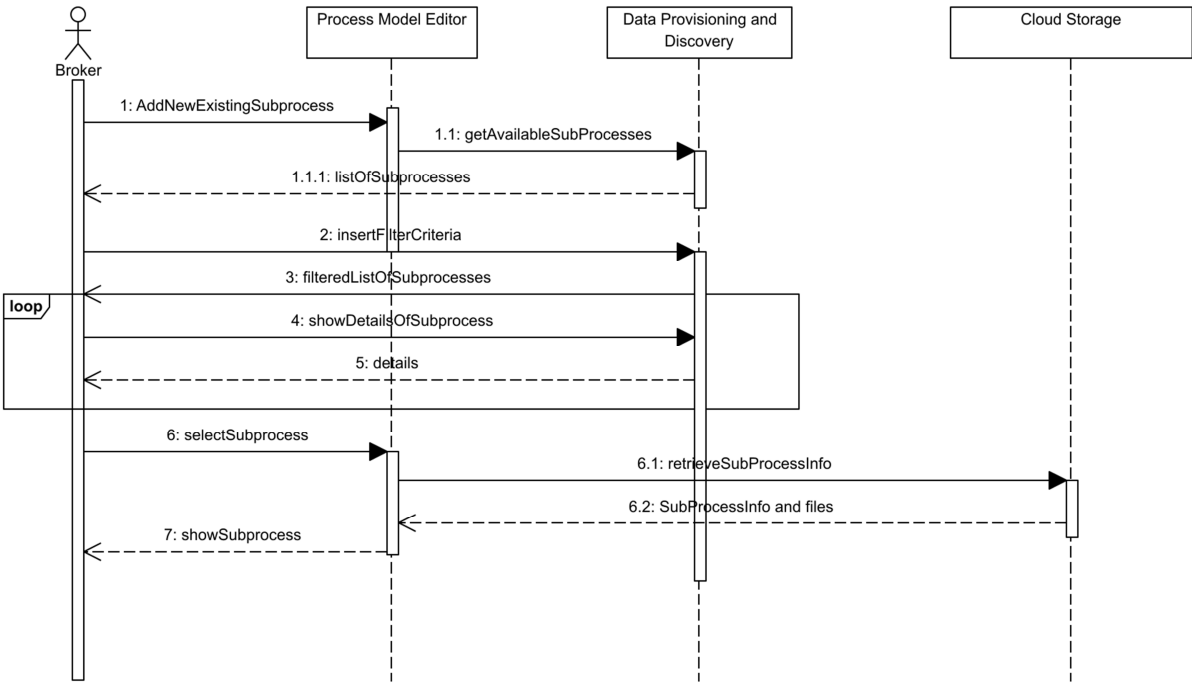


Figure 67 - New sub-process from existing process sequence diagram

3.7.4 Specification of Interfaces, Protocols and Formats

The connection of the Process Model Designer to the rest of components of ADVENTURE will be realized through the APIs exposed at the component. In the following sections these interfaces are explained.

3.7.4.1 API specification

The Process Model Designer component is aggregated into the Dashboard portal although it's will be available as a standalone application as well. The interaction protocol and API between these two components is defined by the Java Portlet specification (JSR 268) and is realized by the Dashboard interface Portlet on the Process Designer side. In addition, the Process Designer will expose a JavaScript API for the Process Editor so that it can be used by other components (like the Process Monitoring to show a read-only, enhanced with real-time data view of a process model instance). This minimal API is compliant and wraps the jBPM JavaScript API to provide direct access to the Process Editor's modeling canvas and the attributes section.

The original jBPM designer is a JavaScript library that operates on the page DOM and the SVG canvas to construct the required behavior and view. Alongside with the standard JavaScript API supported by the browsers, the Process Editor exposes the legacy jBPM designer JavaScript API that can be used to extend or manipulate the editor alongside with configurations in JSON format.

The following diagram shows an abstract overview of the required and provided (new) interface APIs by the Process Designer component.

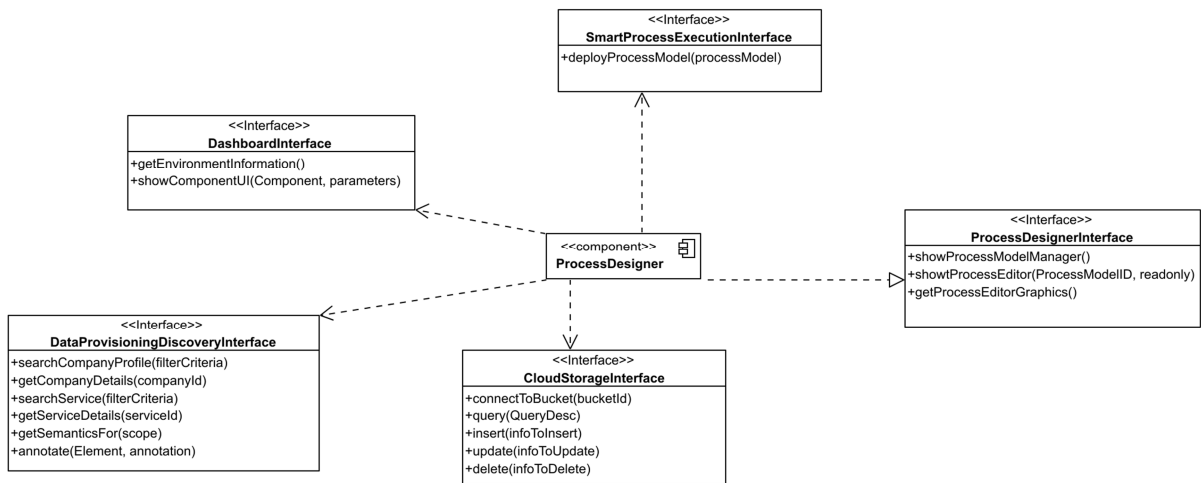


Figure 68 - Process Model Designer interfaces (required and provided)

showProcessModelManager

This method will show the PMM UI in the Dashboard. It will be called from the Dashboard as a response to the ADVENTURE user, clicking a menu option to access the PMM.

```
void showProcessModelManager()
```

showProcessEditor

This method will show the Process Model Editor, and load the process model specified as parameter.

```
void showProcessEditor(String ProcessModelID, boolean ReadOnly)
```



### Parameters

ProcessModelID: Identifier of the process model to be shown in the graphical interface.

ReadOnly: Parameter that describes if the process editor interface has to be shown in editing, or just in visualization mode.

### *getProcessEditorGraphics*

This method will allow external components (usually when the editor is in read-only mode) to obtain the Canvas reference of the currently opened editor, so that they can use it to add graphical elements on it, in order to represent additional information (such as status, etc.) or modify the visualization of the current represented elements (change color, etc.).

```
Graphics getProcessEditorGraphics()
```

### Return Value

Graphics: Root element of the graphics representation elements in the editor. It will contain the tree of SVG elements represented at the process model, so that the external elements could add new SVG elements or modify the existing ones.

## 3.7.4.2 Content Formats and Protocol Definitions

### BPMN 2.0

BPMN 2.0 is an OMG specification for designing and executing process models. It defines different conformance levels (Process Modeling Conformance, Process Execution Conformance, BPEL Process Execution Conformance, etc.).

The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>253</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The vision of BPMN 2.0 is to have one single specification for a new Business Process Model and Notation that defines the notation, metamodel and interchange format. The final version of the specification was released in January, 2011.

The BPMN 2.0 XML specification is based on a set of XML schemas, each one describing different parts of the complete XML specification:

- BPMN20.xsd: the base BPMN2.0 XML Schema definition
- Semantic.xsd: XMLSchema defining BPMN elements semantically (e.g. activities, sequence flows, events)
- DC.xsd: generic XMLSchema notation definition for basic diagram elements, such as diagram, plane, shapes and edges, etc.
- DI.xsd: generic XMLSchema notation definition for diagram presentation style and layout information.
- BPMNDI.xsd: BPMN XMLSchema extending DI.xsd and DC.xsd and complementing with additional definitions, which in total defines the standard XML representation for BPMN diagrams and modeling elements notation.

In the following diagram an example of a BPMN 2.0 diagram and a BPMN 2.0 XML representation are presented.

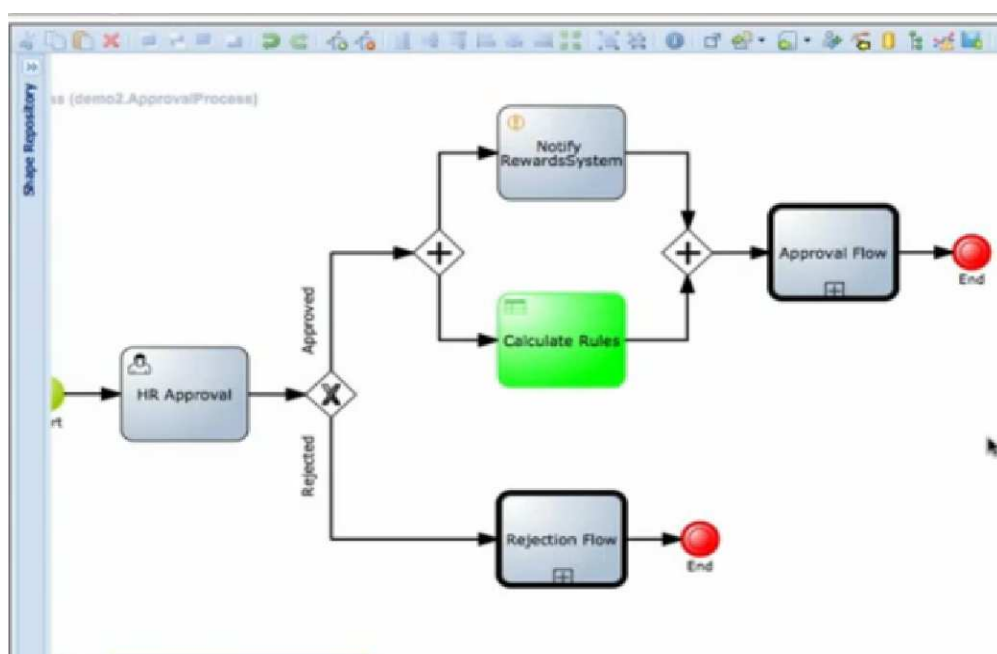


Figure 69 - BPMN 2.0 diagram

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>254</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

```

<?xmlversion="1.0"encoding="UTF-8"?>
<bpmn2:definitionsxmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xmlns="http://www.omg.org/bpmn20"xmlns:bpmn2="http://www.omg.org/spec/BPMN/2010
0524/MODEL"xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"xmlns:dc="http://www.
omg.org/spec/DD/20100524/DC"xmlns:di="http://www.omg.org/spec/DD/20100524/DI"xmlns:drools
="http://www.jboss.org/drools"id="_Hm9cMLuDEeG0coPTsB7EKQ"xsi:schemaLocation="http://ww
w.omg.org/spec/BPMN/20100524/MODEL
BPMN20.xsd"targetNamespace="http://www.omg.org/bpmn20">
<bpmn2:processid="adventureTests.test2"drools:packageName="adventureTests"name="test2"i
sExecutable="true">
<bpmn2:startEventid="_669ABB58-492D-42BF-8119-
62E5CF0A2094"drools:bgcolor="ffffff"name="">
<bpmn2:outgoing>_0A25C962-B829-4DA2-AF3A-1C4628895B56</bpmn2:outgoing>
</bpmn2:startEvent>
<bpmn2:scriptTaskid="_1BAA54A8-A4A3-4469-B57B-4A920F4DAEF2"name="test
element"scriptFormat="http://www.java.com/java">
<bpmn2:incoming>_0A25C962-B829-4DA2-AF3A-1C4628895B56</bpmn2:incoming>
<bpmn2:outgoing>_A05871A1-312C-4CBC-8532-B1E7C3F335A4</bpmn2:outgoing>
<bpmn2:script><![CDATA[return "orderNumber"]]></bpmn2:script>
</bpmn2:scriptTask>
<bpmn2:sequenceFlowid="_0A25C962-B829-4DA2-AF3A-
1C4628895B56"sourceRef="_669ABB58-492D-42BF-8119-
62E5CF0A2094"targetRef="_1BAA54A8-A4A3-4469-B57B-4A920F4DAEF2"/>
<bpmn2:endEventid="_F33CC52E-9893-460D-AF7F-
B249EBCC64EA"drools:bgcolor="ffffff"name="">
<bpmn2:incoming>_A05871A1-312C-4CBC-8532-B1E7C3F335A4</bpmn2:incoming>
</bpmn2:endEvent>
<bpmn2:sequenceFlowid="_A05871A1-312C-4CBC-8532-
B1E7C3F335A4"sourceRef="_1BAA54A8-A4A3-4469-B57B-
4A920F4DAEF2"targetRef="_F33CC52E-9893-460D-AF7F-B249EBCC64EA"/>
</bpmn2:process>
<bpmndi:BPMNDiagramid="_Hm9cMbuDEeG0coPTsB7EKQ">
<bpmndi:BPMNPlaneid="_Hm9cMrudeeG0coPTsB7EKQ"bpmnElement="adventureTests.test2">
<bpmndi:BPMNShapeid="_Hm9cM7uDEeG0coPTsB7EKQ"bpmnElement="_669ABB58-492D-
42BF-8119-62E5CF0A2094">
<dc:Boundsheight="30.0"width="30.0"x="293.0"y="208.0"/>
</bpmndi:BPMNShape>
<bpmndi:BPMNShapeid="_Hm9cNLUDEeG0coPTsB7EKQ"bpmnElement="_1BAA54A8-A4A3-
4469-B57B-4A920F4DAEF2">
<dc:Boundsheight="80.0"width="100.0"x="368.0"y="183.0"/>
</bpmndi:BPMNShape>
<bpmndi:BPMNEdgeid="_Hm9cNbuDEeG0coPTsB7EKQ"bpmnElement="_0A25C962-B829-4DA2-
AF3A-1C4628895B56">
<di:waypointxsi:type="dc:Point"x="308.0"y="223.0"/>
<di:waypointxsi:type="dc:Point"x="418.0"y="223.0"/>
</bpmndi:BPMNEdge>
<bpmndi:BPMNShapeid="_Hm9cNruDEeG0coPTsB7EKQ"bpmnElement="_F33CC52E-9893-
460D-AF7F-B249EBCC64EA">
<dc:Boundsheight="28.0"width="28.0"x="512.0"y="208.0"/>
</bpmndi:BPMNShape>
<bpmndi:BPMNEdgeid="_Hm9cN7uDEeG0coPTsB7EKQ"bpmnElement="_A05871A1-312C-
4CBC-8532-B1E7C3F335A4">
<di:waypointxsi:type="dc:Point"x="418.0"y="223.0"/>
<di:waypointxsi:type="dc:Point"x="526.0"y="222.0"/>
</bpmndi:BPMNEdge>

```

```

</bpmndi:BPMNPlane>
</bpmndi:BPMNDiagram>
</bpmn2:definitions>

```

The BPMN element set that will be supported by ADVENTURE Process Editor is based on the standard Business Process Diagram modeling elements offered by jBPM Web editor. The BPMN XSD allows extensibility by design, provided that the extension elements and attributes use their own namespace and that new elements are wrapped in an extensionElements wrapper (see the **bold** below).

```

<xsd:element name="baseElement" type="tBaseElement"/>
  <xsd:complexType name="tBaseElement" abstract="true">
    <xsd:sequence>
      <xsd:element ref="documentation" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="extensionElements" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" use="optional"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>

```

Using these rules the base XML schema of a serviceTask element for example can be extended (using adv as reference to a non-BPML namespace like <http://adventure.org/bpmn/ns>):

```

<serviceTask id="sometask" adv:anAttribute="somettr" >
  <extensionElements>
    <adv:action>something</adv:action>
  </extensionElements>
</serviceTask>

```

## SVG (Scalable Vector Graphics)

Scalable Vector Graphics (SVG) is a family of specifications of an XML-based file format for two-dimensional vector graphics. It is an open standard that has been under development by the World Wide Web Consortium (W3C) since 1999.

SVG images and their behaviors are defined in XML text files. This means that they can be searched, indexed, scripted, and, if need be, compressed. As XML files, SVG images can be created and edited with any text editor. All major modern Web browsers have at least some degree of support for SVG and can render markup

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>256</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

directly, including Mozilla Firefox, Internet Explorer 9, Google Chrome, Opera, and Safari.

```
<svgxmlns="http://www.w3.org/2000/svg"xmlns:oryx="http://oryx-editor.org" id="_F94F0F41-4923-4AF3-8670-EED001C96BAD"width="241.87109375"height="267.5"xmlns:xlink="http://www.w3.org/1999/xlink"xmlns:svg="http://www.w3.org/2000/svg"><defs/><gstroke="none"font-family="Verdana, sans-serif"font-size-adjust="none"font-style="normal"font-variant="normal"font-weight="normal"line-height="normal"font-size="12"><gclass="stencils"transform="translate(14.49609375, 4.5)"><gclass="me"/><gclass="children"><gid="_D3374C96-2E67-4F21-8AD6-990F51E8FE96"><gclass="stencils"transform="translate(105, 208)"><gclass="me"><gpointer-events="fill" id="_D3374C96-2E67-4F21-8AD6-990F51E8FE96" title="Start Event">

<defsid="_D3374C96-2E67-4F21-8AD6-990F51E8FE96__D3374C96-2E67-4F21-8AD6-990F51E8FE96_5">
    <radialGradientid="_D3374C96-2E67-4F21-8AD6-990F51E8FE96background"cx="10%"cy="10%"r="100%"fx="10%"fy="10%">
        <stopoffset="0%"stop-color="ffffff"stop-opacity="1" id="_D3374C96-2E67-4F21-8AD6-990F51E8FE96__D3374C96-2E67-4F21-8AD6-990F51E8FE96_6"/>
        <stopid="_D3374C96-2E67-4F21-8AD6-990F51E8FE96fill_el"offset="100%"stop-color="ffffff"stop-opacity="1"/>
    </radialGradient>
</def>

<circleid="_D3374C96-2E67-4F21-8AD6-990F51E8FE96bg_frame"cx="15"cy="15"r="15"stroke="#000000"fill="url(#_D3374C96-2E67-4F21-8AD6-990F51E8FE96background) white"stroke-width="1">
    <textfont-size="11" id="_D3374C96-2E67-4F21-8AD6-990F51E8FE96text_name"x="15"y="32"oryx:align="top center"stroke="black"stroke-width="0pt"letter-spacing="-0.01px"fill="#000000"text-anchor="middle"transform="rotate(0 15 32)"visibility="inherit"oryx:fontSize="11"/>
</g></g><gclass="children"style="overflow:hidden"><gclass="edge"></g><gclass="controls"><gclass="dockers"><gclass="magnets"transform="translate(105, 208)"><gpointer-events="all"display="none"transform="translate(7, 7)"><circlecx="8"cy="8"r="4"stroke="none"fill="red"fill-opacity="0.3"/></g></g></g></g><gclass="edge"><pathid="processFormP1" title="Edit Process Form"style="opacity:1;fill:#339966;stroke:#000000"d="M0.585,24.167h24.083v-7.833c0,0-2.333-3.917-7.083-5.167h-9.25c-4.417,1.333-7.833,5.75-7.833,5.75L0.585,24.167z"onclick="ORYX.Plugins.CanvasTitle.editProcessForm()"transform="translate(10, 20)"/><pathid="processFormP2" title="Edit Process Form"style="opacity:1;fill:none;stroke:#000000"d="M 6 20 L 6 24"onclick="ORYX.Plugins.CanvasTitle.editProcessForm()"transform="translate(10, 20)"/><pathid="processFormP3" title="Edit Process Form"style="opacity:1;fill:none;stroke:#000000"d="M 20 20 L 20 24"onclick="ORYX.Plugins.CanvasTitle.editProcessForm()"transform="translate(10, 20)"/><circleid="processFormP4" title="Edit Process Form"fill="#000000"stroke="#000000"cx="13.002"cy="5.916"r="5.417"onclick="ORYX.Plugins.CanvasTitle.editProcessForm()"transform="translate(10, 20)"/><pathid="processFormP5" title="Edit Process Form"style="opacity:1;fill:#FFCC99;stroke:#000000"d="M8.043,7.083c0,0,2.814-2.426,5.376-1.807s4.624-0.693,4.624-0.693c0.25,1.688,0.042,3.75-1.458,5.584c0,0,1.083,0.75,1.083,1.5s0.125,1.875-1.3s-5.5,1.25-6.75,0s8.668,12.834,8.668,12s0.583-1.25,1.25-1.917C8.835,9.5,7.419,7.708,8.043,7.083z"onclick="ORYX.Plugins.CanvasTitle.editProcessForm()")
```

```
transform="translate(10, 20)"/><textid="_39DEA1E5-6B2B-4017-94DA-
D2F3A3E71530"style="stroke-width:1;fill:rgb(177,194,214);font-family:arial;font-weight:bold"font-
size="12"onclick="ORYX.Plugins.CanvasTitle.openTextualAnalysis()"onmouseover="ORYX.Plugins
.CanvasTitle.addToolTip('_39DEA1E5-6B2B-4017-94DA-D2F3A3E71530')transform="translate(40,
43)">test2 (adventureTests.test2)</text></g></svg>
```

### 3.7.5 Summary

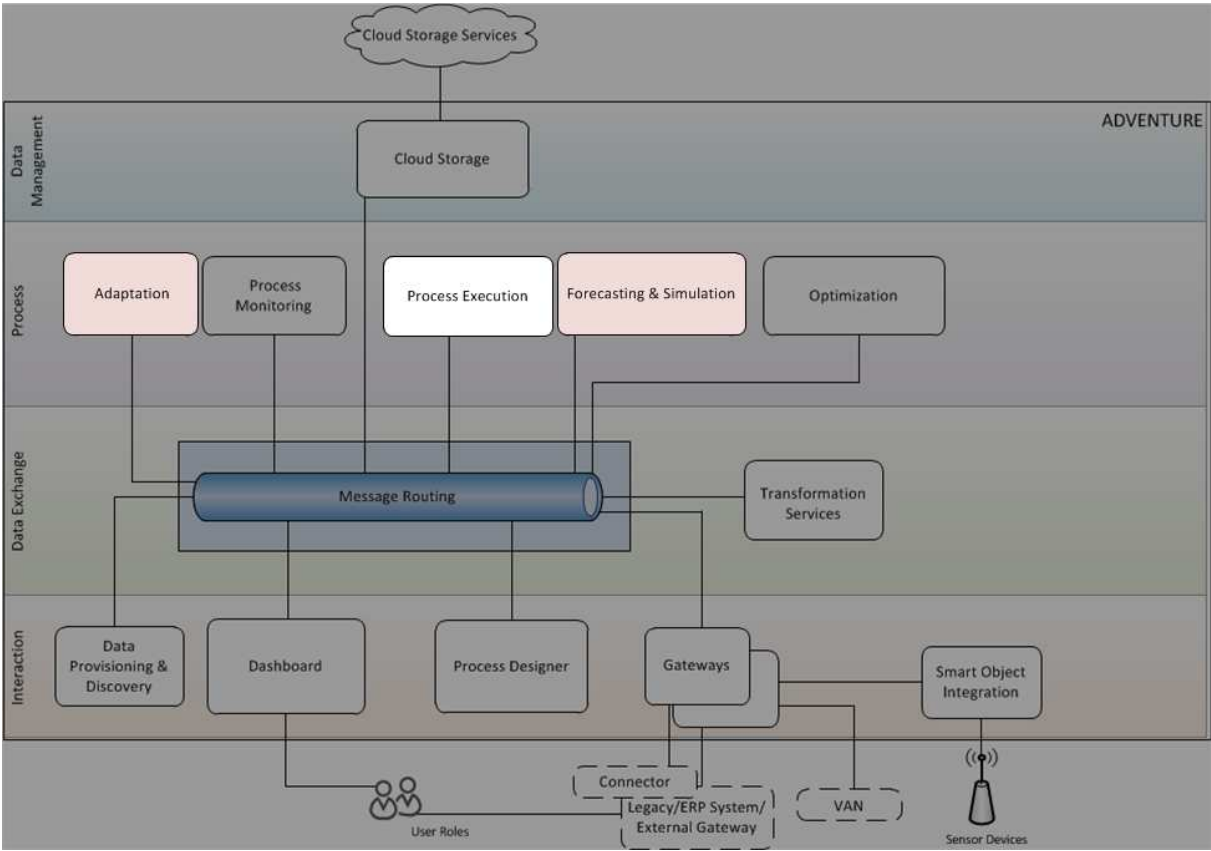
The Process Designer component will use jBPM Designer as base software platform and extend it to implement the full set of functionalities and requirements identified in the previous deliverables. The extensions provided to the base platform include extensions to the standard diagram elements (semantics and notation) that complement them with manufacturing specific task specifications and collaboration-specific modeling elements. The basis is also extended with a set of functionalities that add value to the design process, delivered by other ADVENTURE modules, such as Optimization, Simulation, Data Provisioning and Discovery.

The ADVENTURE users shall be able to use the online, Web-based ADVENTURE design platform to create new collaboration processes, tune them for best performance, and in addition exercise full governance on their process models and lifecycle.

The process designer module will be implemented in coherence with the other ADVENTURE services and according to the general architecture guidelines.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>258</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

# Smart Process Execution, Adaption, Forecasting & Simulation



## 3.8 Smart Process Execution

### Abstract

- The SPE Component coordinates the routing of all messages related to the business logic embodied by a virtual factory.
- The SPE Component deals with the enactment of the control flow (waiting for messages, sending messages, decisions, parallel execution, script tasks) realizing certain aspects of the business logic.
- The SPE Component provides a flexible and extensible way of monitoring and manipulating the enactment of the business logic.

This section also contains the specification of the Adaptation (3.8.3.6,3.8.4.4) and Forecasting & Simulation (3.8.3.7, 3.8.4.5) components. These two components are not presented in their own top-level sections, because they realize algorithms that fully depend on the capabilities of Smart Process Execution. The Adaption applies changes (due to optimization and repair actions) to running processes instances. The Forecasting & Simulation component, instead of calling activities, uses information from the Data Provisioning Component to invoke business logic assigned to activities in order to collect information about these activities. This information is collected, refined, stored and presented.

### 3.8.1 Major Design Decisions

In order to allow for simple and continuous improvement and innovation of the virtual factory environment, five design decisions have been taken:

- Reduced core: deal only with execution, not with data source/user/rule management.
- Service-oriented, supervisor based<sup>14</sup> design.
- Process description language agnostic execution.

---

<sup>14</sup> A term borrowed from the operating system domain, namely microkernel design, where hierarchical protection domains (rings) exist. The innermost ring is envisioned as the Smart Process Engine (SPE), while different functionality with reduced privileges exists as separate services that communicate with the SPE. For more detailed explanation the concepts please see: Walter Binder, Design and Implementation of the J-SEAL2 Mobile Agent Kernel, saint, p. 35, 2001 Symposium on Applications and the Internet (SAINT'01), 2001.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>260</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



- Explicit instance spawning.
- Attached Script Tasks.

All these design decision have been taken independent of the selection of a particular process engine, with the intention of allow for the replacement of core components, e.g. the process engine itself, to provide a sustainable framework for companies / researchers to adapt the system to their own needs or preferences. The Figure below (Replaceable Process Engine at the Core) shows a component diagram that has a process engine at its center, which is encased in a wrapper component that provides stable management interfaces, as well as disperses events (including signals that trigger correlation), calls activities and delegates script task handling.

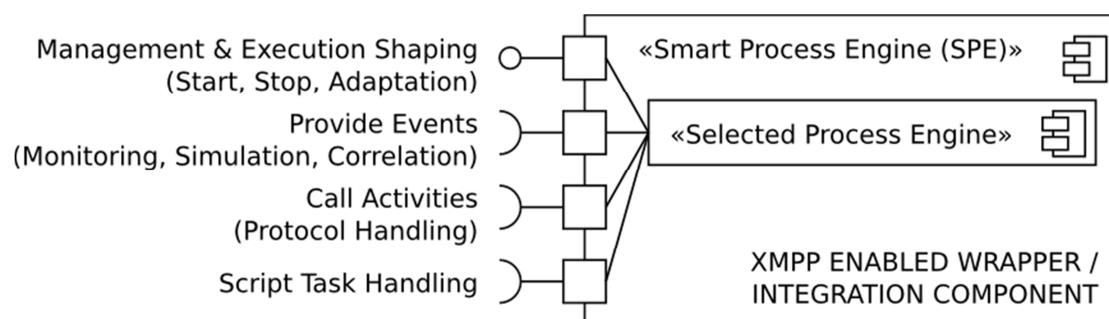


Figure 70 - Replaceable Process Engine at the Core

The following subsections will explain these decisions as well as the rationale behind them.

### 3.8.1.1 Reduced Core

The engine itself should provide only three kinds of functionalities:

- Enact control flow.
- Distribute events about control flow.
- Provide stable interfaces to allow management (e.g. start/stop instances) and execution shaping (event based ad-hoc instance changes) for external services.

### 3.8.1.2 Service-Oriented, supervisor based design

Instead of providing a monolithic process engine with built-in/linked extensions, ADVENTURE aims to separate the supervising core functionality (process execution) from the following aspects:

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>261</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- **Monitoring:** Certain aspects of a process execution may be interesting only for certain components. Example: while the ADVENTURE Monitoring component certainly consumes all events that accrue during the enactment of business logic, the ADVENTURE Simulation component will only consume a subset of the events. Based on this subset the ADVENTURE Simulation component will then change several process execution parameters.
- **Process Activity interaction protocols:** Process activities are placeholders for a communication with services through SOAP, Rest, ODBC, etc. They also are containers for a set of properties (e.g. input, output, non-functional properties) to describe the communication with these services. As ADVENTURE uses a common messaging protocol for all components (XMPP, see section 3.4), this allows for engine independent development in this area.
- **Management and Execution Shaping:** A means to instantiate, manage (e.g. start/stop instances) and shape the execution (event based ad-hoc instance changes). The following components will engage in management tasks:
  - The ADVENTURE Dashboard provides means to instantiate processes from the Process Model Management (PMM) overview (explained in the Process Designer section), as well as means to start/stop process instances from an SPE runtime overview (as described in D2.2).
  - The ADVENTURE Process Editor triggers the ADVENTURE Adaptation Component, which in turn will stop/start certain instances and roll the changes into these instances.
  - The ADVENTURE Process Editor requests a simulation from the ADVENTURE Simulation Component, which in turn will instantiate a process instance for simulation.
  - The ADVENTURE Process Editor requests a forecast from the ADVENTURE Simulation Component, which in turn will instantiate a process instance for simulation, and then guide this instance to execute only a certain activity.

All these components use an abstract interface to manage and shape the execution of instances. This has two aspects:

- An API to allow for interactions.
- A protocol to implement this API (XMPP in the case of ADVENTURE).

While the API itself is to be fine-grained and stable, and part of the core, the protocol for interacting with the API has to be independent of the core.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>262</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- Correlation describes a means to identify messages based on their structure and content, and assign them to running process instances that are currently waiting for them. A correlation (sub-) component is defined in order to comprise of the following parts:
  - Waiting for messages and identifying relevant messages.
  - Assigning messages to instances.

The implementation of a correlation strongly depends on the protocol of the messages to wait for, in the case of ADVENTURE - XMPP (see section 3.4). The XMPP implementation has to be independent of the core.

- Script tasks may require a special purpose language to modify data elements (variables) that exist in the process context and e.g. hold payload of messages received by a certain process instance. For ADVENTURE it will use JavaScript (see section 3.4).

Activities that are waiting for certain messages (i.e. events in BPMN) may of course have process specific logic to decide if they want to either continue existing instances, or spawn new instances. This logic is in script tasks that work in conjunction with the ‘waiting’ activities. Thus there is no need for a separate rule engine, which is present in some Process Engines to handle these cases.

### 3.8.1.3 Process Description Language agnostic execution

The SPE component will include a means to transform the processes saved by the ADVENTURE Process Designer to the format required by the selected process engine. This has two main advantages:

- It is possible to add new convenience, or manufacturing process specific elements in the ADVENTURE Process Designer, without the need to modify the process engine itself. Convenience design elements might for example include ADVENTURE specific events and patterns that could be modeled through more complex structures, but are provided as single elements because of their common use (for example continuous communication as described in D3.2).
- Different components, like the ADVENTURE Optimization, or ADVENTURE Simulation component, can focus on core functionality instead of having to deal with specific file formats.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>263</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The overall advantage is that by decoupling the components the ADVENTURE project can progress faster.

#### 3.8.1.4 Explicit Instance Spawning

Instead of following the scheme pioneered by BPEL, that special activities (or events in BPMN) are spawning new instances of a process, ADVENTURE will invert control, and have running instances spawn new instances. In order to illustrate this decision an example is given as follows:

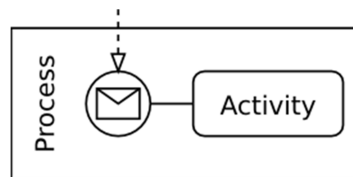


Figure 71 - BPMN New Instance Spawning

A BPMN “message start” event typically is part of a particular process model (see Figure 71). A Process Engine has to analyze all process models, extract all special activities (events) separately, listen for all of them on a single interface, and then spawn instances upon receiving a message (event). This model has several drawbacks:

- Single point of failure.
- In order to deactivate certain events, annotations to the process model have to be made, which means that an ADVENTURE Broker has to understand the process model and modify it (which assumes that the ADVENTURE Broker understands all implications that the deactivation of an event can have to a process).
- A central process repository and shared process model for all instances is assumed, thus limiting the scalability of this approach.

Instead for ADVENTURE it was decided to not include the logic to spawn new instances into the process models that are to be spawned as instances (target), but instead solely rely on explicit instance spawning on behalf of either ADVENTURE Brokers or source process instances. Instead of analyzing the instances, running instances are to deal with the business of spawning other instances through a special activity (see Figure 72).

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>264</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

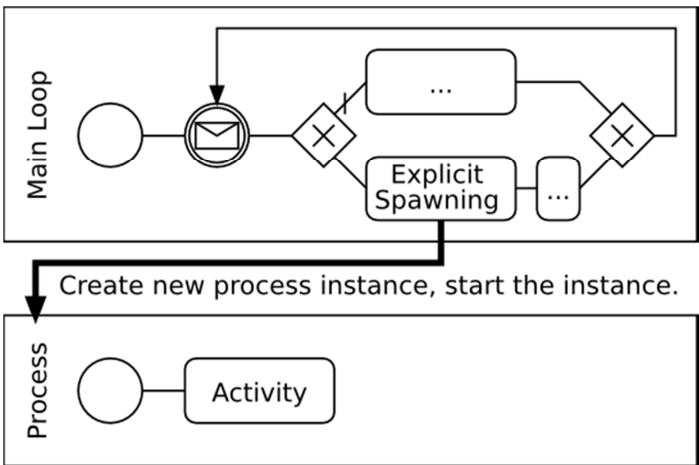


Figure 72 - ADVENTURE Explicit Instance Spawning

In ADVENTURE this is called **Main Loops** – Processes that only have 1 instance. Main Loops cannot be nested, they are either on or off, and can be switched on or off, either by the ADVENTURE Broker, or by some yet to be defined rules: e.g. in Monitoring component there may be is a rule, that when multiple instances of a certain process repeatedly violate some KPI's (Key Performance Indicators) (e.g. producing too much waste), this part of the factory is shut down to prevent further damage. Shutting down in this case would be as easy as stopping the instance that is handling the events. Drawbacks:

- Different design paradigm than e.g. BPMN or BPEL (automatic/human translation needed).
- Higher number of process models than for the other approach.

In contrast, the advantages of this approach are strong, both for ADVENTURE brokers and technically:

- Very clear human understandable means of stopping/pausing event handling: stop instance. This can be done without any semantic knowledge of the process models (that in BPMN can contain a multitude of entry points). Event handling can be resumed by continuing execution.
- As every “Main Loop” instance handles instantiation (in the case of ADVENTURE all instances listen to XMPP server), there is no need for pre-analysis of process models, and no central point of control, which will improve scalability and performance considerably.

From the point of view of the ADVENTURE Broker that does the process design (UI point of view) there is two ways to implement this:

- Explicit guidelines how to design the factory, with the danger of introducing a slightly higher learning curve, but with the advantage achieving easier to manage factories.
- Automatic transformation of the BPMN / BPEL models to extract the “Main Loops”. For the ADVENTURE broker, there will be no difference, although a sane model has to be devised for showing which “Main Loops” belong to which processes, in order to make the transformation transparent for the ADVENTURE Broker.

### 3.8.2 Technology Comparison and Selection

This subsection will compare existing technologies for the Adaptive Execution Engine. Those technologies will be used as a base for the development phase.

#### 3.8.2.1 Selection Criteria

For generic parameters, see the functional specification. For the execution engine the following parameters are important:

- **Performance:** Some factories might have special ‘real time’ requirements, thus ensuring that the process engine can deliver a certain amount of performance is crucial.
- **Scalability:** As virtual factories might become arbitrarily big or complex, which can easily result in thousands of concurrently running processes, scalability is important. There are several ways to handle this problem:
  - Monolithic process engines might scale up by scaling up the machine they run on.
  - Scalability over several nodes might be provided by the framework a process engine is implemented in (message based communication inside engine, instead of shared memory).
  - The architecture of the process engine itself might allow for multiple collaborating nodes (which might be deployed in a cloud solution).
- **Sub processes:** A process instance can spawn new sub process instances. Several patterns how sub process may be spawned exist (see <http://www.workflowpatterns.com/patterns/control/> -multiple instance patterns).

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>266</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- **Engine Management:** The execution engine should offer the possibility to manipulate the execution of processes and instances over web-service interfaces during runtime. This interface should offer possibilities such as the creation, starting and stopping of instances or the modification of data elements.
- **Script tasks:** Script tasks are automated tasks that are run by the execution engine. Script tasks are specified in a programming language and are allowed to read/modify the process context:
  - Process data elements (variables) that contain e.g. results from service calls.
  - Endpoints (services/gateway identifiers). This feature depends on the process language, e.g. for BPEL this is not foreseen. Very useful for dynamic service selection.
- **Event listener:** It is important that the execution engine enables other components to subscribe to event streams that are configurable, depending on the information needs of the other component. These event streams should be communicated using web services (e.g. RESTful).
- **Supported Events:** There are different granularities in terms of what kind of events can be subscribed and what kind of event information is broadcasted. The events that the execution engine conveys should be as fine-grained as possible.
- **Web interface:** The administration of the engine should be possible over a web interface.
- **Platform-independence:** The engine should be platform-independent or at least run on all major operating systems.
- **Familiar Development Environment:** It is important in which environment the engine has been developed - the engine will have to be modified during the course of ADVENTURE, therefore it should be developed in a language that many consortium members are familiar with.
- **Open standard compliance:** The execution engine should be compliant with open standards such as SOAP, WSDL, BPMN.
- **Integration capabilities:** The architecture of the execution engine should base on loose coupling, in order to integrate it into the other components of ADVENTURE.
- **Extensibility:** The execution engine should be easily extendible (e.g. to implement new protocols or script languages).

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>267</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- **Lifecycle management:** For the best integration into the ADVENTURE framework, it should be able to programmatically start, stop, modify or restart process instances.
- **Lifecycle management interface:** In order to provide architecture independence, the life cycle management should be possible over web service, e.g. using the REST paradigm.
- **SOAP activities:** SOAP is a protocol that is widely, industry accepted and therefore the execution engine should support activities that invoke web service using the SOAP protocol.
- **Receive SOAP external events:** In order to run processes that can be integrated into other systems; the engine should support the invocation by using SOAP. The functionality to receive SOAP events can be reused in the ADVENTURE context to manage events that follow the procedural SOAP invocation style, which is prevalent in existing ERP systems.
- **REST activities:** the REST paradigm is gaining increasing acceptance. Thus, activities that invoke web services that base on the REST paradigm should be supported by the engine.
- **Receive REST external events:** In order to run processes that can be integrated into a variety of other systems; the engine should also support the invocation through REST services. This can be reused in the ADVENTURE context to manage events that follow the data-centric REST invocation style.
- **Pluggable external event protocol:** In order to communicate with the other components of ADVENTURE, the execution engine has to support external event protocols.
- **BPMN 2.0 compatible:** Many execution engines can run processes that have been modeled according to the BPMN 2.0 standard. Thus, BPMN compatibility is an important feature of an execution engine.**BPEL 2.0 compatible:** As BPEL is still the most popular business process execution language, it is important that the execution engine is also able to run processes modeled in BPEL.
- **Build infrastructure:** Build automation can save a lot of time and increase the product quality. Therefore, execution engines that have existing build automation are preferable.
- **Interface for activity/engine management calls:** It is important that the execution engine offers APIs for easy manipulation. Ideally, these APIs are not offered in programming languages like Java, but can be queried as web services.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>268</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



- **Architecture independence:** The individual components of the execution engine should be defined as fine-grained and independent as possible, in order to ease the integration into the ADVENTURE framework and to enable the exchange of individual components of the engine.
- **Function-Point Analysis (Code complexity):** As the code of the execution engine will have to be adapted to meet the requirements of ADVENTURE, it is important that the existing code is as understandable as possible.
- **Tests Coverage:** The number of test cases and assertions can be an indication of a project's code quality.
- **Code quality:** The code quality should be as high as possible, as it eases the understanding and maintenance of the code.
- **Consortium affinity/skills:** Another important factor is how familiar the consortium members already are with certain execution engines. If members are already familiar with an engine, the familiarization period can be shortened substantially.

### 3.8.2.2 Possible Technologies

In order not to reinvent the wheel, it seems sensible to base ADVENTURE on one of the many existing execution engines. There are several different open source execution engines available, which usually either run processes that have been designed using BPEL (Business Process Execution Language) or that include editors that allow the creation of executable BPMN (Business Process Model and Notation) diagrams. The execution engines that have been selected for this evaluation are: Apache ODE, Activiti, CPEE, JBoss jBPM and JBoss RiftSaw.

**Apache ODE** - Apache ODE (Apache Orchestration Director Engine) is a business process execution engine developed by the Apache Software Foundation. The project is still active, the latest version is 1.3.5. Apache ODE is written in Java and comes under the Apache License 2.0. It comes with an optional process editor that allows the graphical creation of service compositions that base on the WS-BPEL 2.0 standard.

**Activiti** - developed by Alfresco and the Activity community is an open source business process execution engine. It is written in Java and still maintained. It comes under the Apache License 2.0 and is part of a suite that includes, among others, a process editor. There exists also a process editor that allows the creation of

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>269</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

executable BPMN process diagrams. However, in contrast to Apache ODE, Activiti's main focus is not the creation of web service orchestrations, but instead the execution of processes that are mainly-based on human-tasks (e.g. over web forms) and on java-services. Therefore, the editor, in its latest stable version, supports no web service task. There is however an experimental version that includes such a web services task, but it is advertised as 'not yet stable'. This web service task in addition does not support a wizard-based import of a wsdl-file and a subsequent selection of methods and variables like Apache ODE. Instead, the interfaces and data types of the service have to be described by the user in an XML file.

**CPEE** – The cloud process execution engine is a modular, service oriented workflow execution engine targeted at researchers, developers and system administrators. It has been developed under the LGPL 2.1 license and is continuously updated by the University of Vienna. Its goal is to provide a simple and lightweight, but also more powerful alternative to existing execution engines. Modularity for the CPEE is defined as supporting (1) execution languages (i.e. BPEL, YAWL, BPMN), (2) interaction protocols (SOAP, REST), and (3) execution shaping (runtime ad-hoc changes) by adding simple REST web-services. In contrast to other engines, there is no internal API, dependency on frameworks or programming languages. CPEE can be embedded into any website or application (running on a server, in a cluster or in the cloud) by solely relying on the HTTP protocol. Additional properties:

- The engine is exposed as a REST service.
- Its core has ~ 700 lines of code.
- Startup in less than 10 ms.
- After startup it consumes ~ 17 MiB memory.
- It consumes often less than 2 MiB memory per running instance.
- It heavily optimized for multi-core architectures.
- It beats a wide selection of industry leading Process Management Systems in terms of pattern support.
- It supports several virtual machines (Ruby 1.8.7, Ruby 1.9.x, Java [JRuby])

**JBoss jBPM 5.3.0** – jBPM, developed by the JBoss Community and published under the Apache License 2.0, follows since Version 5 a similar approach as Activiti (focus more on human tasks than on web service orchestration and tight coupling with Java). Similar to Activiti, it bases on executable BPMN diagrams that can be created using a proprietary editor. As one main criterion for selecting the execution engine is

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>270</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

an easy way to run service orchestrations that invoke web-services, the fact that jBPM does not feature out-of-the box web service tasks is a knock-out-criterion. Thus, jBPM will not be evaluated in more detail.

**JBoss Riftsaw 2.3.0** – Riftsaw is an execution engine by the JBoss Community that runs BPEL processes. It is being distributed under the LGPL 2.1 license and bases on Apache ODE. Therefore it can be assumed that it shares most of the features of Apache ODE, which is why only Apache ODE will be analyzed in more detail for this evaluation.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>271</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The following table shows information about potential technologies, as well as integrates ratings regarding the selection criteria above. The table for the SPE is slightly different from the tables in other sections, in that details are included that lead to the rating in the table, or added to the decision. Due to the high number of criteria and the often extensive scope of facts involved in a single criterion, it was decided to include the facts in the actual table instead of discussing them separately after the table. This improves readability and decision traceability considerably. It is also important to note that when comparing different technologies, rating criteria such as flexibility simply as (---+/-+++) proved to be difficult: thus, for the sake of completeness, a description of the criteria properties regarding a certain product has been included in the table.

*Table 13 – Technology selection criterion and comparison of technologies for Smart Process Execution Engine*

Parameter	Importance (---+/-+++)	Apache ODE 1.35 <a href="http://ode.apache.org">http://ode.apache.org</a>	Activiti 5.8 <a href="http://activiti.org">http://activiti.org</a>	CPEE (Cloud Process Execution Engine) <a href="http://www.cpee.org">http://www.cpee.org</a>
Vendor	- - -	Apache Software Foundation	Alfresco and the Activiti community	University of Vienna
License	+	Apache License 2.0	Apache License 2.0	LGPL 2.1
Licenses of libraries	+	Several open-source libraries (liberally-licensed, i.e. non-gpl)	Many open-source libraries (using Apache 2.0, BSD, GPL 2.0, LGPL 2.1, LGPL 3.0, CDDL 1.0, Creative Commons and other licenses)	Riddl - LGPL 2.1 XML-Smart – LGPL 2.1 Nokogiri - MIT License Rack – MIT License
Price model	+++	Free	Free	Free

Maturity	+++	+++	+++	+++
Stability	+++	+++	+++	+++
Maintenance	++	+++	+++	+
Documentation	+	+++	++	-
Community	+/-	+++	++	-
Actuality	++	+/-	++	+++
Performance Test 1 (service that calls another service which returns a 500 k string 1,000 times and assigns it to a variable)	+++	Average (3 calls): 2.217.238 ms/36.95 minutes	Test-process cannot be designed without high efforts using the latest stable version of Activiti Designer	Average (3 calls): 997,094 ms/16.6 minutes
Performance Test 2 (instantiating a process, taking an input, returning the result, 100 times)	+++	Average (3 calls): 28.55 secs	Test-process cannot be designed without high efforts using the latest stable version of Activiti Designer	Average (3 calls): 108.44132781029 secs  Comment:  The difference here is explainable by how the test has been carried out. While for ODE the instance was ready

				waiting to be instantiated, for the CPEE, the process model has to be uploaded through the REST interface, the model has to be transformed to an executable form, then the process has to be started.
Memory utilization	+++	Empty: 251 MiB; Test 1: 270 MiB	N/A	Empty: 20 MiB; Test 1: 40 MiB; Test 1 repeated: 10 MiB
Scalability	+++	Container dependent (Tomcat, JBoss)	Container dependent (Tomcat, JBoss)	Load-Balancing over independent nodes.
Gateways	+++	Based on BPEL: Static Parallel gateway (static number of parallel branches) Exclusive gateway Inclusive gateway	Based on BPMN 2.0: Static Parallel gateway Dynamic Parallel gateway (simulatable with Parallel Event Based Gateway). Exclusive gateway Inclusive gateway	Based on workflowpatterns.org: Static Parallel gateway Dynamic Parallel gateway (mixed with static branches, loops and or event conditions, not through sub processes) Exclusive gateway Inclusive gateway
Sub processes (how?)	- - -	Not directly	Embedded sub process elements Sub process events (sub processes that can be triggered by an event, to be added at	Process element to instantiate new process instances from process models, process context can be transferred (process data elements/variables).

			process or sub process level) Transaction sub process Call activity (sub process)	
Engine Management	+++	Create new instances (soap/rest/no): no Load Process models: no Start new instance: no Stop instance: SOAP Change model for stopped instances: no Change thread of control/position pointers: no Change endpoints (independently of model): no Change endpoints during runtime through process logic (independently of model): no Modify data elements: only at runtime Restart instance: no Pluggable language descriptions for conditions & script tasks: no	Create new instances: no Load process models: only by JAVA-API Start new instance: Rest Stop instance: no Change model for stopped instances: no Change thread of control/position pointers: no Change endpoints (independently of model): no Change endpoints during runtime through process logic (independently of model): no Modify data elements: only at runtime through JAVA-API Restart instance: no Pluggable language descriptions for conditions & script tasks: no	Create new instances: rest Load process models: rest Start new instance: rest Stop instance: rest Change model for stopped instances: rest Change thread of control/position pointers: rest Change endpoints for stopped instances (independently of model): rest Change endpoints during runtime through process logic (independently of model): yes Modify data elements: at runtime through script tasks, while stopped through rest Restart instance: yes Pluggable language descriptions for conditions & script tasks: yes (Ruby, JS, Python)

Script tasks (which languages)	++	BPELscript	Javascript, Groovy	Pluggable: Javascript, Ruby, Python, ...
Events visible	+++	Pluggable: Log, SOAP, Java API	Log/Java API (Event Bus): all events can be subscribed.  REST: only querying of historic execution data.	Pluggable: Rest, Websocket
Supported Events	+++	Activity Calling: Yes (SOAP, Java API) Activity Waiting: No Activity Manipulating: Yes (II) Activity Failed: Yes (II) Activity Update: No Activity Done: Yes (II) Activity Before: Yes (II) Activity After: Yes (II) Position change: No Description change: No Description error: No State change: Yes (II) Status change: No Data elements change: Yes (II)	See above	Activity Calling: rest Activity Waiting: rest Activity Manipulating: rest Activity Failed: rest Activity Update: rest Activity Done: rest Activity Before: rest Activity After: rest Position change: rest Description change: rest Description error: rest State change: rest Status change: rest Data elements change: rest Endpoints change: rest



		Endpoints change: No Protocol change: No Gateway change: No Gateway error: No Gateway debug: No Gateway info: No Condition evaluated: No		Protocol change: rest Gateway change: rest Gateway error: rest Gateway debug: rest Gateway info: rest Condition evaluated: rest
Web interface	---	Yes	Yes	Yes
Platform-independence	---	Yes: all that support Java	Yes: all that support Java	Yes: Windows, Linux, OSX, iOS
Familiar Development environment	---	Eclipse	Eclipse	Eclipse
Open standards compliance	++	Tomcat/JBoss, SOAP, BPEL, WSDL, ...	Tomcat/JBoss, Java, BPMN 2.0, ...	SOAP, REST, BPEL, BPMN, WebSockets, ...
Integration capabilities	+++	Tightly coupled (Management API), in java code	Tightly coupled (Engine API), in java code	Loosely coupled, through REST interface and Websockets -> remote integration into HTML websites, as well as server side.
Extensibility	+++	Well defined apis, protocols for: new protocols (e.g. xmpp): no	Well defined apis, protocols for: new protocols (e.g. xmpp): no	Well defined apis, protocols for: new protocols (e.g. xmpp): yes

		script tasks: yes script languages: no management (start/stop instances: partially execution shaping (delay/modify instances with external logic based on events from instance execution): no -> tightly coupled	script tasks: yes script languages: partially (all JSR-223-compliant languages) management (start/stop instances: partially execution shaping (delay/modify instances with external logic based on events from instance execution): no -> tightly coupled	script tasks: yes script languages: yes management (start/stop instances: yes execution shaping (delay/modify instances with external logic based on events from instance execution): yes -> loosely coupled extensibility through web services
Lifecycle management	+++	Partially: start, stop instances	Partially: only stop instances	YES: create, start, stop, modify, restart instances.
REST lifecycle management interface	+/-	NO	Partially	YES
Activities that invoke SOAP-based Web Services	+/-	YES (WSDL 1.1)	NO	YES (WSDL 1.1, WSDL 2.0)
Build services that can be invoked externally by SOAP	---	YES	NO	NO
Activities that invoke REST-based Web	---	YES (WSDL 1.1 extension)	NO	YES (WSDL 2.0, WADL, RDDL)

Services				
Build services that can be invoked externally by REST	---	YES	NO	YES (Pluggable)
Pluggable external event protocol	++	NO	NO	YES (important for XMPP)
BPMN 2.0 compatible	+++	NO	YES	YES (through transformation - partial prototype available)
BPEL 2.0 compatible	++	YES	NO	YES (through transformation - prototype available)
Build infrastructure	+/-	buildr	Ant, Maven	Rake
Interface for activity/engine management calls	++	Activities: independent integration layer, support for rest and soap Engine management: soap interface	Activities: JAVA-API Engine management: REST interface	Activities: independent integration layer (HandlerWrapper), pluggable soap, rest, mixed, .. Engine Management: rest
Component I independence (reusable components)	+++	Not very - Ode BPEL Runtime without separate repository or worklist, but provides Management API	Components separated (separated Process Repository..) but overall quite bound to Java	Strictly separated, message based system. No process repository included. Language agnostic extensions through rest interfaces / events.

Function-Point Analysis (Code Complexity) - only runtime and SOAP / REST activity calls implementation:	+++	Based on Version 1.3.3: Classes: 351.00 Methods: 1657.00 NCSS Total: 11868.00 NCSS Average: 7.16 CCN Total: 4821.00 CCN Average: 2.91	Not performed as Activiti in the evaluated version did not support web service tasks and therefore is not suitable as an execution engine.	Classes: 16.00 Methods: 75.00 NCSS Total: 524.00 NCSS Average: 6.99 CCN Total: 214.00 CCN Average: 2.85
Tests Coverage	+/-	1 test class, 7 assertions	Hundreds of tests, thousands of assertions	49 test, 137 assertions; based on workflowpatterns.org
Code quality	+/-	High (open source, community review)	High (open source, community review)	High (frequent peer reviews)
Usability	+	+	N/A	++
Adaptability	+++	N/A	N/A	+++
Performance	+/-	+	N/A	++
Consortium Affinity/Skills	+++	+	N/A	+++
Key features (differentiators)	++	Well accepted / integrated into enterprise grade infrastructure.	Editor.	Adaptation of processes, Extensibility, Performance, Scalability, Memory consumption.

### 3.8.2.3 Technology Selection

Of the three execution engines that were evaluated in detail, Activiti was ruled out as it did not offer (at least in the tested version 5.8) a comfortable way to build service orchestrations. The lack of web service tasks would make it hard to use the editor to build processes that invoke web services without the need to program invocation functionality in Java. Activiti for now relies solely on realizing activities as Java objects, which of course allows for arbitrarily powerful solutions, but would also require re-inventing a lot of features that are already present in other solutions. Due to Activiti's focus and its tight integration with Java, it can be concluded that it is probably not very recommendable to use Activiti in projects that do not base primarily on Java. Of the two remaining engines, Apache ODE and CPEE, the decision was made in favor of CPEE based on features and potential implementation effort for the ADVENTURE project. The main reasons for the decision were:

- **Performance:** the basic performance tests showed that the performance of CPEE is substantially higher than that of Apache ODE. This statement is supported by the fact that the CPEE offers explicit load-balancing through a cluster of nodes, whereas the load balancing of Apache ODE depends on possibilities of the used container (e.g. Tomcat). According to Apache however, clustering support for Apache ODE is currently under development.
- **Engine Management:** Of the desired features, Apache ODE offers only the stopping of process instances over a web service. CPEE on the other hand offers rest-based APIs for a variety of functionalities, such as the creation, starting and stopping of process instances or the modification of data elements during runtime.
- **Event publishing over web services:** The CPEE offers a much finer-grained and wider-ranged event-publishing mechanism than Apache ODE.
- **Extensibility:** the architecture of the CPEE is much looser coupled than that of Apache ODE, which makes it easier to extend. In contrast to Apache ODE, the CPEE e.g. offers well defined APIs and protocols to enable new script languages or messaging protocols.
- **Consortium Affinity/Skills:** As the CPEE has been developed for other projects by the University of Vienna, UVI has very in-depth knowledge of all aspects of the CPEE. The CPEE can be easily adapted and extended to fit the needs of ADVENTURE.

#### 3.8.2.3.1 Conclusion

The adaptive execution engine CPEE was selected, as it is open source, has a high performance, offers extensive features concerning engine management and event publishing, and is easily extensible. It is concluded that these aspects make the CPEE a great research and prototyping tool, which will save not only time, but will facilitate the development of independent components that will be reusable for other projects and engines. Furthermore the responsible partner (UVI) already has in-depth knowledge of all CPEE's aspects.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>281</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### 3.8.2.4 Missing Elements and Implementation Needs

The CPEE covers many of the aspects that are needed by ADVENTURE, but has to be extended to support the following:

- Correlation Engine that can directly deal with XMPP messages and XMPP's asynchronous nature instead of SOAP/REST messages.
- XMPP based Interaction (instead of calling SOAP/REST services, interact through XMPP with gateway), implementing ADVENTURE XMPP synchronous, asynchronous, and continuous invocation.
- Special ADVENTURE Activity Types (as specified in D3.2):
- Spawning ADVENTURE Processes
- Translating the format saved by the Process Editor:
  - To internal CPEE format.
  - To simulation specific format.
  - To format suitable for optimization.
- Extend execution / execution shaping event system to meet ADVENTURE needs.
- Delegate evaluation of conditions / script tasks to external services: allow for REST/XMPP delegation. Transfer of process context as well endpoints. Each evaluation from the point of view of the engine is an atomic operation and therefore behaves like the implementation of STM<sup>15</sup> (Software Transactional Memory).
- Optimizations for ADVENTURE specific needs
- Performance: Automatic load balancing for CPEE clusters and fast instance spawning.
- Storage: Currently the whole process engine with all instances is kept in memory. The engine needs to be expanded to support the persisting of finished or temporarily stopped instances into pluggable storage containers (e.g. ADVENTURE), through pluggable protocols (REST/XMPP).
- Improvements regarding Adaptation & Simulation: These will be described in more detail in the respective sections.

### 3.8.3 Technical Component Specification

As explained above, the ADVENTURE SPE component will be a set of loosely coupled cooperating services that cover all execution aspects required by a virtual factory. As mentioned this component structure has been selected, in order to make it possible to easily replace the process engine at the core of the project. As shown in Figure 73, ADVENTURE requires the core process engine to expose four interfaces, which are augmented by 3 external services (red border) that implement the ADVENTURE requirements (which may not necessarily be a feature of a particular

<sup>15</sup> I.e. script tasks are atomic blocks that are able to modify the process context, but not in parallel. They also behave like actors (Actor pattern), as they do their work asynchronously.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>282</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

process engine) as independent components. Also included in Figure 73 are other ADVENTURE components (dashed border) that reuse the same interfaces.

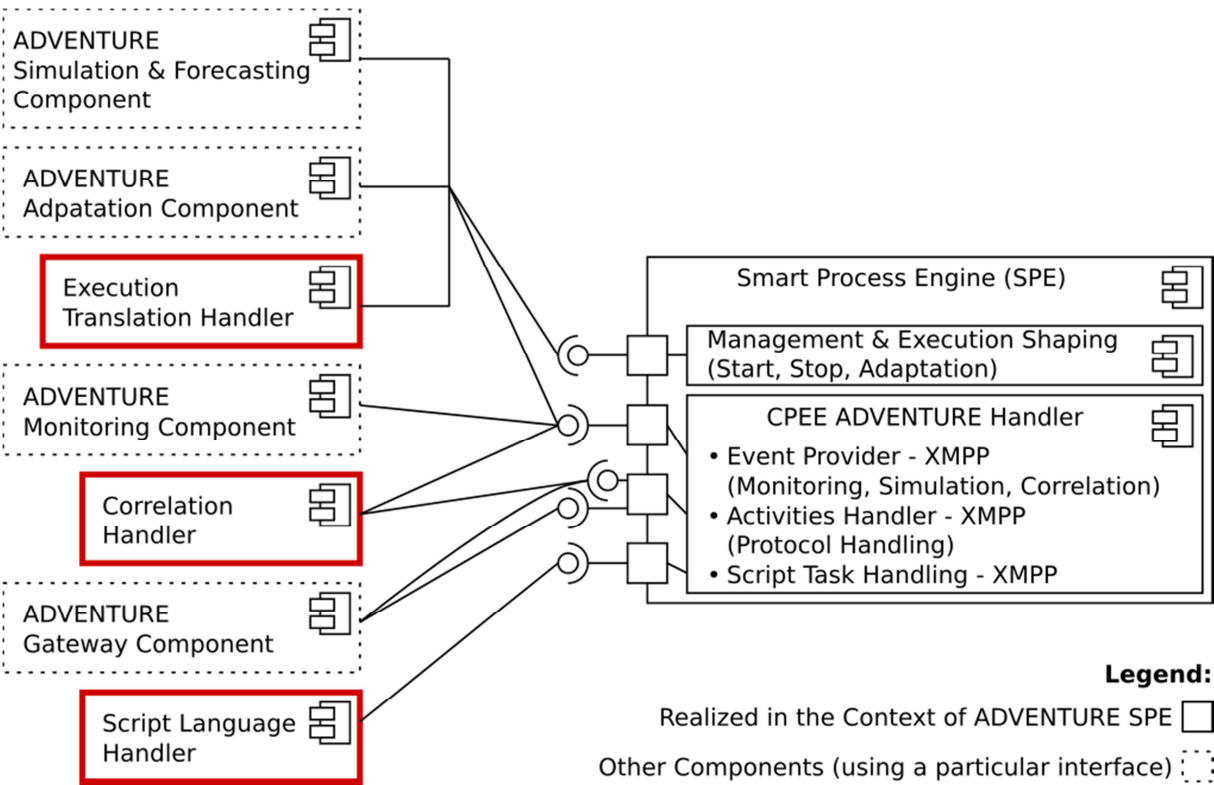


Figure 73 - ADVENTURE SPE Component Structure

Thus the ADVENTURE SPE component itself consists of the following parts:

3.8.3.1 Management & Execution Shaping Interface

The Management & Execution Shaping interface will offer a standard set of operations to instantiate, start, stop and modify properties of process instances. A detailed overview of all possible operations is shown in Figure 73.

3.8.3.2 CPEE ADVENTURE Handler

As shown in Figure - ADVENTURE SPE Component Structure - above, this handler is an internal component of the ADVENTURE SPE component, dealing with three different tasks:

- Provide a set of events, covering all aspects of process management and execution (see sub-section 3.8.4.3 for details).
- Allow the invocation of gateways (or transformations) represented by activities in a process.
- Delegate the evaluation of script tasks or conditions in a script language that the engine does not support (e.g. JavaScript) to an external component. The location

(address) of the external component handling is assumed to be configured as part of each process or process instance model.

Because the first two aspects provide integration with other ADVENTURE components, they have to adhere to the messaging requirements - XMPP. The third aspect however, is only used by the SPE component, and will thus be integrated as described in subsection Script Language Handler below.

### 3.8.3.3 Execution Translation Handler

The purpose of the Execution Translation (ET) Handler is to translate the process description language to the internal format used in the process engine, whenever:

- A process is instantiated from a process model saved by the ADVENTURE process designer component, or
- A process instance model is changed by the ADVENTURE adaptation component.

As for ADVENTURE the CPEE was selected, this will be the CPEE Domain Specific Language (DSL), described in<sup>16</sup>.

The Translation Handler will be implemented as follows:

- The language selected by ADVENTURE is expected to be gradually refined during the project. Whenever possible we will refine an accompanying XSLT to transform the process model.
- Should the model be too complex for XSLT (e.g. ADVENTURE decides to allow multiple end states, or BPMN flexible ad-hoc groups), the XSL transformation will be replaced with a pluggable implementation in a programming language.

The Transformation Handler will be a simple request/response XMPP service with the following signature:

- Input: (1) process model, (2) target format, e.g. "CPEE DSL", (3) transformation logic, i.e. either a XSL snippet, or a JavaScript function to be evaluated in the Script Language Handler.
- Output: New Process Model Format.

### 3.8.3.4 Correlation Handler

The purpose of the Correlation Handler is to listen to the stream of XMPP messages, select messages addressed to the SPE component and deliver it to particular process instances. In order to find addressable process instances it will listen to the events provided by the SPE component, and find all instances with activities that currently wait for messages. In order to identify the messages it will:

<sup>16</sup> Juergen Mangler, Gerhard Stuermer and Erich Schikuta: „Cloud Process Execution Engine - Evaluation of the Core Concepts”, Arxiv 2010, <http://arxiv.org/abs/1003.3330>.

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>284</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



- Let pass messages that are already correctly addressed to process instances and process activities, by using the XMPP resource property for identifying instances, and the id property for identifying individual activities.
- Analyze the contents of the message according to the logic specified in the process (by the Process Designer Component). This logic can be either simple expressions or complex conditions (JavaScript functions returning true/false) evaluated by the Script Handler Component.

The Correlation Handler will be realized as an XMPP service, and provides no special interface.

### 3.8.3.5 Script Language Handler

The Script Language Handler will be an independent service, handling the choice of script language specified in a process saved by in the ADVENTURE Process Designer. For ADVENTURE JavaScript was selected – see chapter 3.7.1 The Script Language Handler is an independent component in order to isolate it from the rest of the SPE – it will be possible to use standalone runtimes, even if the license of the runtime does not fit to the rest of the ADVENTURE provided tools. The Script Language Handler will handle two different types of scenarios in ADVENTURE:

(1) Evaluate conditions for xor's, loops, etc.

- Input: JavaScript condition, process context (endpoints, data elements)
- Output: true/false

(2) Evaluate code for script tasks

- Input: JavaScript code, process context (endpoints, data elements)
- Output: new process context (endpoints, data elements)

The selection of JavaScript as a script language, led to the selection of Node.js as an implementation technology. Node.js<sup>17</sup> is a JavaScript-based platform to ease the creation of fast and scalable network applications. The Script Task Handler will provide a REST interface with the two above described resources (PUT operation). It will use JSON to communicate with the SPE. XMPP and XML have been discarded as protocols, due to the relative complexity (and overhead) of this solution, in the light that the Script Handler in use only (internally) by the ADVENTURE SPE component.

---

<sup>17</sup> <http://nodejs.org/>

3.8.3.6 Adaptation

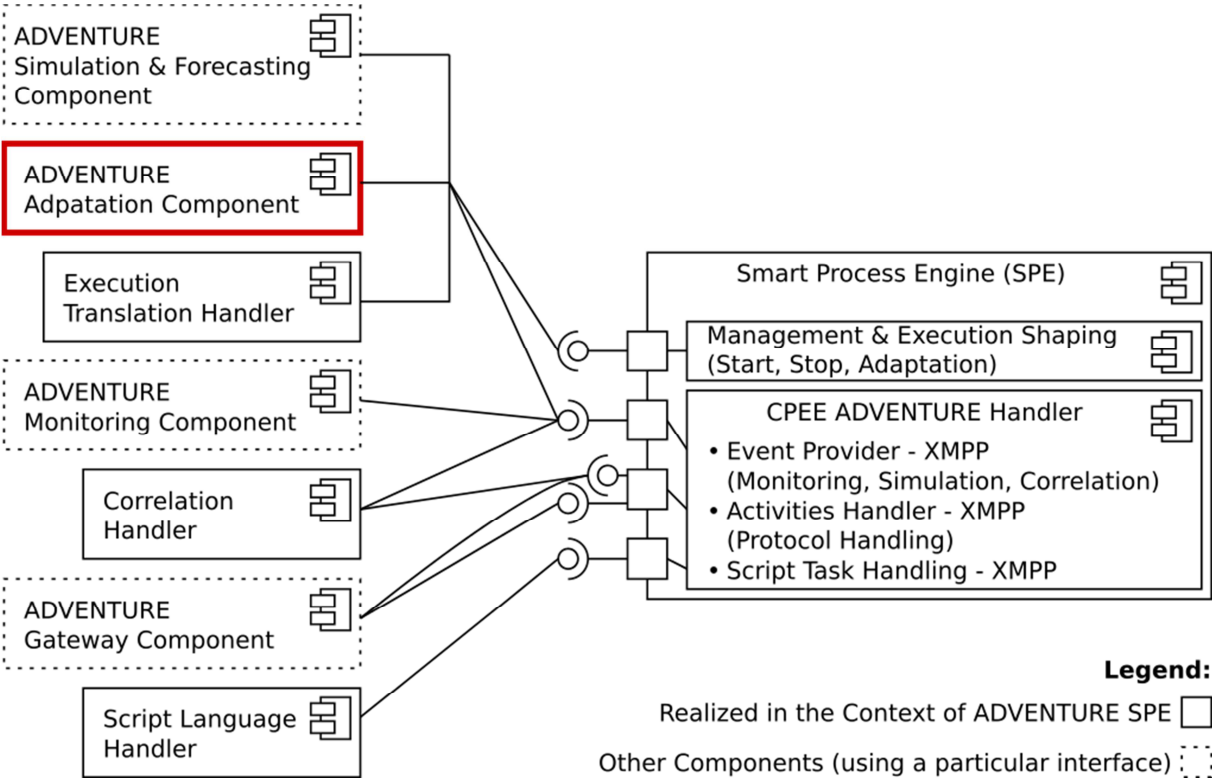


Figure 74 - Adaptation

The Adaptation Component will apply changes (due to optimization and repair actions) to running process instances. In order to do so it will utilize the Management & Execution Shaping Interface, as well as listen to events from process instances to wait for certain points in process execution before interrupting (i.e. while parallel branches are executed, an adaptation or repair application may not be feasible).

3.8.3.6.1 Major Design Decisions

This component is independent from Smart Process Execution as many existing Process Engines miss capabilities that are needed for such advanced functionality at instance level. By separating this component from the SPE, and reliance on only the well interfaces described above, this functionality can be reused for other process engines (whenever they gain the required feature set, and thus can be wrapped in a layer that implements the above specified interfaces).

It was decided that this component realizes algorithms developed by Rinderle-Ma et al. (see <sup>18</sup> and <sup>19</sup>), which are to the best of the consortiums knowledge still state of the art, and not fully realized in any existing Process Management Environment.

3.8.3.6.2 Missing Elements & Implementation Needs

The algorithms mentioned above have to be realized by in conjunction with the set of events and execution shaping & management operations detailed above.

3.8.3.7 Forecasting & Simulation

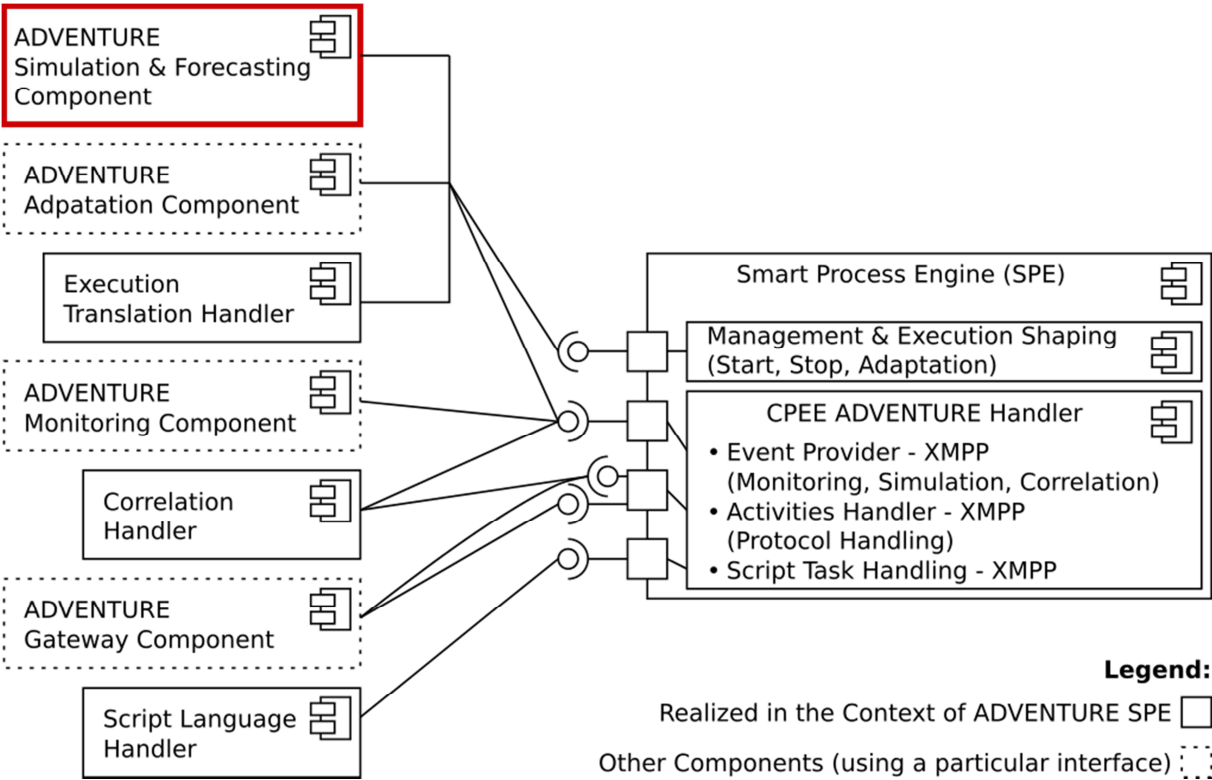


Figure 75 - Forecasting & Simulation

While Forecasting is about calling simple activities and receiving statistical (duration, other KPIs) and semantic (actual return values) information, Simulation is about executing multiple activities from different partners that depend on each other through input and output – process.

<sup>18</sup> [1] S. Rinderle, M. Reichert, and P. Dadam, “Correctness Criteria for Dynamic Changes in Workflow Systems – A Survey.,” *Data and Knowledge Engineering*, vol. 50, no. 1, pp. 9–34, 2004.

<sup>19</sup> [1] S. Rinderle, A. Wombacher, and M. Reichert, “On the Controlled Evolution of Process Choreographies.,” in *Int’l Conf. on Data Engineering*, 2006, p. 124.

### 3.8.3.7.1 Major Design Decisions

This component relies solely on the features of the Smart Process Execution to achieve its goals. Processes are executed with a mode of the CPEE ADVENTURE Handler, that instead of calling activities, calls the respective simulation interfaces, and they makes the information available as normal event. The simulation component then collects, refines, stores and presents this information.

Because of the above given definition of simulation, and the way it is intended to work, no technology selection has been performed. Instead it was ensured that the SPE can fulfill all functional requirements from simulation.

## 3.8.4 Specification of Interfaces, Protocols and Formats

### 3.8.4.1 XMPP integration

The SPE, as all other component will communicate through the XMPP protocol with other ADVENTURE components. For the interaction of the SPE component with other components the following aspects have to be considered:

- How to address the SPE component?
- How to address particular process instances, if a message is known to belong to particular instance?
- How to address particular activities inside process instance, as multiple branches in a process might wait for messages in parallel?

XMPP addressing (as explained in section 3.4) typically has the following form:

component@factory/resource

In the case of the SPE component this translates to:

adventure\_processexecution@fp7-adventure.eu/instance

Thus it is possible to identify the sending or receiving instance for each message. Furthermore the *id* attribute of each XMPP message is reused as a means to identify single activities in a process instance. The Message may either originate or be sent to these activities. Thus a valid XMPP message from the process instance 7, activity “order”, to the gateway of the company “ETMCO” may look like:

```
<adventuremessage>
  <msgid>someId</msgid>
  <from>adventure_processexecution@fp7-adventure.eu/7</from>
  <to>adventure_gateway@fp7-adventure.eu/ETMCO</to>
  <content>...<id>order</id>...</content>
</adventuremessage>
```

The XML payload itself has to conform:

- To syntactic requirements described in Section 3.6.4.2 ADVENTURE Gateway component).

D3.3-Technical-Specification	Author: UVA and Partners	Date: 2012-09-24	Page: <b>288</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- To semantic requirements specified by an ADVENTURE broker while modeling a particular process.

The example above shows a specifically addressed message. But the ADVENTURE SPE component itself may receive messages that could not be assigned to instances or activities – the resource (instance) and id (activity) are missing.

```
<adventuremessage>
  <messageld>someld</messageld>
  <from> adventure_gateway@fp7-adventure.eu/ETMCO </from>
  <to> adventure_processexecution@fp7-adventure.eu</to>
  <content>...</content>
</adventuremessage>
```

In this case the ADVENTURE Correlation Engine has to extract the operation from the message and find out which instance/activity is waiting for such a message, as well as possibly further analyze the content according to the correlation rules specified in the process instance model.

#### 3.8.4.2 Management & Execution Shaping Interface

This interface contains a set (tree) of resources, to create and explore instances, find out their status and change their properties. The structure of this interface is depicted in Figure 76.

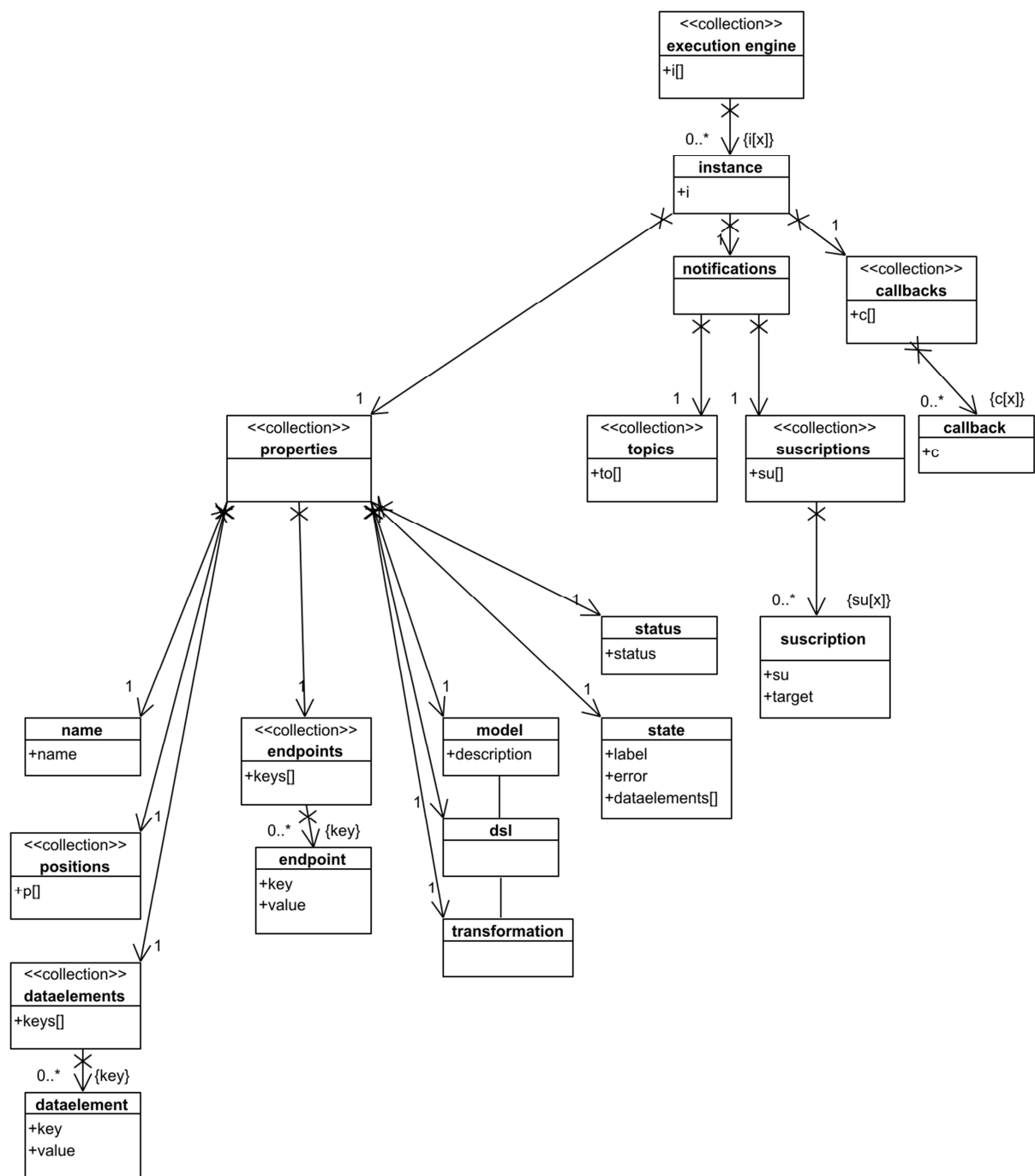


Figure 76 - Process Instances – Structure of Interface

A detailed analysis of each object in the tree can be found in Table 14 - Management & Execution Shaping Interface.

Table 14 - Management &amp; Execution Shaping Interface

Locator	Operation	Input	Output	Description
/	READ		XML <sup>1</sup> {i <sub>1</sub> ..i <sub>x</sub> }	List of instances
/	CREATE	{name}	{i <sub>x</sub> }	Create new instance
/i <sub>x</sub> }	READ			Instance Details
/i <sub>x</sub> }	DELETE			Delete Instance
/i <sub>x</sub> }/properties	READ		XML <sup>2</sup>	List of properties
/i <sub>x</sub> }/properties	UPDATE	XML <sup>2</sup>		Set all properties
/i <sub>x</sub> }/properties/name	READ		{name}	Name as set above
/i <sub>x</sub> }/properties/position	READ		XML <sup>3</sup>	Where are the users in a running process instance? May be more than one position, as parallel branches are possible.
/i <sub>x</sub> }/properties/position	UPDATE	XML <sup>4</sup>		Set positions
/i <sub>x</sub> }/properties/dataelements	READ		XML <sup>4</sup> {d <sub>1</sub> ..d <sub>x</sub> }	List of data elements with name and value
/i <sub>x</sub> }/properties/dataelements	UPDATE	XML <sup>4</sup>		Update data elements
/i <sub>x</sub> }/properties/dataelements	CREATE	XML <sup>4a</sup>		Data element key/value pair
/i <sub>x</sub> }/properties/dataelements/{d <sub>x</sub> }	READ		{d <sub>x</sub> }	Value of data element

/ {i <sub>x</sub> } / properties / dataelements / {d <sub>x</sub> }	UPDATE	{d <sub>x</sub> }		Value of data element
/ {i <sub>x</sub> } / properties / dataelements / {d <sub>x</sub> }	DELETE			Delete data element
/ {i <sub>x</sub> } / properties / endpoints	READ		XML <sup>5</sup> {e <sub>1</sub> .. e <sub>x</sub> }	List of endpoints with name and value
/ {i <sub>x</sub> } / properties / endpoints	UPDATE	XML <sup>5</sup>		Update endpoints
/ {i <sub>x</sub> } / properties / endpoints	CREATE	XML <sup>5a</sup>		Endpoint key/value pair
/ {i <sub>x</sub> } / properties / endpoints / {e <sub>x</sub> }	READ		{e <sub>x</sub> }	Value of endpoint
/ {i <sub>x</sub> } / properties / endpoints / {e <sub>x</sub> }	UPDATE	{e <sub>x</sub> }		Value of endpoints
/ {i <sub>x</sub> } / properties / endpoints / {e <sub>x</sub> }	DELETE			Delete endpoint
/ {i <sub>x</sub> } / properties / description	READ		XML <sup>6</sup>	Unspecified description, e.g. BPEL
/ {i <sub>x</sub> } / properties / description	UPDATE	XML <sup>6</sup>		Set description
/ {i <sub>x</sub> } / properties / status	READ		XML <sup>7</sup>	Semantic process status. E.g. reason for an error; result when finished.
/ {i <sub>x</sub> } / properties / state	READ		ready   running   stopping   stopped   finished	Current status of instance.
/ {i <sub>x</sub> } / properties / state	UPDATE		ready → running running → stopping	Only the states running and stopping can be set from outside, all other states are reached internally.
/ {i <sub>x</sub> } / notifications	READ			Information about notifications



/i <sub>x</sub> /notifications/topics	READ		XML <sup>8</sup> {to <sub>1</sub> ..to <sub>x</sub> }	List of all possible events that are observable from outside. Component that may consume these events: Process Monitoring.
/i <sub>x</sub> /notifications/subscriptions	READ		XML <sup>9</sup> {su <sub>1</sub> ..su <sub>x</sub> }	A list of all active subscriptions, e.g. the process monitor may have an active subscription, as well as the process editor to show the progress of the execution of a single instance.
/i <sub>x</sub> /notifications/subscriptions	CREATE	{to <sub>1</sub> ..to <sub>x</sub> }, {target}		A list of topics a certain {target} wants to subscribe to. The target is intended to be a unique identifier that is understood by the messaging component, so that events are ONLY delivered to a certain component, in a certain context (e.g. logging of instance 7).
/i <sub>x</sub> /notifications/subscriptions/{su <sub>x</sub> }	READ		XML <sup>10</sup>	XML <sup>10</sup> contains details about the subscription to each topic.
/i <sub>x</sub> /notifications/subscriptions/{su <sub>x</sub> }	DELETE			Delete subscription.
/i <sub>x</sub> /callbacks	READ		XML <sup>11</sup> {c <sub>1</sub> ..c <sub>x</sub> }	List of all callbacks, for asynchronous data exchange. These callbacks are created during execution whenever an process step waits for some input. It is also used for events that require feedback from the event {target} (see above). Example: allow Adaptation Engine to deny starting of process (changing state to running). Used by the correlation engine.
/i <sub>x</sub> /callbacks/{c <sub>x</sub> }	UPDATE	ANY XML		Answering with ANY information will forward this information to the process step or event that

				waits for the information. Afterward the callback is automatically deleted.
<code>/i_x/callbacks/c_x</code>	DELETE			Manually delete callback. May lead to state: stopped.

The above enumerated paths are to be used as the *subject* of an XMPP message. The operation is to be included in the *body* section as payload. Example:

```
<message
  to='adventure_processexecution@fp7-adventure.eu/7'
  from='adventure_gateway@fp7-adventure.eu/ETMCO'
  subject='properties/state'
<body>
  <operation>GET</operation>
  <payload>XML payload</payload>
</body>
</message>
```

The remainder of this subsection is dedicated to give examples of payload, according to the specification in the table above:

**XML<sup>1</sup>:** The list of instances has to at least include an identifier. Example:

```
<payload>
  <instance id='1' base='process1'>Order 1</instance>
  <instance id='2' base='process1'>Order 2</instance>
</payload>
```

The parameter base holds the information, which process model the instance was initially created from.

**XML<sup>2</sup>:** The minimum sets of properties to be included are:

name: short description of the instance

positions: while an instance is execution, identify the currently active steps. A stopped instance holds steps that have been finished so far.

dataelements: all data elements that are available to the process instance. This includes results from the execution of process steps as well as data created by the process through script tasks.

endpoints: specific ADVENTURE partner gateways.

The properties are only to be changed while the instance is in state stopped, as explained in D2.2. Example:

```

<payload>
  <name>Order 1</name>
  <state>finished</state>
  <handlerwrapper>CPEE_ADVENTURE_Handler</handlerwrapper>
  <positions>
    <activity1/>
  </positions>
  <dataelements>
    <element1>"Hello World"</element1>
    <input1>"</input1>
    <test1>"</test1>
  </dataelements>
  <endpoints>
    <ep1>adventure_gateway@adventure-fp7.eu/ETMCO</ep1>
  </endpoints>
  <dsl>Internal Representation</dsl>
  <status>
    <id>0</id>
    <message>undefined</message>
  </status>
  <description>
    Process Model as specified by ADVENTURE Process Designer.
  </description>
  <transformation>
    Means of transforming the Process Model to an
    Internal Representation.
  </transformation>
</payload>

```

**XML<sup>3</sup>:** A list of process step identifiers. Example:

```

<positions>
  <activity1/>
</positions>

```

Identifiers (in this example activity1) can occur more than once, in case they have been spawned by a loop that generates separate threads (i.e. can occur with exclusive and parallel event based gateways in BPMN).

**XML<sup>4</sup>**: A list of data elements, consisting of unique key and value. The value itself is allowed to be expressed in JSON and consist of complex data (e.g. arrays, hashes) as allowed by the JSON format. Example:

```
<dataelements>
  <element1>"Hello World"</element1>
  <input1>" "</input1>
  <test1>" "</test1>
</dataelements>
```

**XML<sup>4a</sup>**: A single key, value pair. For values the same restriction as above is mandatory. Example:

```
<element1>"Hello World"</element1>
```

**XML<sup>5</sup>**: A list consisting of unique key and value. The value itself is allowed to be arbitrary data, e.g. REST urls, SOAP operations identifiers (including the endpoint of the soap service). Example:

```
<endpoints>
  <ep1>adventure_gateway@adventure-fp7.eu/ETMCO</ep1>
</endpoints>
```

The values are to be interpreted by the SPE to decide at runtime, in order to be able to contact the “ETMCO” gateway (including data preparation) and interpret the resulting.

**XML<sup>5a</sup>**: A single key, value pair. For values the same restriction as above is mandatory. Example:

```
<ep1>adventure_gateway@adventure-fp7.eu/ETMCO</ep1>
```

**XML<sup>6</sup>**: The Process Model format as delivered by the process editor. This format cannot yet be specified as it may include ADVENTURE research specific bits and pieces. The model format has to include a transformation specification to a unified execution format, as specified by Mangler et al. <http://arxiv.org/abs/1003.3330>.

**XML<sup>7</sup>**: Holds all kinds of information regarding the status of a running instance (not about currently executed process steps):

Error codes and error messages in case of errors.

Data elements that are marked as result in case of state **finished**.

Information about milestones that have been passed while executing an instance.

Example:

```
<status>
  <id>0</id>
  <message>undefined</message>
</status>
```

**XML<sup>8</sup>**: A preliminary table of possible notifications is shown below. For most topics (i.e. properties as enumerated above) events are to be sent out when they are changes. For the state running, an event has to be sent out whenever an activity is called, fails, manipulates data elements, or is finished. The concept of **Voting** is needed to define special events that require feedback from the event target (as explained above). By using this mechanism it is planned to realize the rule-based interaction with processes.

```
<topics
  <topic id="running">
    <event>calling</event>
    <event>failed</event>
    <event>manipulating</event>
    <event>done</event>
    <vote>before</vote>
    <vote>after</vote>
  </topic>
  <topic id="position">
    <event>change</event>
  </topic>
  <topic id="description">
    <event>change</event>
    <event>error</event>
  </topic>
  <topic id="state">
    <event>change</event>
    <vote>change</vote>
  </topic>
  <topic id="status">
    <event>change</event>
  </topic>
  <topic id=" dataelements">
    <event>change</event>
```

```

</topic>
<topic id="endpoints">
  <event>change</event>
</topic>
</topics>

```

All mentioned events are detailed in subsection Event Model below.

**XML<sup>9</sup>:** A list of all targets that currently receive notifications for a particular instance.

```

<subscriptions>
  <subscription>adventure_monitoring@adventure-fp7.eu</subscription>
  <subscription>adventure_adaptation@adventure-fp7.eu</subscription>
</subscriptions>

```

**XML<sup>10</sup>:** An enumeration of all active events as well as the target of a particular subscription.

```

<subscription>
  <topic id="activity">
    <event>calling</event>
    <event>manipulating</event>
    <event>failed</event>
    <event>done</event>
  </topic>
  <topic id="position">
    <event>change</event>
  </topic>
  <topic id="description">
    <event>change</event>
    <event>error</event>
  </topic>
  <topic id="state">
    <event>change</event>
  </topic>
  <topic id="dataelements">
    <event>change</event>
  </topic>
  <topic id="endpoints">
    <event>change</event>
  </topic>
</subscription>

```

```

</topic>
<topic id="handlers">
  <event>change</event>
</topic>
</subscription>

```

**XML<sup>11</sup>:** A list of callback identifiers. The callback identifiers are known to the correlation engine to find out which activities are waiting for input.

```

<payload>
  <callback id='xxx1ffad2' source='activity1'>
    Correlation Rules
  </callback>
  <instance id='xxx1ffad3' source='activity7' />
</payload>

```

In the example above, *activity1* had some correlation specified in the ADVENTURE Process Designer, whereas, *activity7* can be ignored (unless specified wrong this is a synchronous communication, thus directly addressed with instance and activity).

### 3.8.4.3 Event Model

To communicate its state to other components the ADVENTURE SPE can deliver a set of events. By default the ADVENTURE Monitoring component receives all events, other components (ADVENTURE Adaptation, Simulation components) can subscribe to a subset of events as described above. The following diagram (Figure 77) shows the event model of the Smart Process Engine:



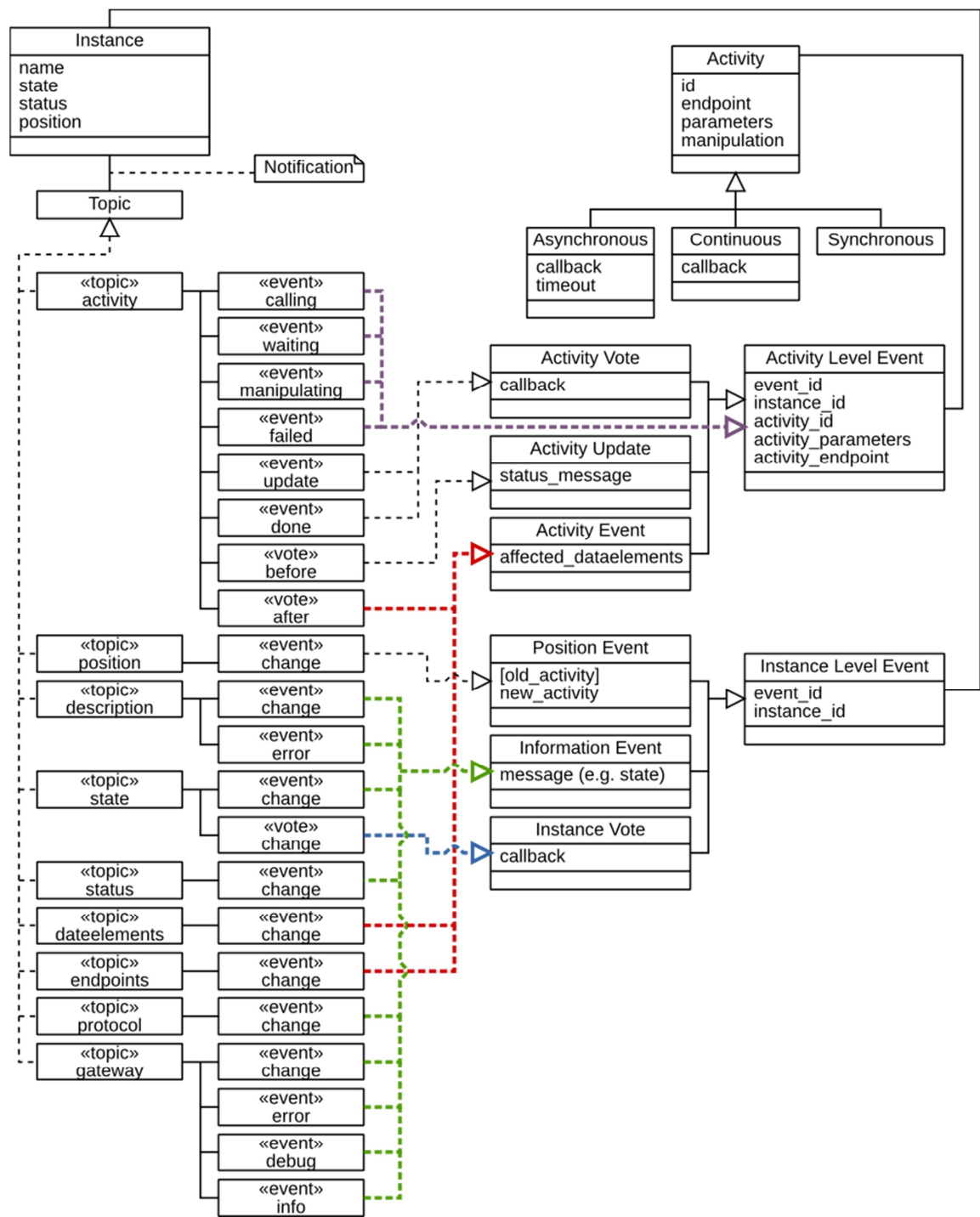


Figure 77 - SPE Event Model

As shown Figure 77, SPE Event Model above:

**Instance Level Events** are fired when something at instance level happens. E.g. changes to data elements can originate from activities, but they do not have to, as also the ADVENTURE Broker (human), or ADVENTURE Adaptation Component may

change dataelements. These events can occur at runtime (while an instance is enacted) or while an instance is stopped.

- **Activity Level Events** are fired solely during the enactment of activities at runtime.

In the first group belong:

- **Position Events:** The current position (runtime) or position, from which a process instance may start, is changed.
- **Information Events:**
- **gateway/\*:** Semantic information about the state of an ADVENTURE gateway. As e.g. errors may either occur in the process (syntactic) or during the execution of an activity (semantic), the gateway may push information about the health of services that it encapsulates, independently of the execution of activities. These messages contain the interpreted information received by an ADVENTURE gateway.
- **protocol/change:** For ADVENTURE debugging and testing purposes, the XMPP interface of the ADVENTURE SPE component will be changeable at runtime. These messages contain an internal name referencing the handler currently used, e.g. "CPEE ADVENTURE Handler".
- **endpoints/change, dataelements/change:** These are changeable either at runtime or while an instance is stopped. These events may occur in parallel with Activity Level Events. These messages contain the before/after version of endpoints/dataelements in the format specified in subsection Management & Execution Shaping below.
- **description/change:** Whenever the process instance model is changed, this event is sent out. Can only occur while process instance is stopped. These messages contain a process model in the format specified in section 3.7.
- **status/change** events are triggered by script tasks during execution. These messages contain the new status as specified in subsection Management & Execution Shaping below.
- **state/change** events occur whenever an instance is created, stopped, started or finishes. These messages contain the state (ready, finished, running, stopping, stopped) as specified in D2.2.
- **Instance Votes:** Whenever a state is to be changed, a vote might occur as specified in D2.2. By this mechanism e.g. the ADVENTURE Adaptation component can disallow an ADVENTURE broker to start an instance while it is modifying it.

The second group deals solely with the stream of events that occurs while an activity is executed. They have the following structure:

D3.3_Technical Specification_v2.2-final	Authors: UVA and Partners	Print Date: 2012-09-24	Page: 302 / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

```

<event>
  <event_id>activity/calling</event_id>
  <activity_id>activity1</activity_id>
  <activity_endpoint>
    adventure_gateway@fp7-adventure.eu/ETMCO
  </activity_endpoint>
  <activity_parameters>
    All input parameters to this activity as key, value pairs.
    If the size of a particular value exceeds a certain size,
    the it may be replaced by a link to the cloud storage.
  </activity_parameters>
  ...
</event>

```

The *instance\_id* mentioned in Figure 77– SPE Event Model - above, is included at the XMPP level in the *from* property.

For this generic format several subtypes exist:

- **activity/update, activity/done** events are only sent during the execution of a *continuous* activity as specified in D2.2. These events include an element *<status\_message>* that holds an interpreted string containing an update message (e.g. 15% done) sent through the ADVENTURE Gateway.
- **dataelements/change, endpoints/change, activity/after** contain These messages contain the before/after version of endpoints/dataelements in the format specified in subsection Management & Execution Shaping Interface below.
- **activity/before, activity/after** votes can be used to delay the enactment of a process, by external services. This will be used by the ADVENTURE Simulation component.

#### 3.8.4.4 Adaptation

Table 15 - Adaptation Interface

Locator	Operation	Input	Output	Description
/	CREATE	$m_{id}, \{i_1..i_x\}$	$\{i_1..i_y\}$	Apply a change to a set of instances.

The Adaptation Interface has only a minimal interface. It takes the process model with included changes  $m_{id}$  and a list of instances,  $\{i_1..i_x\}$  to which to apply the changes as input. Each process models  $m_{id}$  needs to include information about its preceding revision in order to be able to calculate the difference. Each instance  $i_x$  also needs to include information about the model it has been instantiated from, or preceding revisions in case of earlier repair or adaptations. The result is subset of the instances,

D3.3_Technical Specification_v2.2-final	Authors: UVA and Partners	Print Date: 2012-09-24	Page: 303 / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

$\{i_1..i_y\} \subseteq \{i_1..i_x\}$  as it might not be able to apply the changes to all instances. It is also possible that some instances receive part of the changes, as explained above.

### 3.8.4.5 Forecasting & Simulation

This subsection contains a description of interface that can be used to trigger forecasts / simulations, as well as access the results of these operations.

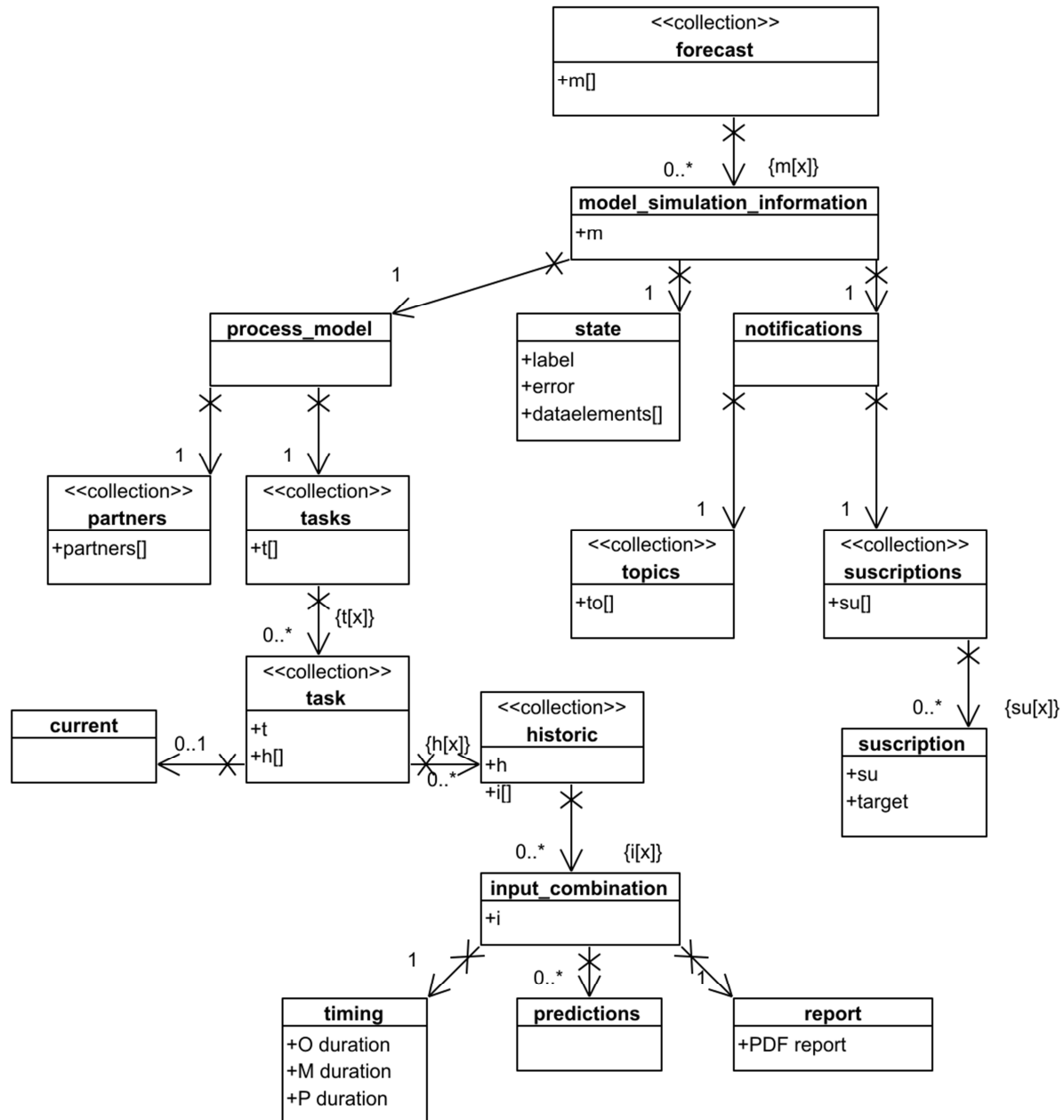


Table 16 - Forecasting &amp; Simulation Interface

Locator	Operation	Input	Output	Description
/	READ		XML <sup>1</sup> {m <sub>1</sub> ..m <sub>x</sub> }	List of available process models for simulation.
/ {m <sub>x</sub> }	READ			Overview of all simulation information for a given process model.
/ {m <sub>x</sub> }/model	READ		XML <sup>2</sup>	Process model used for this particular simulation.
/ {m <sub>x</sub> }/model/partners	READ		XML <sup>3</sup>	All partners associated with all task in the process model
/ {m <sub>x</sub> }/model/tasks	READ		XML <sup>4</sup> {t <sub>1</sub> ..t <sub>x</sub> }	A list of tasks contained in the process model.
/ {m <sub>x</sub> }/model/tasks/{t <sub>x</sub> }	READ		XML <sup>5</sup> {h <sub>1</sub> ..h <sub>x</sub> }	A (historic) list of forecast results for a given task.
/ {m <sub>x</sub> }/model/tasks/{t <sub>x</sub> }/current	READ		XML <sup>6</sup>	Details about the most recent forecast. The substructure of this resource will be identical to / {m <sub>x</sub> }/model/tasks/{t <sub>x</sub> }/{h <sub>x</sub> }/...
/ {m <sub>x</sub> }/model/tasks/{t <sub>x</sub> }/{h <sub>x</sub> }	READ		XML <sup>6</sup> {i <sub>1</sub> ..i <sub>x</sub> }	Overview of a particular forecast. As tasks may depend on each other, and return a range of results, their will also be multiple invocations with multiple values for each task. Thus for each input combination separate forecasting results may be possible.

/ {m <sub>x</sub> } / model / tasks / {t <sub>x</sub> } / {h <sub>x</sub> } / {i <sub>x</sub> }	READ			Overview of the forecast details for a certain input combination.
/ {m <sub>x</sub> } / model / tasks / {t <sub>x</sub> } / {h <sub>x</sub> } / {i <sub>x</sub> } / timing	READ		XML <sup>7</sup>	O, M and P durations as specified above.
/ {m <sub>x</sub> } / model / tasks / {t <sub>x</sub> } / {h <sub>x</sub> } / {i <sub>x</sub> } / predictions	READ		XML <sup>8</sup>	A set of predicted values as specified above. Each item in the set is structured according to the result that a normal invocation of the gateway would yield.
/ {m <sub>x</sub> } / model / tasks / {t <sub>x</sub> } / {h <sub>x</sub> } / report	READ		PDF	An aggregated report for a particular simulation.
/ {m <sub>x</sub> } / state	READ		ready   running   stopping   stopped	Current status of simulation. After a simulation finishes, the instance changes to state ready. A simulation may be suspended.
/ {m <sub>x</sub> } / state	UPDATE		ready → running running → stopping	Only the states running and stopping can be set from outside, all other states are reached internally.
/ {m <sub>x</sub> } / notifications	READ			Information about notifications
/ {m <sub>x</sub> } / notifications / topics	READ		XML <sup>9</sup> {to <sub>1</sub> ..to <sub>x</sub> }	List of all possible events, that are observable from outside. Component that may consume these events: Process Monitor.
/ {m <sub>x</sub> } / notifications / subscriptions	READ		XML <sup>10</sup> {su <sub>1</sub> ..su <sub>x</sub> }	A list of all active subscriptions, e.g. the process monitor may have an active subscription, as well as the process editor to show the progress of the execution of a simulation.

/ {m <sub>x</sub> }/notifications/subscriptions	CREATE	{to <sub>1</sub> ..to <sub>x</sub> }, {target}		A list of topics a certain {target} wants to subscribe to. The target is intended to be a unique identifier that is understood by the messaging component, so that events are ONLY delivered to a certain component, in a certain context (e.g. logging of instance 7).
/ {m <sub>x</sub> }/notifications/subscriptions/{su <sub>x</sub> }	READ		XML <sup>11</sup>	XML <sup>12</sup> contains details about the subscription to each topic.
/ {m <sub>x</sub> }/notifications/subscriptions/{su <sub>x</sub> }	DELETE			Delete subscription.

In this section we analyze the exchanged data as described in the Table 16:

**XML<sup>1</sup>:** The list of models has to at least include an identifier for each model.

**XML<sup>2</sup>:** A particular version of the process model  $m_x$ . The format is not yet specified and will probably evolve during the creating of the process editor.

**XML<sup>3</sup>:** A list of partners, with each partner represented by a resource identifier as specified in section Data Provisioning and Discovery (i.e. each activity in the process model can have multiple partners assigned). A process model  $m_x$  may contain multiple tasks  $t_x$ , and each task may be carried out by several partners. This list will be a superset of all partners connected to all tasks.

**XML<sup>4</sup>:** A list of tasks  $t_x$  contained in a particular process model  $m_x$ . Control structure is not enumerated as only for tasks information is delivered from the gateways.

**XML<sup>5</sup>:** A list of historic simulations for this process model. Every simulation has a unique ID, a date and comments.

**XML<sup>6</sup>:** A list of possible input value combinations for each task. Each list item is structured according to the input parameters of the call the gateway for this particular task.

**XML<sup>7</sup>:** The O, M and P durations as described above.

**XML<sup>8</sup>:** The set of predicted values as described above. Example:

```
<?xml version="1.0"?>
<predictions xmlns="http://fp7-adventure.eu/ns/simulation/prediction/1.0">
  <prediction probability="0.7">
    <attribute
      name='status'
      type='numeric'
      characteristic='range'>1..7</attribute>
    <attribute
      name='quality'
      type='other'
      characteristic='value'>JSON</attribute>
    ...
  </prediction>
  ...
</predictions>
```

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>308</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



</predictions>

**XML<sup>9</sup>:** There is only few notification topics available, as the only interaction with the simulation is starting/stopping it. **State** change notifies connected components if a simulation is started/stopped or finished. We further need the concept of **Voting** to define special events that require feedback from the event target (as explained above). By using this mechanism we plan to realize the rule-based interaction with processes. Furthermore the simulation will provide events regarding the progress of the simulation; in particular which tasks have already received forecast values.

```

<?xml version="1.0"?>
<topics
  xmlns="http://fp7-adventure.eu/ns/common/notifications-producer/1.0">
  <topic id=" state">
    <event>change</event>
    <vote>change</vote>
  </topic>
  <topic id="running/task">
    <event>partner_called</event>
    <event>partner_done</event>
    <event>partner_error</event>
  </topic>
</topics>

```

**XML<sup>10</sup>:** A list of all targets that currently receive notifications for a particular process model / simulation.

**XML<sup>11</sup>:** An enumeration of all active events as well as the target of a particular subscription.

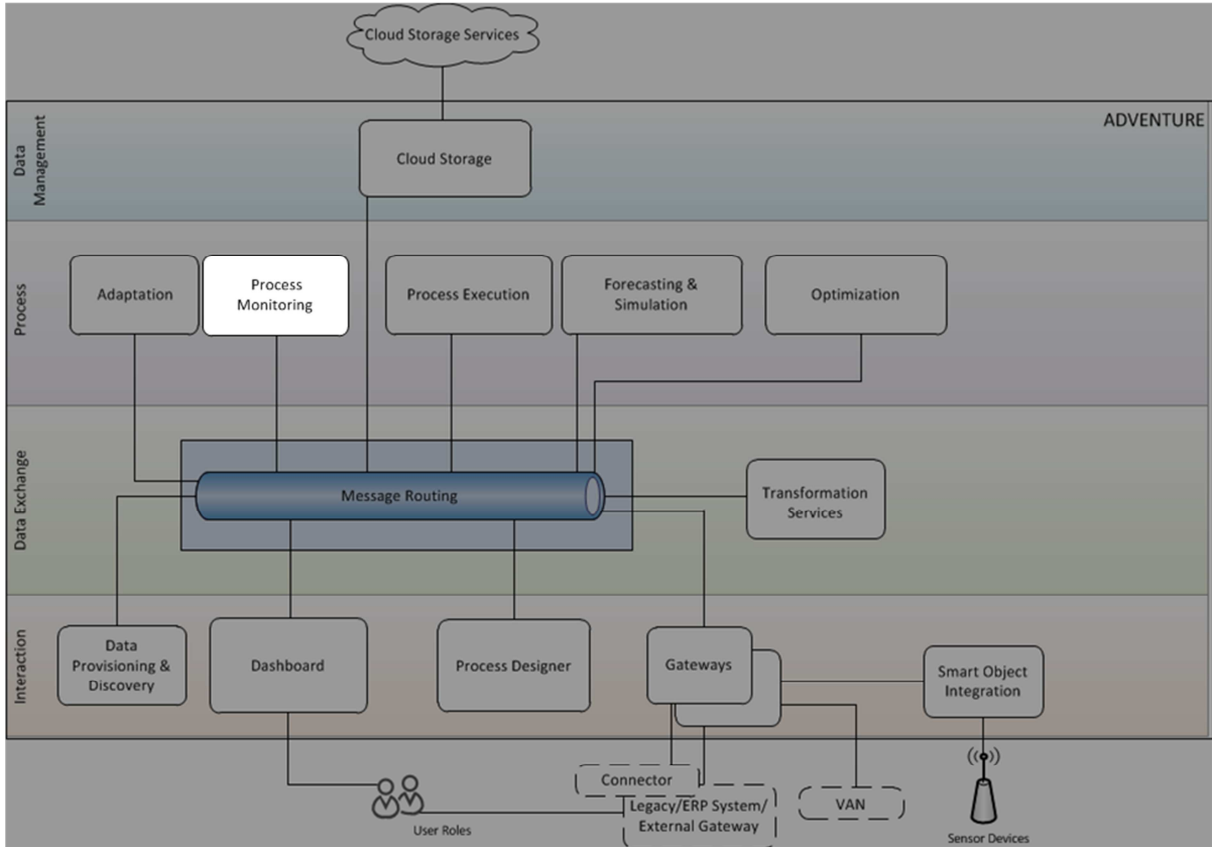
### 3.8.5 Summary

The Smart Process Execution Component will be based on the CPEE (Cloud Process Execution Engine). The CPEE covers many of the aspects needed by ADVENTURE, but needs to be extended in several areas such as correlation, XMPP-based interaction, translations for different process model formats, performance and several others. The Process Adaption component will be designed as an independent component, realizing algorithms developed by Rinderle-Ma et al. The Forecasting and Simulation Component will rely solely on the features of the Smart Process Execution to achieve its goals. Therefore, no separate technology selection has been performed for the later components.

The Smart Process Execution Component will expose strict, simple and powerful service interfaces and protocols that all extensions (which are not internal, but independent services) will have to use.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>310</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

# Process Monitoring



## 3.9 Process Monitoring

Abstract:

Process Monitoring captures the events subscribed at the Smart Process Execution component and stores all the relevant data in the cloud.

It provides a graphical UI, presenting the ongoing processes status, process log and process analytics.

It allows definition of rules, based on process constraints (e.g. due dates), throwing alerts to the Dashboard and Acting upon Smart Process Engine through the alarms and notifications engine.

### 3.9.1 Major Design Decisions

The Process Monitoring (PM) Component will be an independent component that relies on events provided by the Smart Process Execution (SPE) component and indirectly by the Dashboard, external systems and smart objects via the Gateways. Events triggered by the user on the dashboard will be captured and processed by the SPE and it will publish the events subscribed by the Process Monitoring component.

This way, the PM component can be directly applied to every process engine that uses the same publish/subscribe mechanism. In other words, it will be possible to simply switch the mechanism and integrate with other process execution engines.

The PM component will be as modular as possible, in order to make it flexible and useful to other projects and applications. It will be composed by the following subcomponents:

**Events Receiver** - This sub-component will be responsible for receiving the events published by the Smart Process Execution and storing them in the Cloud Storage. In this case, a XMPP publish subscribe mechanism will be applied, but it can be replaced in the future without any implication on the PM functionality.

All the data will be stored in a noSQL bucket but any other database system can be employed.

**Real Time Monitoring** – This sub-component will show the actual status of process instances in the Dashboard and will have its own User Interface (UI). It will use the same UI as Process Designer in order to have the same look and feel. This way, it will be easy for the user to identify and track the process instances. In other words, this sub-component will be a live graphical display of the process diagrams.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>312</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Process Log** – This sub-component will be an independent service that just queries finished process instances and show them to the user. It will act like a search engine for historical data.

**Process Analytics** – This sub-component will be an independent service that will rely on the data stored by the Events Receiver in the Cloud Storage. The purpose is to allow the user to configure Key Performance Indicators related to the manufacturing processes and then provide a graphical display in the dashboard in order to track KPIs.

**Rules Engine and Editor** – this rules editor will allow the user to define rules and associated actions. For that purpose, a graphical rules editor will be developed. This sub-component can work separately as well, so it can be integrated in any system to define/evaluate rules and throw events.

### 3.9.2 Technology Comparison and Selection

This section shows the technology selection criteria and evaluates existing technologies, candidates for the realization of PM component. Within the sub-section the areas that need to be implemented or improved to fully meet ADVENTURE specific requirements are also identified and presented. The selected technologies will be used as a base for the development phase in the realization of the technical design of the PM component.

#### 3.9.2.1 Selection Criteria

The selection criteria constitute generic and specific parameters. The generic properties are described in *D3.2 Functional Specification*. The specific properties identified in the functional specification that are to be taken into account for technology selection are listed and explained below.

*Table 17 – Specific parameters of technology selection criteria for Real Time Monitoring*

Name	Description
User Interface aspect / quality	Required to provide a good quality presentation of information results to the user.
Usability	The PM module should provide a user friendly interface and associated functionalities to (e.g, filter data, export data, compare information, introduce time filters, etc...)

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>313</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Customizable	The User interface should be totally customizable. Thus, the user can configure the Monitoring dashboard with the more important data with the aspects the user wants (graphical, table, etc..)
Web 2.0 Built-in graphical framework	It is well known that the application aspect is important for the end user. Thus, the graphical framework to be used should be of good quality. JavaScript libraries should be considered and investigated, in order to achieve the best look and feel for the application.
External data loading	Reasoning is a powerful feature in the systems that add value to the existing data, by providing implicitly encoded facts in it and thus further expanding the knowledge base to unanticipated limits. Since it also comes at a performance cost and the tradeoff is to be evaluated, it is put as a nice to have.

### 3.9.2.2 Possible Technologies

Most of the functionalities defined for the process monitoring such as real-time monitoring, process analytics, process log and notifications can be found in different software applications and domains. However, there is no specific and isolated technology that fully covers the PM requirements in the scope of ADVENTURE project. Business Process Management Suites (BPMS) usually have most of these functionalities, but they are not as customizable as ADVENTURE requires. Therefore, the focus in the state of the art study in this case is to find out what type of process monitoring functionalities were used by different relevant research projects in the area of business collaborations, in order to address certain aspects applicable for the PM domain model. Then a set of BPMS suites were evaluated in terms of the required functionalities.

The projects on which the study focuses are:

**Net-Challenge** – “Innovative networks of SMEs for complex products manufacturing”, is an EU research project in the area of collaborative business networks, supported by the 7th FP of the EC. Specific objectives include the design and development of an integrated framework, composed of a methodology, reference business processes and IT tools, to support SMEs to create and manage such networks. Net-Challenge features a collaboration portal (a customized implementation of the Open Source Liferay portal), and a portal plug-in for partner profile management that allows simple manual search operations, as well as template support. Web site: <http://www.netchallenge.org/>.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>314</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**VFF (Virtual Factories Framework)** – Virtual Factory Framework (VFF) is a Collaborative Research Project funded by the European Commission under the 7th Framework Programme. The project uttermost objective is to foster and strengthen the primacy of Future European Manufacturing, by defining the next generation Virtual Factory Framework. The VFF will promote major time and cost saving while increasing performance in the design, management, evaluation and reconfiguration of new or existing facilities, supporting the capability to simulate dynamic complex behavior over the whole life cycle of the Factory, approached as a complex long living product.

Apart from the above described projects relevant external classification systems were also explored:

**BizAGI** – is a software suite with two complementary products, a Process Modeler and a BPM Suite. Bizagi Process Modeler is a Freeware application with thousands of users worldwide for diagramming and documenting processes, using the Business Process Modeling Notation (BPMN) standard.

Bizagi BPM Suite is a Business Process Management (BPM) and Workflow solution that enables organizations to automate processes /workflows. Bizagi BPM Suite integrates with applications such as SAP, Documentum, SharePoint and e-mail. There is an entry-level edition (Xpress Edition) and two corporate editions (Enterprise .NET and Enterprise JEE).

Bizagi provides management indicators that are fully comprehensive and easy to interpret based on accurate, real-time business information, allowing process owners to make agile flow adjustments and better, more efficient decisions to optimize the performance of business processes. In order to display accurate information in all the options in this module, it is necessary to define not only the duration of the tasks, but also the process estimated duration in the processes' properties.

As a tool for continuous processes improvement, Bizagi provides a set of indicators that show the current status of ongoing cases and activities in the system, and the performance that the different processes have had. Entering the Analysis Reports menu from the Web application, Bizagi provides four indicator groups:

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>315</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- Process Analytics - These indicators present summaries of the cases and activities that are closed. The user can choose to filter by Process or any of the other criteria previously defined by the user regarding the process.
- Task Analytics - presents information of closed activities that belong to closed cases (not including aborted cases). Activities can be filtered by process version and User dimensions.
- Process Business Activity Monitoring (BAM) - BAM indicators provide information on the current status of ongoing cases. Process BAM is divided in two sections: Load Analysis and Work in Progress. The Process BAM shows the correct information only when the Process Duration is configured in the Properties of the process.
- Task BAM - Provides information on the current status of ongoing activities
- Sensor Analytics - Provides information of the phases (or paths) defined by the user.

**WSO2 Business Activity Monitor (BAM)** – is a tool designed to exercise Business Activity Monitoring. WSO2 BAM is intended to serve the needs of both business and IT domain experts to monitor and understand business activities within a SOA deployment. It is specifically designed for monitoring SOA deployments, and can be extended to cater for other general monitoring requirements as well. WSO2 BAM is an Open Source software product and is released under Apache Software License v2.0

**Bonita Open Solution Subscription Pack, Teamwork version (BonitaSoft)** - provides the process analytics functionalities requested by PM module in ADVENTURE. Key Performance Indicators (KPIs) in Bonita Open Solution refer to any data that may be collected on a process for BAM.

It is possible to identify the KPIs which are to be associated with a process, create them and add them to the appropriate tasks or at the process (pool) level. They will then capture specific data from each case of the process as it runs and insert it into an external database.

BAM reporting (in Bonita User Experience) will pull data from this same database and show it in reports. These reports are defined and installed in Bonita Studio.

*Table 18 – Technology selection criterion and comparison of different technologies for Real Time Monitoring*

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>316</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



Parameter	Importance (- - - +/- +++)	Net- Challeng e	VFF	WSO2	BIZAGI	BONITA SOFT
<b>Generic Parameters</b>						
Maturity & Stability	+++	+/-	++	+	++	++
Regularly Updated	+	-	-	+	+	++
Technical Up-to-Dateness / Appeal	+	+	-	+	+	+
Open Source	YES/NO	NO	YES	YES	NO	YES
Non-Infecting	YES	NO	YES	YES	YES	YES
Code-Quality	+	+	++	+/-	+	+
Extensibility	+	+	++	++	+	+/-
Community	+	-	+	+	+	+/-
EU project origin	+/-	++	++	-	-	-
Open Standards Compliance	+	+	++	++	-	-
Interoperability	++	+	+	+	-	-
<b>Specific Parameters</b>						
User Interface aspect / quality	+++	-	+	++	++	++
Usability	++	-	+	++	++	++
Modular	+	+/-	+/-	++	-	-
Web 2.0 Built-in graphic framework	++	-	+	+	-	-
User Interface aspect / quality	++	-	+	+	++	+

### 3.9.2.3 Conclusion

From the previous analysis, we conclude that either EU project or BPMS products contains functionalities that help in the conception of PM Module, however, they cannot be applied directly to PM Module, Thus, they will be considered as references for the development. Some of the technologies used in Net challenge project will be used (e.g. Liferay and vaadin) and the type of database used in WSO2 seems to be the best choice, due to scalability and extensibility, so it will be applied here. The BizAgi default KPIs and also process log search functionalities are very attractive and user friendly, so it will be a reference for these functionalities implementation.

None of the analyzed project results or products will be directly applied to the monitoring component, but in some cases they will be the reference for the component development.

### 3.9.2.4 Technology Selection

The PM Module will be based on the current Business Process Management Suites, but they will be extended in order to accomplish the ADVENTURE requirements, as there is no specific technology that can be directly applied/adapted to ADVENTURE. Each sub-component (Real-time monitoring, Process log, Process Analytics, Rules Engine) will be integrated in the Dashboard. Their UI shall comply with the Dashboard look-and-feel and will employ compatible client-side UI frameworks.

As Liferay uses the Vaadin technology and Google Web Toolkit (GWT) libraries, the PM will make use of it as far as possible as well.

**Vaadin** – As already mentioned above Vaadin is an Open source Web application framework for rich Internet applications. In contrast to JavaScript libraries and browser-plugin based solutions, it features a server-side architecture, which means that the majority of the logic runs on the servers. Ajax technology is used at the browser-side to ensure a rich and interactive user experience. On client-side Vaadin is built on top of and can be extended with GWT ([http://en.wikipedia.org/wiki/Google\\_Web\\_Toolkit](http://en.wikipedia.org/wiki/Google_Web_Toolkit)).

One of the most prominent features of Vaadin Framework is the ability to use Java (using a Java EE platform) as the programming language, while creating content for the Web. The framework incorporates an event driven programming and widgets, which enables a programming model that is closer to GUI software development, than traditional web development with HTML and JavaScript.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>318</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Vaadin Framework utilizes GWT for rendering the resulting Web page. While GWT operates only on client-side (i.e. a browser's JavaScript engine) – which could lead to trust issues – Vaadin adds server-side validation to all actions. This means that if the client data is tampered with, the server notices this and doesn't allow it.

Vaadin Framework's default component set can be extended with custom GWT widgets and themed with CSS.

From application developers' point of view, Vaadin is just one JAR-file that can be included in any kind of Java Web project developed with standard Java tools. In addition, there are Eclipse and NetBeans plugins for easing the development of Vaadin applications, as well as direct support of (and distribution through) Maven.

Vaadin applications can be deployed as Java Servlets to any Java server, including Google App Engine. Applications can also be deployed as Portlets to any Java portal, with deeper integration to **Liferay Portal**.

Regarding Data storage, a NoSQL bucket from Cloud Storage component will be employed. Modern web applications are built to scale out – simply add more commodity Web servers behind a load balancer to support more users. Scaling out is also a core tenet of the increasingly important cloud computing model, in which virtual machine instances can be easily added or removed to match demand.

In contrast, relational database (RDBMS) technology has not fundamentally changed in over 40 years, but remains the default choice for holding data behind many Web applications. Handling more users means adding a bigger server.

While implementations differ, NoSQL database management systems share a common set of characteristics:

No schema required – Data can be inserted in a NoSQL DB without first defining a rigid database schema. The format of the data being inserted can be changed at any time, without application disruption. This provides immense application flexibility, which ultimately delivers substantial business flexibility.

Auto-sharing – A NoSQL database automatically spreads data across servers, without requiring applications to participate.

Integrated caching – To reduce latency and increase sustained data throughput; advanced NoSQL database technologies transparently cache data in system memory. This behavior is transparent to the application developer and the operations

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>319</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

team, in contrast to RDBMS technology, where a caching tier is usually a separate infrastructure tier that must be developed to, deployed on separate servers and explicitly managed by the operations team.

### 3.9.3 Technical Component Specification

#### 3.9.3.1 Component structure

The Process Monitoring (PM) component will receive events from the Smart Process Execution by using a publish-subscribe mechanism. All relevant events such as step start, step exception, step deadline reached and so on will be captured by the Events Receiver sub-component and stored in the Cloud Storage, which also has a copy of the process instance structural data. This way the process monitoring knows which events and data corresponds to what process instance.

The Real-time Monitoring sub-component will then detect the changes and update the data on the real-time monitoring UI on the Dashboard component.

The Process Analytics allows users to define Key Performance Indicators related to a set of processes, process types or partners and show the analytical results in the Process Analytics Dashboard. This will rely on the data stored by the Events Receiver in the Cloud Storage.

Process Log is basically a search engine for finished process instances. This will rely on the data saved in the Cloud Storage too.

The Monitoring Rules Engine allows users to configure rules in order to trigger alarms and show them at the alerts page in the Dashboard. These rules will be evaluated by the Monitoring Rules Engine sub-component.

The component breakdown of the PM module is presented on Figure 78.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>320</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

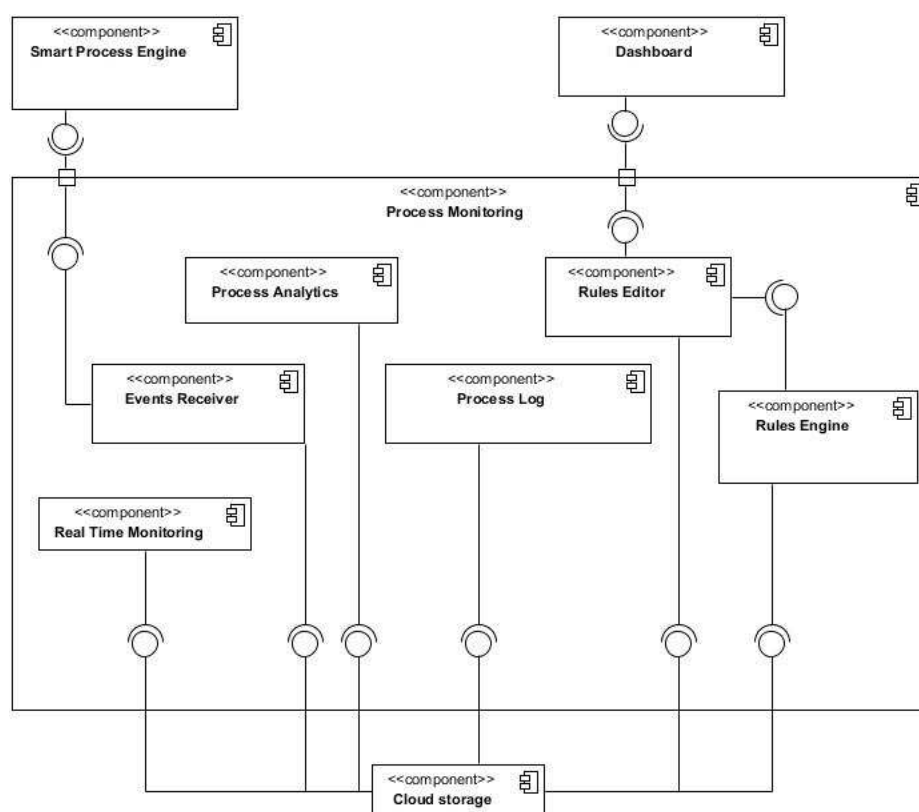


Figure 78 - PM component structure

## Events Receiver

The Events Receiver captures the events produced by the Smart Process Execution and stores the relevant event data in the Cloud Storage. The different events may be outlined as follows:

Events from the Dashboard and Process Designer will include all activities conducted by ADVENTURE brokers regarding the modeling (Process Designer) and optimization of processes. These events will be subscribed by monitoring and provided by the Smart Process Execution Engine.

Events from the Process Execution component will include all process execution related aspects. Processes describe the flow of invocation of all other components (including Gateways and Transformation services). Events generated by Smart Objects can trigger a process in the Process Execution component.

As such, events from other systems (e.g. ERP or Smart Objects) will be handled via the Execution Engine, which means that the Monitoring will be connected to the Execution Engine via Message Routing.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>321</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

So far, the following events from the Smart Process Execution component were identified that the Process Monitoring component will rely on:

Task is about to be executed

Gateway service called

Gateway service call finished

Task is about to finish

Task has failed

Data element was modified

Process model was changed

Process model has an error

Partner was changed

Invalid Partner

Process has been stopped / started / finished

Process is stopping / finishing (intermediate state)

Events should be named, versioned and typed by event source. Moreover, the event structure should consist on (name, value) duples of business data, metadata and correlation data.

PM needs a high performance, low latency API for receiving large volumes of business events. To do so, several technologies like Apache Thrift, REST, HTTP, or Web services were evaluated. The events storage will be highly scalable by using NoSQL database and columns families per event type. The use of Thrift, HTTP and Web services allows event publishing from any language or platform.

The Events Receiver will subscribe for events in the Smart process engine using a publish-subscribe mechanism. This way, when some event occurs, the SPE will send a message to the events receiver. An Authentication Service will be responsible for authenticating and authorizing the source of the events, before the events reach the receiver service. This service will check the credentials set by the data agent against the PM user base. It will be hosted on the https port of the Events Receiver server. There is no additional configuration needed to use this service.

PM will access the Cloud Storage data bucket, as it provides an API for all CRUD operations.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>322</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

DPD component will access the Monitoring Service in order to obtain updated data from partners that are participating in the process. PM will implement a Data Provider API to provide this information to the DPD.

Figure 79 shows the conceptual data model for the monitoring component, PM module will save a copy of all process instances data and will store all the data related with the process instances. This data will be then aggregated to show useful information to the user.

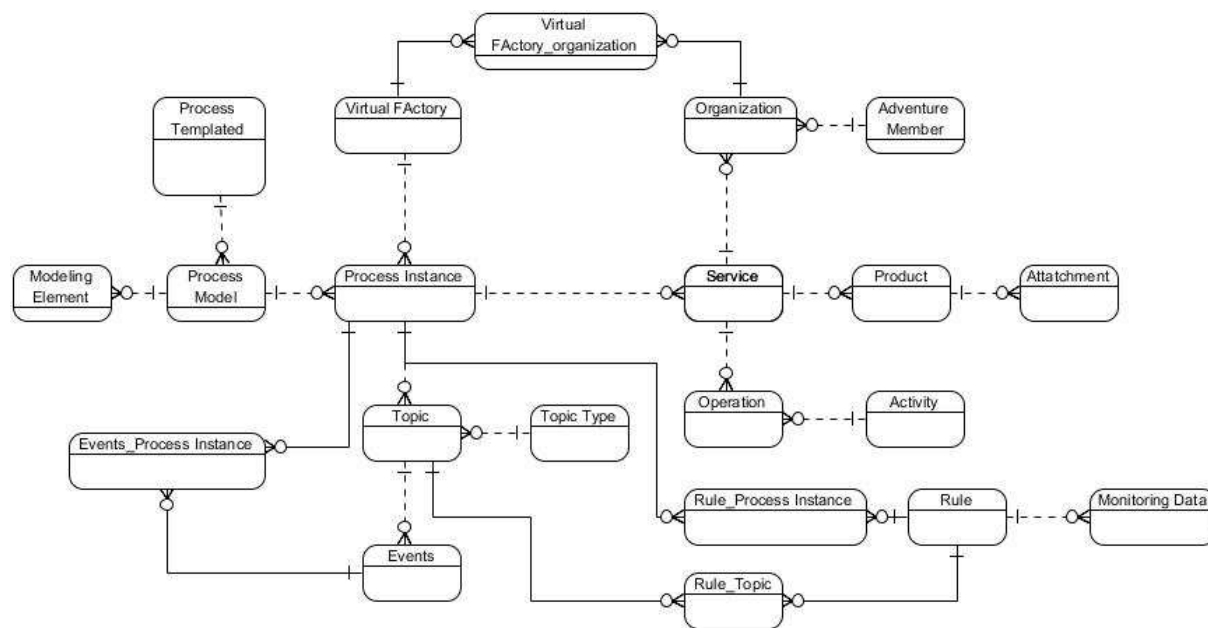


Figure 79 - Monitoring Conceptual Data Model

## Real Time Monitoring

The Real-time Monitoring component provides a live view of the ongoing processes, using the same interface as Process Editor Component, so that Virtual Factories brokers may decide to undertake flow adjustments and efficient decisions, in order to improve the performance of the manufacturing processes. Thus, the interface of the Process Model Editor must be extensible, in order to include on click events, change graphic elements properties (e.g. change the color of a given task depending on its status).

In order to provide a graphical representation of the process models, the Process Designer will save the graphical representation of the model into SVG format. In this way the Real-time Monitoring will be able to manipulate and enrich the graphical representation of the model, adding the necessary information to show all the real-

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>323</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

time monitoring information captured from the SPE. The Process Editor will also expose read only behavior mode, and calling interfaces, so that Monitoring can use the SVG option to represent information on top of a specific technology for visualization. Thus, monitoring can show the current execution status of a process model instance.

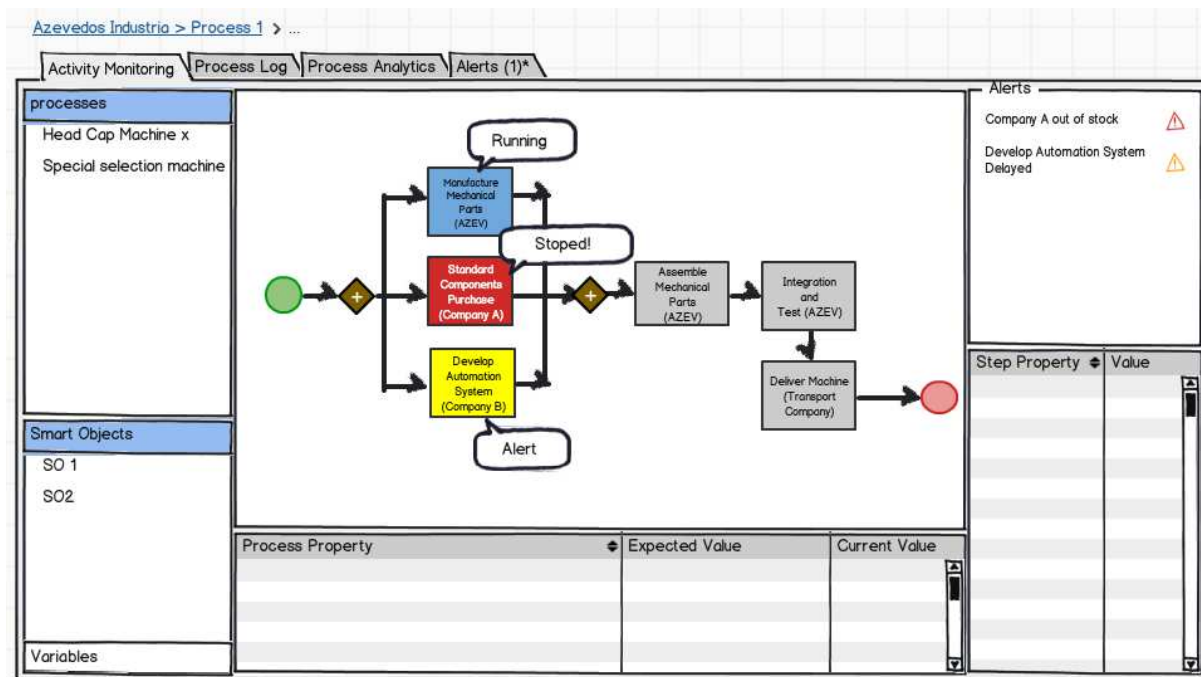


Figure 80 - Real Time Monitoring Mock-up

## Process Log

The Process log component allows users to search for finished process instances and visualize historical data in a graphical interface.

Serving the needs of both business and IT domain experts to monitor and understand business activities within SOA and Cloud deployments PM is not only designed to monitor SOA metrics, but can also be configured to monitor Key Performance Indicators through the process Analytics component.

## Process Analytics

Process Analytics component provides the Key Performance Indicators (KPIs) related to the manufacturing processes and ADVENTURE partners.

Most process analyses are based on the aggregation, correlation and evaluation of events that occur during the execution of a process. These events represent state

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>324</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



changes of objects within the context of a business process. These objects may be activities, actors, data elements, information systems, or entire processes, among others. For example, activities can begin and end, actors can log on and off, the values of data elements may change, and many other events can occur over the typical lifespan of a process instance. The scope of events considered for analysis determines the context envelope of the analysis. In other words, a narrowly scoped process analysis might focus on a single activity, by examining just those events that originated from this activity, its performers, and the resources that are input in and output from the activity. In contrast, a more widely scoped process analysis might include events from multiple processes; involve data sources outside the organization, and events from non-process-centric information systems.

The Process Analytics will send analytics events to an embedded Rules Engine that may trigger automated actions such as automatic notification of decision makers or automatic reprioritization of work. This configuration allows the automation of certain exception handling mechanisms, and results in the implementation of a simple sense-and-respond environment.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>325</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



Figure 81 - Process Analytics Mock-up

Businesses need to evaluate the success of their activities. The user interface allows users to manipulate data and create KPIs, which can be used for performance measurement. First, businesses must define specific tasks, since this allows them to manipulate data and carry out custom data operations.

Analyzer tasks are used to organize the flow of functions, It is possible to build KPIs such that they become an inherent part of Process Monitoring.. By building KPIs, you build business intelligence.” Once KPIs have been built, this data has to be extracted out of the database. In WSO2 BAM, the Open Source Apache Hadoop software framework pulls the data from NoSQL and supports high analyzer performance.

### Rules Engine and Editor

Rules Engine and Rules Editor component allows the definition of rules based on process execution delays (e.g. if a process has 5 days to execute and it is now day 6) that are evaluated by the Rules Engine. This component throws alerts to the Dashboard, as well as performs actions upon the Smart Process Engine. Process

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>326/ 372</b>
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

- Monitoring also has to include a rule engine that deals with creating notifications, alerts and messages that may:
- Trigger the creation of new process instances, e.g. to write a message that updates an ERP system.
  - Trigger the adaptation of a process, e.g. the automatic selection of a new Virtual Factory partner/service, to improve delivery time.
  - Stop a process and require action from an ADVENTURE broker.

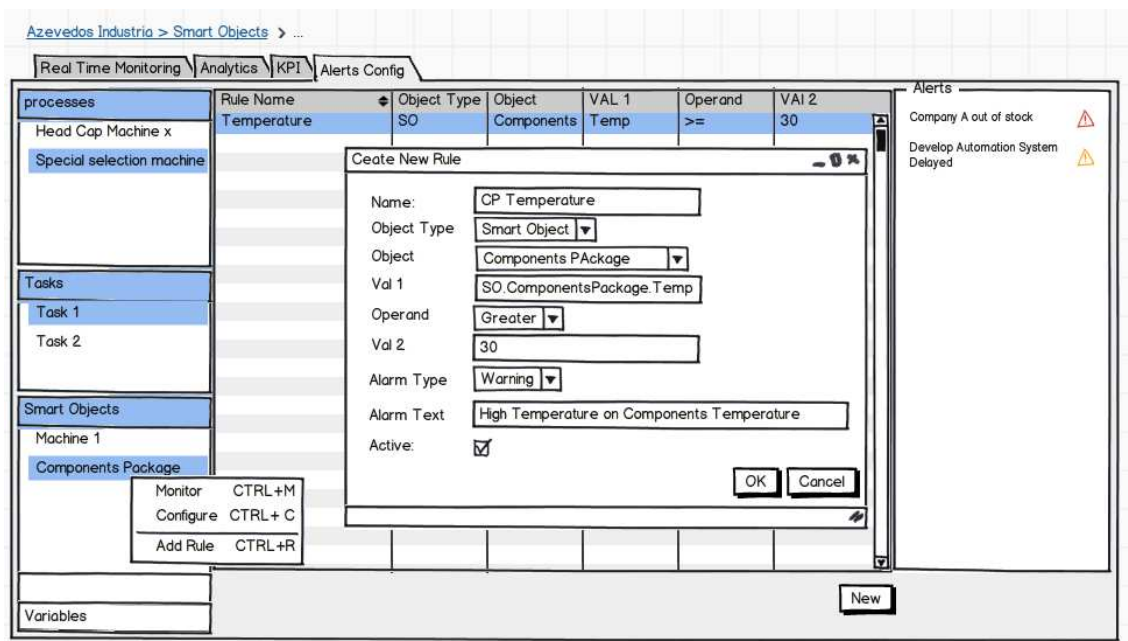


Figure 82 - Rules Engine MockUp

3.9.3.2 Internal Sequence Diagram

This section will give an insight of the internal component communication sequence based on an example scenario. The sequence diagram presented below shows an overview of the interactions between the sub-components. They do not show every single detail of this interaction, but the most important for ADVENTURE ones. Further, components that have secondary role in terms of the use case are not shown for better readability and focus.

## Receive and Update Order Status

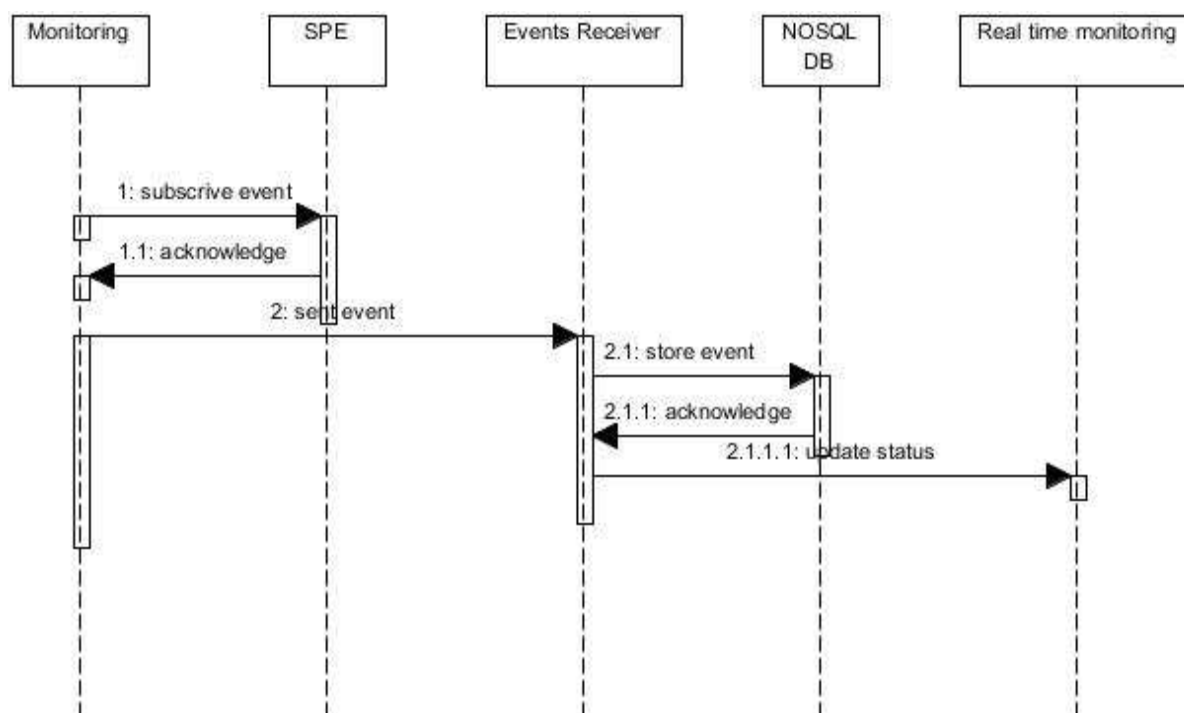


Figure 83 - Sequence diagram for update order status

### 3.9.4 Specification of Interfaces, Protocols and Formats

The connection of the Process Monitoring to the rest of components of ADVENTURE will be realized through the APIs exposed by the components. In the following sections these interfaces are explained.

#### 3.9.4.1 XMPP integration

The PM, as all other component will communicate through the XMPP protocol with other ADVENTURE components.

XMPP addresses (as explained in section 3.4) typically has the following form:

*component@factory/resource*

Thus it is possible to identify the sending or receiving instance for each message. Furthermore the *id* attribute of each XMPP message is reused as a mean to identify single activities in a process instance. The Message may either originate or be sent to these activities.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>328</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The PM module will use the XMPP publish-subscribe extension that provides a framework for a wide variety of applications. This technology uses the classic "publish-subscribe" or "observer" design pattern: a person or application publishes information, and an event notification (with or without payload) is broadcasted to all authorized subscribers. In general, the relationship between the publisher and subscriber is mediated by a service that receives publication requests, broadcasts event notifications to subscribers, and enables privileged entities to manage lists of people or applications that are authorized to publish or subscribe. The focal point for publication and subscription is a "node" to which publishers send data and from which subscribers receive event notifications. Nodes can also maintain a history of events and provide other services that supplement the pure publish subscribe model.

This is a generic protocol that all publish subscribe applications can use.

Although this technology specification is large because it defines many side use cases and possible error flows, the basic idea is simple:

An entity publishes information to a node at publish-subscribe service.

Published subscribe service pushes an event notification to all entities that are authorized to learn about the published information.

#### **3.9.4.2 Event Model**

To communicate its state to other components the ADVENTURE SPE can deliver a set of events. By default the PM component receives all events, but can subscribe to a subset of events as described above (SPE component section). The event model is presented in the SPE section.

The connection of the Process Monitoring to the rest of components of ADVENTURE will be realized through the APIs exposed at the component.

#### **3.9.4.3 API specification**

The PM component is aggregated into the Dashboard portal although it's will be available as a standalone application as well. The interaction protocol and API between these two components is defined by the Java Portlet specification and is realized by the Dashboard interface Portlet on the Process Monitoring side. As Process Designer will expose a JavaScript API, it can be used by Process Monitoring to show a read-only, enhanced with real-time data view of a process model instance. An API will be developed in order to provide information about partner analytics to the

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>329</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

DPD, so that the information about partner can also be enriched during instance executions.

#### 3.9.4.4 Content Formats and Protocol Definitions

##### BPMN 2.0

BPMN 2.0 is an OMG specification for designing and executing process models. It defines different conformance levels (Process Modeling Conformance, Process Execution Conformance, BPEL Process Execution Conformance, etc.).

The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

The vision of BPMN 2.0 is to have one single specification for a new Business Process Model and Notation that defines the notation, metamodel and interchange format. The final version of the specification was released in January, 2011.

The BPMN 2.0 XML specification is based on a set of XML schemas:

- BPMN20.xsd
- BPMNDI.xsd
- DC.xsd
- DI.xsd
- Semantic.xsd
- SVG (Scalable Vector Graphics)

Scalable Vector Graphics (SVG) is a family of specifications of an XML-based file format for two-dimensional vector graphics. It is an open standard that has been under development by the World Wide Web Consortium (W3C) since 1999.

SVG images and their behaviors are defined in XML text files. This means that they can be searched, indexed, scripted, and, if need be, compressed. As XML files, SVG images can be created and edited with any text editor. All major modern web browsers have at least some degree of support for SVG and can render markup directly, including Mozilla Firefox, Internet Explorer 9, Google Chrome, Opera, and Safari.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>330</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### 3.9.5 Summary

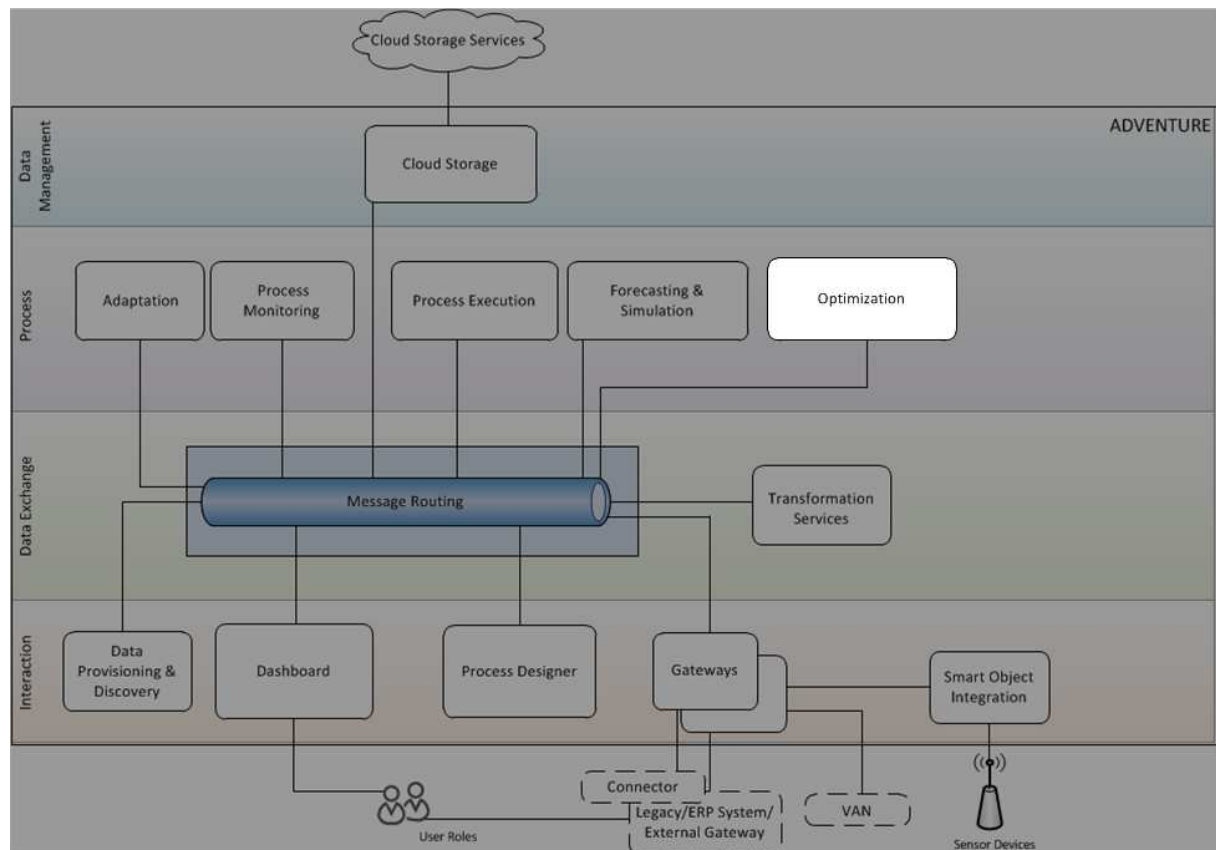
The Process Monitoring component will be developed after a state-of-the-art study in Business Process Activity Monitoring and Analytics and thus, it will be implemented considering all the functionalities included in the best BPMS systems. The component design will serve all requirements defined by ADVENTURE. However its goal is to provide a Process Monitoring Software that can be applied outside of Adventure.

The PM model will be designed, based on recent research and development in the field such of Business Activity Monitoring (during process instance execution) and Business Analytics (closed process instances), functionalities that are actually part of BPMS systems and are not provided as standalone products.

The software is also designed for cloud platforms primarily, but can also be adopted as a tightly integrated component in a system.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>331</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

# Process Optimization





## 3.10 Process Optimization

Abstract:

Specification of objectives and restrictions on non-functional service properties for single activities of Smart Processes as well as for overall Smart Processes which are to be satisfied by the optimization

Optimization of Smart Processes

Modeling of interdependencies among partner services which are to be considered for the optimization

### 3.10.1 Major Design Decisions

The Optimization component addresses the task of optimizing Smart Processes regarding non-functional aspects such as restrictions on cost, delivery time, carbon footprint, etc. (details are available in the functional specification D3.2) , i.e., aspects concerning the functionality or functional aspects in terms of “what is a service doing” as opposed to “how is a service doing it” are not considered.

Regarding the major design decisions, it has to be stated that the optimization component is actually designed as two loosely coupled building blocks. The first building block concerns the preparation and the specification of the actual problem which is to be optimized, whereas the second building block provides the optimized solution to the specified optimization problem. Regarding the first building block, the Optimization component will work on the process models (representing the Smart Processes which are to be optimized) as provided by the ADVENTURE Process Designer (cf. Section 3.7) – i.e., process models compliant with the BPMN 2.0 (cf. Section 3.7.1). In addition, for providing and processing input data regarding values for non-functional aspects, an XML-based data format will be used. Further details on interfaces, protocols and formats are provided in Section 3.10.4. For the second building block, a *Solver* providing standard solution algorithms is to be selected and applied. Further details in this respect are provided in the following Section 3.10.2

The actual task of the Optimization component – optimizing Smart Processes – will be split into three major subtasks, which are

Collect input data

Develop optimization problem

Solve optimization problem

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>333</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

First, data containing the structure of the Smart Process as well as appropriate services able to realize the activities of the Smart Process have to be collected, provided and prepared appropriately. This is basically the duty of the Model composer which is indicated in. The development of the optimization problem based on the collected data constitutes the second subtask, which is performed by the Converter (Figure 84). The third and last subtask for the Optimization component is solving the developed optimization problem. In addition to these major subtasks, a Graphical User Interface (GUI) serves a Broker as a means to initiate and perform the mentioned major subtasks. All the communication with other ADVENTURE components is realized via the Message Routing Interface.

Thus, these different, self-contained subtasks are designed to be realized by respective, self-contained subcomponents, which are briefly described in Section 3.10.1. It has to be noted that those subtasks must not be confused with the building blocks as mentioned before. Actually, the Model composer and the Converter are realized by the first building block, while the second building block provides a solver for solving the developed optimization problem. As for the second building block standardized technologies are required, whereas for the first building block, an implementation tailored to the particular needs of ADVENTURE is required, a comparison and selection of possible technologies is only performed for the second building block – for a solver – and provided in the following section.

### 3.10.2 Technology Comparison and Selection

This subsection will compare existing technologies for the second building block of the Optimization component. Those technologies will be used as a base for the development phase. In order to realize the Optimization component, standard solution algorithms should be applied in order to compute solutions to linear programs constituting linear optimization problems. Thus, it is necessary to find adequate technologies which are capable of solving linear programs. A linear program in the context of optimization constitutes an optimization problem where the objective (the goal which is to be pursued), as well as the constraints (requirements which are to be satisfied) comprise linear expressions regarding the decision variables which are to be computed by a linear programming solver. An example for a linear expression constituting a constraint is provided in the following. For instance, if two manufacturing partners, Partner A and Partner B, are possible candidates for

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>334</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

performing an activity A1 of a Smart Process, and also two manufacturing partners, Partner C and Partner D, may perform activity A2, there are four decision variables, namely  $x_A$ ,  $x_B$ ,  $x_C$ ,  $x_D$ . Each decision variable  $x$  may take the value “0” or “1”, where a value of “0” would indicate that this partner should not be selected and a value of “1” would indicate that this partner is selected. Further, it is assumed that there is a fixed budget limit “b” regarding the execution of activity A1 and A2. Partner A would occasion cost  $c_A$ , whereas Partner B occasions cost  $c_B$ . Respectively,  $c_C$  and  $c_D$  indicate the occasioned cost of Partner C and D. The requirement that the total cost must not exceed the restricted budget limit b constitutes a constraint on cost. The according, linear expression indicating this constraint on cost would be

$$c_A \times x_A + c_B \times x_B + c_C \times x_C + c_D \times x_D \leq b.$$

For the issue of solving linear programs, a lot of possible linear programming (LP) solvers exist and come into question. In the following, the selection criteria used to differentiate one linear programming solver from others, as well as possible technologies are described. Also the final technology selection and further implementation needs are elaborated.

### 3.10.2.1 Selection Criteria

For generic parameters, see the functional specification D3.2. For the Optimization component, the following parameters are important:

**Linear Programming Solver:** Ability to solve linear programs

**Efficiency:** Computation time a solver requires per problem size

**Scalability:** Support of the solver for increasing problem size

**Allow semantic annotation:** Possibility for enriching (services) descriptions with semantic annotations

**Process model descriptions based on standard:** Compliance of process model descriptions regarding established standards

The importance of the selection criteria is indicated in Table 19. While the criteria *Linear Programming Solver*, *Allow semantic annotation*, and *Process model descriptions based on standard* actually take “yes or no” values, the criteria *efficiency* and *scalability* can be higher or lower for different technologies.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>335</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

### 3.10.2.2 Possible Technologies

Software technologies able to solve linear programs, i.e., LP solvers, come into question. Basically, existing LP solvers differ in the way they implement the required standard solution algorithms. As such, the following selection will compare the two leading commercial and three promising charge-free solvers.

IBM ILOG CPLEX Optimizer<sup>20</sup> – is a commercial, high-performance mathematical programming solver for linear programming, including mixed integer linear programming, and for quadratic programming. Thus, CPLEX is additionally capable of also solving quadratic programs.

Gurobi Optimizer<sup>21</sup> – is a commercial, mathematical programming solver for linear programming, mixed-integer linear programming, and quadratic programming. Thus, analogously to CPLEX, also Gurobi is able to solve quadratic programs.

Ip\_solve<sup>22</sup> – is a free (LGPL licensed) mathematical programming solver for integer linear programming and mixed-integer linear programming. It is based on the revised simplex method and the Branch-and-bound method for the integers. Ip\_solve provides only functionality to solve pure linear, semi-continuous, and special ordered sets models, i.e., it cannot solve quadratic programs.

JOptimizer<sup>23</sup> – is a free (Apache 2.0 licensed) mathematical programming solver for integer linear programming, mixed-integer linear programming, and quadratic programming. Thus, analogously to CPLEX and Gurobi, JOptimiser is able to solve quadratic programs. It is implemented in pure java programming language.

Sat4j<sup>24</sup> – is a free (EPL and LGPL licensed) boolean satisfaction and optimization library for Java. It is able to solve integer linear programs, but only for binary variables and integer coefficients.

The two commercial solvers receive highest scores in recent benchmarks (e.g., <http://plato.asu.edu/ftp/milpc.html>). For Ip\_solve, the computation time for the tested example problems was 19 times higher. As JOptimizer and SAT4J are JAVA based,

<sup>20</sup> <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>

<sup>21</sup> <http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview>

<sup>22</sup> <http://lpsolve.sourceforge.net/5.5/>

<sup>23</sup> <http://www.joptimizer.com/index.html>

<sup>24</sup> <http://sat4j.org/>

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>336</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

their performance will be even worse compared to Ip\_solve, which is written in ANSI C. For instance, according to the SAT4J people in charge, “a SAT solver in Java is about 3.25 times slower than its counterpart in C++” (<http://sat4j.org/>). Although it can be argued that computation time is not the only critical factor or is not a critical factor at all, especially if the computation time is in the magnitude order of milliseconds, but it anyhow serves as an objective means for comparing different technologies.

The following table (Table 19) shows all potential technologies as well as their rating:

*Table 19 – Technology selection criterion and comparison of different technologies for Structured Data Storage*

Parameter	Importance (--- +/- +++)	CPLEX	Gurobi	Ipsolve	JOptimizer	SAT4J
<b>Generic Parameters</b>						
Maturity & Stability	++	+++	+++	+++	+++	+++
Regularly Updated	-	+++	+++	+	+++	+++
Technical Up-to-Dateness / Appeal	+++	+++	+++	++	++	++
Open Source	+	---	---	++	+++	++
Non-Infecting	+	NO	NO	YES	YES	YES
Code-Quality	-	+++	+++	+/-	+/-	+/-
Extensibility	-	---	---	+	+++	+++
Community	+	+/-	+/-	+++	-	+++
Performance	+++	+++	+++	++	+	+
Reuse of existing developments	++	+++	+++	+++	+++	+++
EU project origin	+	NO	NO	NO	NO	NO
Platform (Portability)	+	+++	++	++	-	-
Open Standards Compliance	+	+/-	+/-	+/-	+/-	+/-

Interoperability	++	+++	+++	+++	+++	+++
<b>Specific Parameters</b>						
Linear Programming Solver	+++	+++	+++	+++	+++	+
Efficiency	++	+++	+++	++	+	+
Scalability	+	+++	+++	+	-	-
Allow semantic annotation	++	---	---	---	---	---
Process model descriptions based on standard	+	---	---	---	---	---

Regarding the specific parameters “Allow semantic annotation” and “Process model descriptions based on standard”, it has to be stated that none of the examined solvers address these issues. Actually, they have to be addressed separately, i.e., appropriately implemented (Section 3.10.2.4).

### 3.10.2.3 Technology Selection

In this section, the decision for one of the technologies presented in the previous section is taken and justified.

In fact, CPLEX is very fast and scalable solver, which is able to also solve quadratic problems. It is a very popular solver in academia, but it is a commercial solver, which is only free for use in an academic environment. I.e., provided that the outcome of ADVENTURE will be used in a commercial environment, it has to be paid for its usage, which is less the nature of the problem, but it cannot be delivered under Apache License 2.0, which is aimed at in ADVENTURE.

The same argumentation applies to the Gurobi solver. It also provides technology for computing solutions to linear programs very fast. In fact, it outperforms CPLEX.

Referring to the benchmark provided above, lp\_solve is about 19 times slower than CPLEX and Gurobi. But it still is a full LP solver, which comes with LGPL license. Thus, it would fit and comply with the aim of putting the results of the ADVENTURE project under Apache License 2.0. In order to use lp\_solve, it is required to link to the lp\_solve libraries, which written in C and, thus, promise higher efficiency. They can

be called from almost any programming language. In fact, `lp_solve` is a very popular open source solver, which is often used in academia for solving linear programs.

`JOptimizer` is a promising new technology and is distributed under Apache License 2.0. In addition, it is a pure-java optimizer. Thus, in contrast to `lp_solve`, it becomes not necessary to link to external libraries. But, given the fact that its first public release was in 2012, `JOptimizer` is probably not mature enough to serve the needs of ADVENTURE. In addition, it is a pure-java optimizer, so that it becomes not necessary to link to external libraries.

`SAT4j` is also a java-based optimizer, implementing a binary solver. That is, the solver is restricted to assigning binary values for the decision variables. This is not necessarily a limitation in the context of ADVENTURE, as the solver will be used in order to decide, whether a service is selected or not. Thus, binary values for the decision variables are appropriate. `SAT4j` comes with EPL in addition to LGPL license. Thus, it would fit and comply with the targeted Apache License 2.0.

Regarding the aim of providing the ADVENTURE project results with a free license, the two regarded commercial solvers drop out. As `JOptimizer` is a very new solver and, thus, probably not mature enough, it is also discarded. Providing only a solver to binary decision values should admittedly be sufficient, but prevents ADVENTURE to be extended to more complex functionalities regarding the selection of partner services based on mixed-integer values. For instance, solutions where the values for the decision variables are between 0 and 1 indicating that the best possible solution lies in between, cannot be computed and addressed by `SAT4j`.

Thus, all things considered, `lp_solve` is selected to serve as technology for computing solutions to linear programs. However, the solver's functionality will only be loosely coupled to the implementation of the Optimization component (see the following section "Missing Elements and Implementation Needs"). Thus, it will be possible to easily exchange `lp_solve` for another LP solver during the course of the project and for potential further purposes.

### 3.10.2.4 Missing Elements and Implementation Needs

The technology and software tool which is to be selected for the Optimization component provides *only* the capability to solve linear programs. The actual challenge of the optimization component therefore is not computing a solution to a

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>339</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

problem but to *create* the problem. I.e., it is required to appropriately translate the purposes of the Optimization component into a mathematic optimization problem.

As it is the aim of the Optimization component to optimize Smart Process with regards to objectives and restrictions issued by an ADVENTURE Broker as, e.g., limitations in cost, delivery time, carbon footprint, etc., respective objectives and constraints have to be developed and formalized. For this, the Optimization component provides other ADVENTURE components as well as a Broker with means in terms of APIs and a GUI that allow for specifying objectives and restrictions on non-functional aspects of the whole Smart Process.

For instance, a Broker may require and, thus, provide the Optimisation component with the respective input data that the whole Smart Process must be executed within 3 days. Thus, the Broker specifies a constraint, i.e., a restriction, on “delivery date”, e.g., “delivery date < 3 days”. For providing such input, the Broker could either use the Process Designer or the Optimization component, which enables the Broker to review his input data and adapt it, if necessary. Furthermore, the Broker prefers, e.g., selecting rather cheap services to achieve minimal total cost. With this information and information regarding the structure of the Smart Process as well as respective candidate services for each activity of the Smart Process, which also may be provided by a Broker (or another ADVENTURE component), the Optimization component formalizes and implements objectives and constraints representing the provided restrictions and preferences. The objectives and constraints thereby strongly depend on the provided structure of the Smart Process. This is due to the fact that for implementing an objective or a constraint, the values of the non-functional aspects of the candidate services require to be aggregated over the whole Smart Process. Thus, if the structure of the Smart Process differs, this aggregation differs too. Thus, depending on the respective Smart Process, the objectives and constraints actually *look* different.

Also, interdependencies among the potential candidate services or special capabilities require being explicitly modelled and implemented in order to be accounted for during the optimization. Such special capabilities could for instance be the provisioning of manufacturing services which realize different and multiple activities of a Smart Process. An example for such interdependency might be that combining specific partner services A and B may result in reduced delivery time as partner A and B work very closely together. Or the opposite might be true that A's

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>340</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

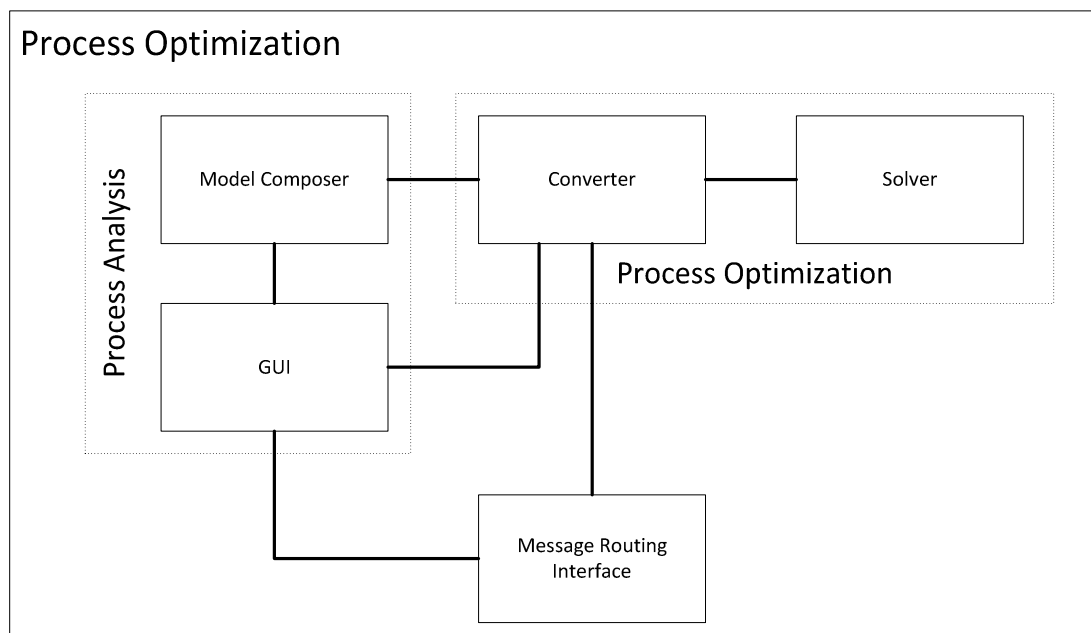


pre-products might be incompatible to B's requirements, so that additional steps are necessary to prepare appropriate input for B which increases the total delivery time. Such information about interdependency might either have been experienced during monitored, former interaction with partners A and B, or it might have been provided by A and B statically during the provisioning phase, where A and B registered their services using the ADVENTURE platform.

In addition to implementing the objectives and constraints according to the respective Smart Processes, the resulting optimization problem requires to be transformed into a *linear* optimization problem. Depending on the considered non-functional services aspects, the Optimization component firstly develops a non-linear optimization problem. If, for instance, the availability of the candidate partner services has to be accounted for, a multiplication of the respective availability values becomes necessary, which actually turns the optimization problem into a non-linear one. Thus, appropriate approaches have to be developed and implemented in order to transform non-linear optimization problems into linear ones which can be optimized by using standard linear programming solvers (reviewed in section 3.10.2).

### 3.10.3 Technical Component Specification

#### 3.10.3.1 Component Structure



*Figure 84 - Optimization Component Structure*

The GUI, which will be integrated into the Dashboard using a portlet, enables the Broker to provide the Optimization component with the structure of the Smart Process which is to be optimized. In addition, also the respective candidate services as well as the objectives and restrictions on non-functional aspects can be provided via the GUI. It has to be noted that all this information may equally be provided by the Process Designer, i.e., having specified a Smart Process including all its activities, the Broker uses the DPD to find and assign appropriate candidate services to each of the activities of the Smart Process. The Broker may also make use of the Process Designer to provide objectives and constraints on non-functional aspects, so that this entire data can be provided to the Optimization component. Alternatively the Broker may use the GUI of the Optimization component (which implicitly triggers the DPD) for these purposes, i.e., for assigning candidate services for each activity of the Smart Process as well as providing objectives and constraints on non-functional aspects, or for reviewing the input data the Optimization component received from the Process Designer.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>342</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The Model Composer validates the structure of the Smart Process and encodes it in an appropriate, recursive, tree-based format in order to account for different, recursively interlaced process models. The Model Composer provides the encoded process models to the Converter. The same applies for the provided candidate services, i.e., the Model Composer validates that at least one service is assigned as a candidate for each activity of the Smart Process. Further, the candidate services are also encoded in a special format appropriate for the Optimization component.

The Converter uses the encoded process model and candidate services to create and implement objectives and constraints. It therefore uses the objectives and restrictions which are provided by the Broker via the GUI. The Converter further converts the obtained optimization problem into a linear one if necessary and encodes it in a format such that the selected LP solver can solve it. Based on the results provided by the solver, the Converter develops an invocation plan, i.e. the assignment of services to the different activities of the Smart Process

The Solver takes as input a linear optimization problem and applies standard approaches to calculate a solution to it. The solver provides the Converter with this solution.

The Message Routing Interface enables the GUI and the Converter to send messages to the DPD in order to find services which are functionally appropriate to realize the activities of the considered Smart Process along with their values on non-functional aspects as provided by the DPD.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>343</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

3.10.3.2 Internal Sequence Diagram

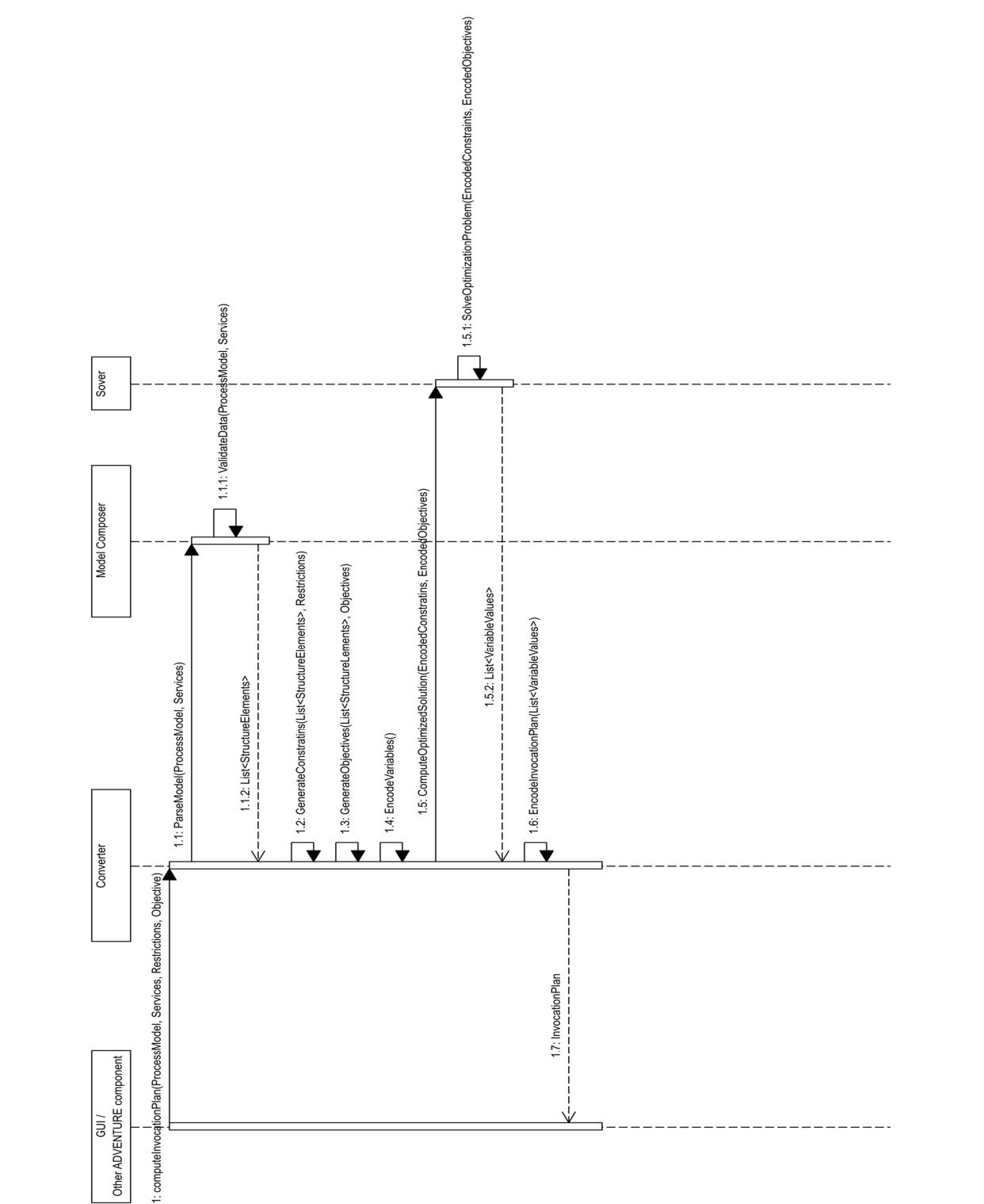


Figure 85 - Sequence Diagram for Computing an optimized Invocation Plan

In case, a Broker aims at computing an invocation plan, the respective method call “computeInvocationPlan(ProcessModel, Services, Restrictions, Objectives)” is performed. The actual model of a Smart Process as well as possible candidate services is thereby provided to the Converter. Also, respective restrictions and objectives regarding non-functional aspects are passed to the Converter. The Converter calls the method “ParseModel(ProcessModel, Services)” from the Model Composer in order to initiate the encoding of the Smart Process model and respective services.

The Model Composer validates, whether the received process model is a valid process model and whether for each activity at least one candidate service is assigned. If this is not the case, an error notification is provided to the Broker visualized via the GUI. If the process model is valid and each activity has at least one candidate service, the recursive tree-based representation is developed. For this, a list of structure elements (such as “AND-blocks” or “process activities”) retaining the created tree structure is created and passed back to the Converter. Using this list, i.e., the provided structure elements, the Converter generates objectives and constraints, based on the preferences and requirements provided by the Broker. Afterwards, the generated objectives and constraints are encoded in a format which is interpretable by the selected LP solver.

Finally, the Converter calls a solver method to trigger the computation of an optimized solution according to the encoded objectives and constraints. The solver optimizes the provided optimization problem (comprising objectives and constraints) and provides a list of variable values indicating the values of the used decision variables back to the Converter. Using these values, the Converter encodes an invocation plan, i.e., a process model along with the proposition of which services to select. It finally gives this plan back to the ADVENTURE component, which triggered the optimization, which is primarily the Process Designer, or to the GUI.

#### 3.10.4 Specification of Interfaces, Protocols and Formats

In order to communicate with other ADVENTURE components, the Message Routing Interface is used. As previously stated, the Message Routing Interface is used by the GUI and the Converter to send messages to the Data Provisioning & Discovery Component in order to find services which provide appropriate functionalities. On the other hand, other components may call the Messages Routing Interface to access

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>345</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

the Optimization component's functionality. The interface will be called using commands within the message. To clarify how the functionality of the Optimization component can be accessed, the API specification will be shown Section 3.10.4.1. Section 3.11.4.2 shows the communication protocol for the Optimization component.

#### 3.10.4.1 API specification

The Optimization component offers an ADVENTURE Broker and other ADVENTURE components the functionality of computing an optimized assignment of partner services to activities of a Smart Process. As the Optimization component provides an additional GUI enabling an ADVENTURE Broker to configure the optimization model, there is only one API method which can be used indirectly by other ADVENTURE components via the Message Routing Interface. This does not mean that a Broker necessarily requires using the GUI in order access the functionality of the Optimization component, but rather that he may use the GUI additionally to review and adapt the optimization problem. The full functionality is available by using the API of the Optimization component, which is described in the following.

- *computeInvocationPlan(ProcessModel, Services, Restrictions, Objectives)*

This method will compute an optimized invocation plan, i.e., an optimized assignment of partner services to the different activities of the Smart Process.

```
public String computeInvocationPlan (String ProcessModel,  
                                   String Services, String Restrictions,  
                                   String Objectives)
```

#### Parameters

**ProcessModel:** Process model in a XML-based structure describing the structure of the Smart Process model which is to be used for computing an optimized assignment of partner services to the activities of the Smart Process.

**Services:** Set of candidate services in a XML-based structure for each of the activities the Smart Process which is to be optimized.

**Restrictions:** Demands on non-functional aspects regarding the whole Smart Process in a XML-based structure describing lower and upper bounds for these non-functional aspects which are to be considered for the optimization.

Objectives: Preferences on non-functional aspects regarding the whole Smart Process in a XML-based structure describing the objectives on these non-functional aspects.

#### *Return Value*

Process model in a XML-based structure along with a concrete assignment of partner services to activities of the Smart Process.

#### 3.10.4.2 Content Formats and Protocol Definitions

For exchanging data between the Optimization component and other ADVENTURE components, XML documents will be used which are transferred via the Message Routing Interface. The XML-schema, which is shown in Figure 88,, contains all needed parameters for invoking the API method specified in the previous section.

The XML document contains a process model, which will be transformed internally into a “ProcessModel”, which contains “StructuredElements”. In the example request in Figure 86, this process model contains only one sequence comprising the activities and process steps, respectively, “P1” and “P2”. Also, the respective candidate services are indicated. Under “Services”, there are 4 services described, i.e., Service 11, Service 12, Service 21, and Service22. Also their values regarding “Cost” and “DeliveryTime” is indicated. In addition, a reference to the respective process step they are able to realize is provided. Regarding the “Restrictions”, there is the “Bound” that the “DeliveryTime” must not exceed “20” (indicated by the “BoundDirection”). In the “Objectives” section, it is indicated that there is only a single objective, i.e., minimize “Cost” (indicated by the ObjectiveDirection).

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:ComputeInvocationPlan xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/Optimization/Optimization.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/Optimization/Optimization.xsd Opt-request.xsd">
  <tns:ProcessModel>
    <tns:StructureElement>
      <tns:Sequence>
        <tns:ProcessStep>
          <tns:name>PS1</tns:name>
          <tns:number>1</tns:number>
        </tns:ProcessStep>
        <tns:ProcessStep>
          <tns:name>PS2</tns:name>
          <tns:number>2</tns:number>
        </tns:ProcessStep>
      </tns:Sequence>
    </tns:StructureElement>
  </tns:ProcessModel>
</tns:ComputeInvocationPlan>
```

```

</tns:StructureElement>
</tns:ProcessModel>
<tns:Services>
  <tns:ServiceName>Service11</tns:ServiceName>
  <tns:ServiceNumber>1</tns:ServiceNumber>
  <tns:ProcessStepNumber>1</tns:ProcessStepNumber>
  <tns:QoS>
    <tns:Cost>15</tns:Cost>
    <tns:DeliveryTime>5</tns:DeliveryTime>
  </tns:QoS>
</tns:Services>
<tns:Services>
  <tns:ServiceName>Service12</tns:ServiceName>
  <tns:ServiceNumber>2</tns:ServiceNumber>
  <tns:ProcessStepNumber>1</tns:ProcessStepNumber>
  <tns:QoS>
    <tns:Cost>10</tns:Cost>
    <tns:DeliveryTime>8</tns:DeliveryTime>
  </tns:QoS>
</tns:Services>
<tns:Services>
  <tns:ServiceName>Service21</tns:ServiceName>
  <tns:ServiceNumber>1</tns:ServiceNumber>
  <tns:ProcessStepNumber>2</tns:ProcessStepNumber>
  <tns:QoS>
    <tns:Cost>25</tns:Cost>
    <tns:DeliveryTime>15</tns:DeliveryTime>
  </tns:QoS>
</tns:Services>
<tns:Services>
  <tns:ServiceName>Service22</tns:ServiceName>
  <tns:ServiceNumber>2</tns:ServiceNumber>
  <tns:ProcessStepNumber>2</tns:ProcessStepNumber>
  <tns:QoS>
    <tns:Cost>30</tns:Cost>
    <tns:DeliveryTime>13</tns:DeliveryTime>
  </tns:QoS>
</tns:Services>
<tns:Restrictions>
  <tns:QoS>DeliveryTime</tns:QoS>
  <tns:BoundDirection>LowerOrEqual</tns:BoundDirection>
  <tns:Bound>20.0</tns:Bound>
</tns:Restrictions>
<tns:Objectives>
  <tns:ObjectiveDirection>Minimize</tns:ObjectiveDirection>
  <tns:ObjectiveType>
    <tns:SingleObjective>
      <tns:QoS>Cost</tns:QoS>
    </tns:SingleObjective>
  </tns:ObjectiveType>
</tns:Objectives>
</tns:ComputeInvocationPlan>

```

*Figure 86 – ComputeInvocationPlan Example Request*

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>348</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



The corresponding xml-schema is visualized in Figure 87 and Figure 88. As can be seen from the schema, it is also allowed to use and combine further orchestration components, such as “AND” or “XOR”, and to use further non-functional service aspects. In addition, it is also possible to specify multiple objectives instead of only one objective, which is not shown in the example request in Figure 86.

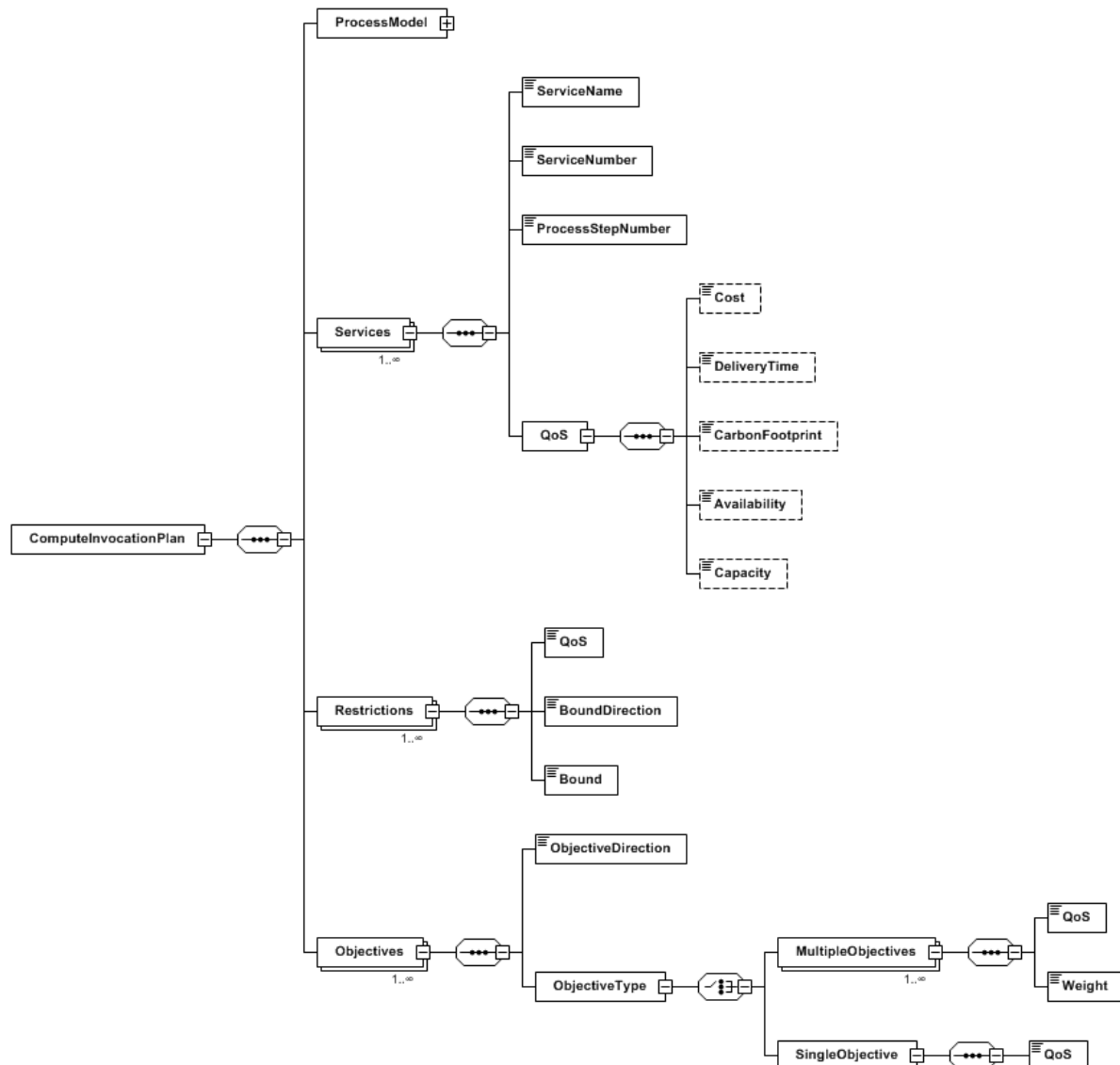


Figure 87 - ComputeInvocationPlan Schema (part 1)

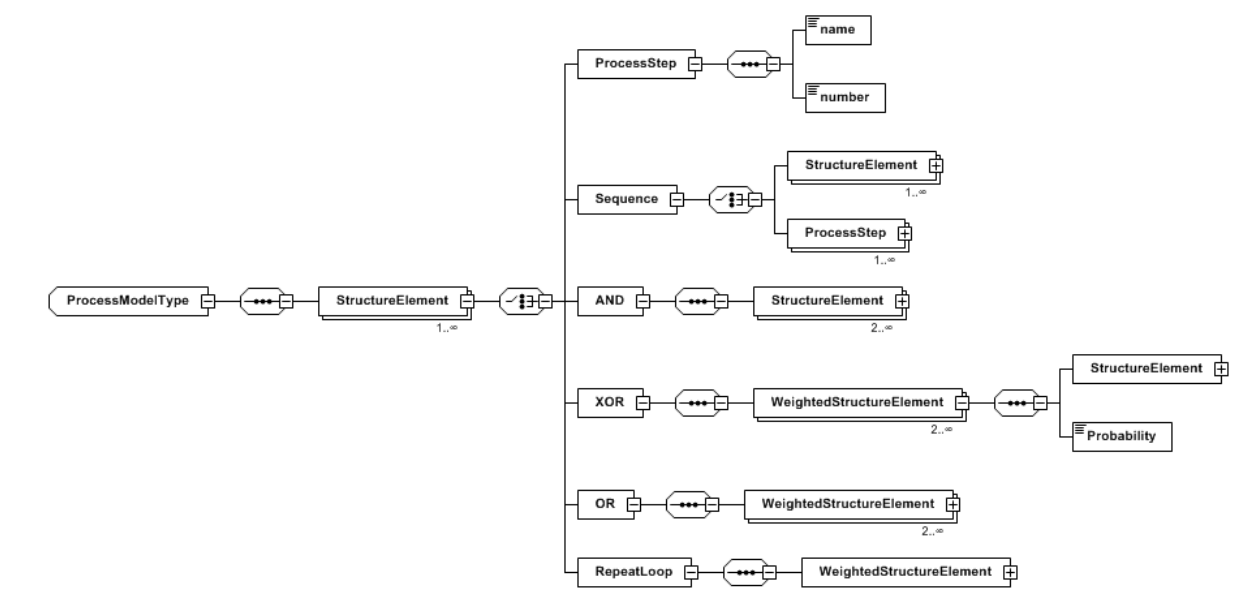


Figure 88 - ComputeInvocationPlan Schema (part 2)

A response to the example request in Figure 86 indicates an invocation plan, i.e., an assignment of services to the activities and process steps, respectively, of the Smart Process. An example response for the example request is shown in Figure 89, whereas the respective xml-schema is indicated in Figure 90.

```

<?xml version="1.0" encoding="UTF-8"?>
<tns:InvocationPlan xmlns:tns="http://www.fp7-
adventure.eu/xmlSchema/Optimization/Optimization.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fp7-
adventure.eu/xmlSchema/Optimization/Optimization.xsd Opt-response.xsd">
  <tns:Assignment>
    <tns:ProcessStep>
      <tns:name>PS1</tns:name>
      <tns:number>1</tns:number>
    </tns:ProcessStep>
    <tns:Services>
      <tns:ServiceName>Service11</tns:ServiceName>
      <tns:ServiceNumber>1</tns:ServiceNumber>
      <tns:ProcessStepNumber>1</tns:ProcessStepNumber>
    </tns:Services>
  </tns:Assignment>
  <tns:Assignment>
    <tns:ProcessStep>
      <tns:name>PS2</tns:name>
      <tns:number>2</tns:number>
    </tns:ProcessStep>
    <tns:Services>
      <tns:ServiceName>Service21</tns:ServiceName>
      <tns:ServiceNumber>1</tns:ServiceNumber>
      <tns:ProcessStepNumber>2</tns:ProcessStepNumber>
    </tns:Services>
  </tns:Assignment>
</tns:InvocationPlan>

```

Figure 89 - InvocationPlan Example Response

As indicated in the example response in Figure 89, Service11 is proposed for the first activity PS1 and Service21 is proposed to use for the second activity PS2.

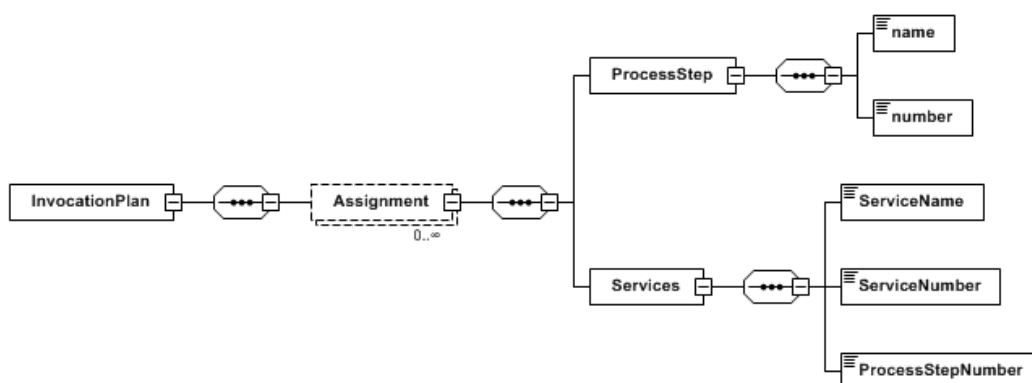


Figure 90 - InvocationPlan Schema

As can be seen from the schema in Figure 90, it could also happen that no assignment of services to activities of the process model is conducted. This can happen if the optimization problem is infeasible, i.e., if no solution satisfying the provided constraints was found.

### 3.10.5 Summary

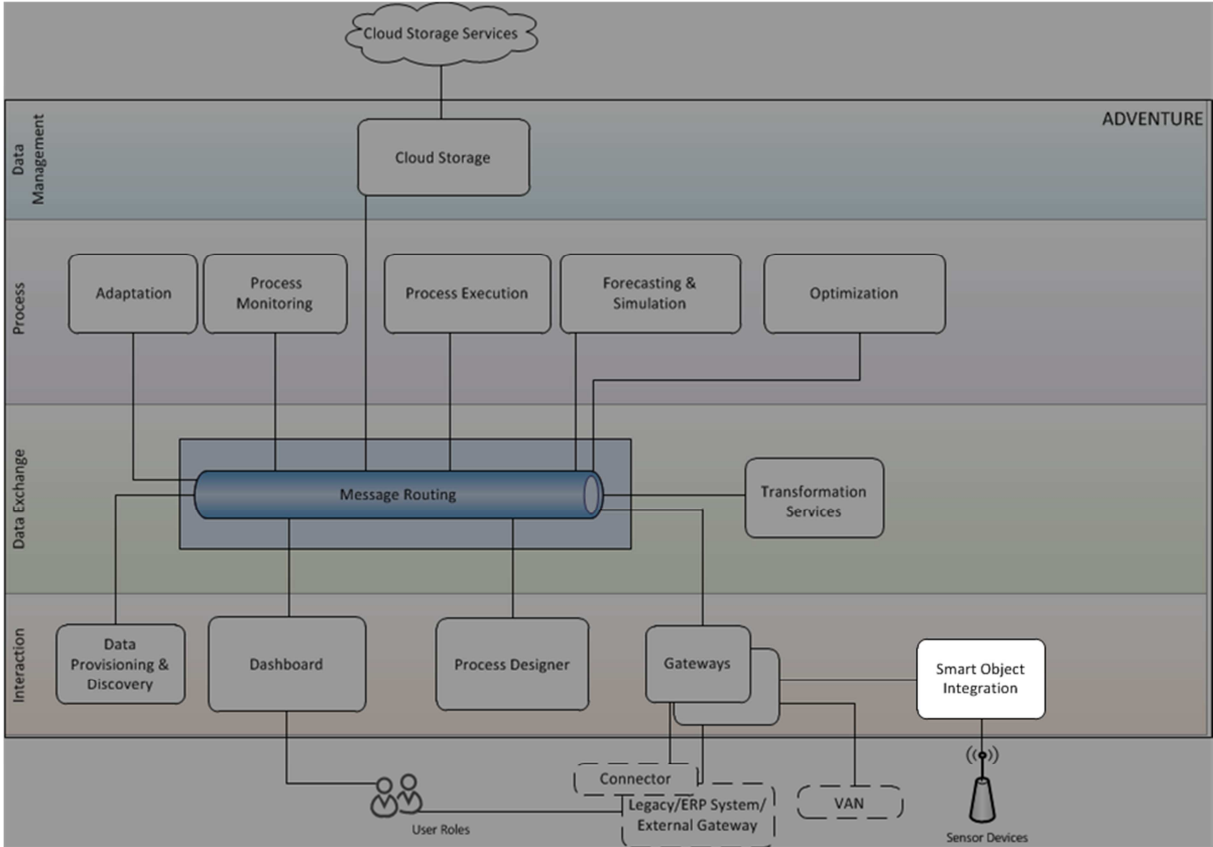
The Optimization component addresses the task of optimizing Smart Processes regarding non-functional aspects as, e.g., restrictions on cost, delivery time, carbon footprint, etc. For this, it receives input from the Process Designer regarding the (BPMN compliant) process model, which is to be optimized along with candidate service for each activity of this model. In addition, input regarding objectives and constraints is also provided by the Process Designer. Anyhow, a Broker has the possibility to use the GUI of the Optimization component, which will be integrated into the Dashboard using a portlet, in order to review and adapt the received input data or to specify and provide this data directly.

Making use of this input data, the Optimization component develops an optimization model aiming at proposing an optimized assignment of partner services to each activity of the Smart Process (see also D3.2 Functional Specification). For this, the Optimization component makes use of a standard LP solver – lp\_solve.

Analogously to all other components the communication will be realized via the Message Routing component using defined message formats.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>352</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

# Smart Object Integration



## 3.11 Smart Object Integration

Abstract:

Smart Object Integration provides different access possibilities to Smart Objects

It supports diverse Smart Objects based on different hardware

It allows Smart Objects to push data to ADVENTURE and ADVENTURE to pull data from Smart Objects.

### 3.11.1 Major Design Decisions

The Smart Object Integration has to provide bi-directional data channels between Smart Objects and the ADVENTURE system through Gateways. Thus, it has to ensure communication to and from heterogeneous Smart Objects in different application scenarios.

The Smart Object Integration will support two different communication paradigms:

Event-based push communication: Smart Objects initiate a data transmission to the ADVENTURE system, because a certain condition has been fulfilled and thus an event detected. Thus, data has to be pushed to the ADVENTURE system because of the occurrence of an event.

Query-based pull communication: The ADVENTURE system initiates a data transmission from a Smart Object, because currently context data from a Smart Object is required in the ADVENTURE system. Thus, data is pulled from a Smart Object on the basis of a specific query.

The Smart Object Integration will support different communication technologies, in particular:

The IEEE 802.15.4 standard: A communication standard particularly designed for low-rate wireless personal area networks and currently one of the prevalent standards in the field of wireless sensor network technology. It specifies the lower two layers of the OSI Reference Model (Physical Layer and Data Link Layer) and focuses in particular on energy-efficient operation. It provides the technological basis for the ZigBee standard, as ZigBee is built upon the communication layers defined in IEEE 802.15.4.

Bluetooth: A wireless communication standard for short distance communication for personal area networks and currently one of the prevalent standards for interconnecting different peripheral devices for PC, laptops, and smartphones. It

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>354</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

usually comes as built-in technology of current smartphone generations and already some wireless sensor nodes possess built-in Bluetooth capabilities. Further, extension kits to provide wireless sensor nodes with Bluetooth capabilities exist.

### 3.11.2 Technology Comparison and Selection

This subsection will compare existing technologies for the Smart Object Integration component. Those technologies will be used as a base for the development phase.

#### 3.11.2.1 Selection Criteria

For generic parameters, see the functional specification. For the Smart Object Integration the following parameters constitute additional parameters in particular important:

**Energy efficiency:** Wireless sensor nodes as building blocks of wireless sensor networks and enabling technology for Smart Objects usually use batteries as power supply. Thus, their energy budget is restricted and therefore energy efficient operation is mandatory.

**Cost efficiency:** As in particular SMEs have to face a huge cost pressure in the market, the integration of Smart Objects has to be realized in a cost-efficient way.

**Wireless standard support:** To enable interoperability and provide for a sufficient ease of operation relevant wireless standards should be supported by the Smart Object Integration solution.

**Legal usage in all EU states:** As wireless communication will play a major role within the context of data exchange with Smart Objects using the Smart Object Integration and several different regulations exist concerning the usage of transmission bands for wireless communication this has to be specifically taken into account to provide a solution which is compliant to different regulations in this context in different parts of the continent.

**Extensibility:** It is envisioned that Smart Objects can and will be added to a deployment of different Smart Objects during its lifetime. Thus, the Smart Object Integration must provide the corresponding extensibility to incorporate newly added Smart Objects.

**Ease of use:** To allow for a wide user acceptance and reduce the error-proneness due to operating errors of users, a sufficient ease of use has to be provided by the Smart Object Integration.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>355</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

**Support for variety of Smart Objects:** Furthermore, it is expected, that heterogeneous Smart Objects will be used within ADVENTURE and as already stated, it is envisioned that Smart Objects can and will be added to a Smart Object deployment during its lifetime. These additional Smart Objects are not necessarily of the same type as the Smart Objects already existing within the deployment, but might be new Smart Objects with new characteristics and thus introduce a new level of heterogeneity. Therefore, a sufficient support for a certain variety of Smart Objects has to be provided.

### 3.11.2.2 Possible Technologies

To realize the Smart Object Integration component it is needed to find adequate technologies to communicate with Smart Objects on the one side and with the ADVENTURE system through specific Gateways on the other side. This means that sufficient communication capabilities and simultaneously sufficient computing and memory capabilities to transform data between the different recipients is required. In this context, smartphones provide a promising opportunity as technological basis for the Smart Object Integration component. As currently prevailing technology platforms in this domain, we will compare Android and Apple's iOS in combination with the above mentioned communication technologies of IEEE 802.15.4 and Bluetooth.

The following table shows the potential technologies as well as their rating:

*Table 20 - Technology selection criteria for Smart Object Integration*

Parameter	Importance (- - - +/- +++)	Android + IEEE 802.15.4	Android + Bluetooth	iOS + IEEE 802.15.4	iOS + Bluetooth
<b>Generic Parameters</b>					
Maturity & Stability	+	++	+++	++	+++
Regularly Updated	-	++	+++	+	++
Technical Up-to-Dateness / Appeal	++	+++	+	+++	+
Open Source	-	Yes	Yes	No	No
Non-Infecting	-	++	++	+++	+++



Code-Quality	+/-	N/A	N/A	N/A	N/A
Extensibility	+/-	++	++	+/-	+/-
Community	+	+++	++	++	+
Performance	++	+++	++	+++	++
Reuse of existing developments	+/-	++	++	+	+
EU project origin	-	No	No	No	No
Platform (Portability)	++	+++	+++	+	+
Open Standards Compliance	+/-	+++	-	++	--
Interoperability	++	++	+++	+/-	+
<b>Specific Parameters</b>					
Energy efficiency	+++	++	+/-	++	+/-
Cost efficiency	+	++	+++	+	++
Wireless standard support	+	Yes	Yes	Yes	Yes
Legal usage in all EU states	+++	Yes	Yes	Yes	Yes
Extensibility	++	++	++	+/-	+/-
Ease of use	+	++	+++	+	++
Support for variety of Smart Objects	+/-	+++	+++	+++	+++

### 3.11.2.3 Technology Selection

As technology platform for the Smart Object Integration component, we will be using Android, because it provides easy to use software development facilities, a strong community support and enables the usage of a variety of hardware, which allows to use cheaper hardware with positive effects on cost efficiency and furthermore eases extensibility and portability, even e.g., to different of the new Android Pads.

For the underlying communication technology, we will use Bluetooth in the first iteration, because it is inherently supported by smartphones and thus allows to easily

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>357</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

reaching first results as proof-of-concepts and provides the ability to feed the whole ADVENTURE system with required data from Smart Objects from an early stage of the project without the need to extend the smartphone. Nevertheless, due to several advantages of the IEEE 802.15.4 standard, e.g., the wide dissemination in the wireless sensor network domain with the consequently huge and active community and the ability to support a significant variety of wireless sensor nodes and reuse existing developments, and IEEE 802.15.4's better energy efficiency compared to Bluetooth, in parallel solutions using IEEE 802.15.4 will be investigated so that in the course of the project IEEE 802.15.4-enabled solutions will be used as well.

### 3.11.2.4 Missing Elements and Implementation Needs

The selected technology only provides the basic technology platform, only Bluetooth is usually ready to use. Thus, besides enabling the usage of IEEE 802.15.4 on the smartphone the whole logic and communication requirements have to be developed and implemented on smartphone and Smart Object side.

A communication protocol for communicating with different Smart Objects.

A communication protocol for communicating with the Gateways to the ADVENTURE system.

Capability for data adaptation to transform data between the formats required from a specific Smart Object, respectively from a specific Gateway.

Connection possibilities for hardware extensions to support the different underlying communication technologies.

### 3.11.3 Technical Component Specification

#### 3.11.3.1 Component Structure



*Figure 91 - Smart Object Integration component structure*

### Gateway Interface

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>358</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

The Gateway Interface provides the communication interface to a gateway through which the communication with the ADVENTURE system is realized. It will thus provide the ability to push data, e.g., in form of a JSON object, received from Smart Objects to a gateway and vice versa receive data from the ADVENTURE system, like a query, through a gateway.

### Integration Engine

The Integration Engine comprises the main logic of the Smart Object Integration component. It will adapt the data received from a Smart Object to a format fitting to the receiving gateway's requirements. Vice versa, it will take care that data received from the ADVENTURE system through a corresponding gateway will be adapted to the format required by the receiving Smart Object and direct the data to the appropriate Smart Object.

### Connector

The Connector will realize the communication with a Smart Object. Thus, it must be able to (physically) transmit and receive data to and from the intended Smart Object based on the specifically required communication standard.

#### 3.11.3.2 Internal Sequence Diagram

This section will give an insight about the internal component communication sequence based on sequence diagrams building upon the use cases specified in D3.2.

#### Receiving and forwarding a query message

As described in *Figure 92*, a query message issued by the ADVENTURE system is transmitted through an appropriate gateway to the Smart Object Integration component, where it is received by the Gateway Interface. The Gateway Interface forwards the query message to the Integration Engine. The Integration Engine interprets the query message and adapts it to the required target format expected by the addressed Smart Object. Afterwards, the Integration Engine passes the adapted message to the Connector which then transmits the message to the addressed Smart Object.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>359</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

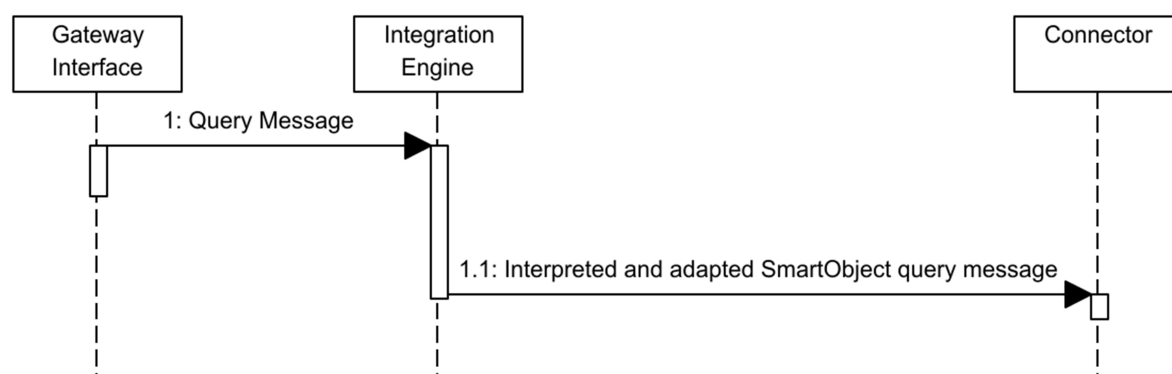


Figure 92 - Sequence diagram for receiving and forwarding a query message

### Forwarding a Smart Object message

Figure 93 describes the interaction sequence for forwarding a message from a Smart Object to the ADVENTURE system through a corresponding Gateway. The message is received by the Connector and forwarded to the Integration Engine. The Integration Engine interprets the message and adapts it to the required target format expected by the addressed Gateway. Afterwards, the Integration Engine passes the adapted message to the Gateway Interface which then transmits the message to the addressed Gateway.

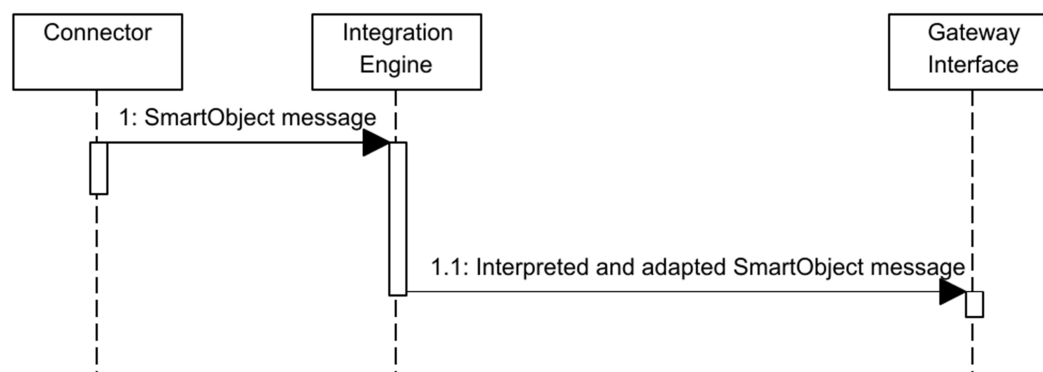


Figure 93 -Sequence diagram for forwarding a Smart Object message

#### 3.11.4 Specification of Interfaces, Protocols and Formats

The communication between Smart Objects and the ADVENTURE system will be realized with the two paradigms of push-based communication and pull-based communication as already highlighted in Deliverable D.3.2. Thus, events detected by Smart Objects will be pushed to the ADVENTURE system via the Smart Object

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>360</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Integration and through the appropriate Gateways (push communication) and queries can be issued from the ADVENTURE system to Smart Objects asking for current context data through the appropriate Gateways and via the Smart Object Integration (pull communication). Consequently, we want to employ Web Service technologies, e.g., to realize the querying of a Smart Object. In the domain of Smart Objects, the described communication possibilities are still part of ongoing research. Thus, the following must be understood as initial communication concept which will be refined based on the research results gathered during ADVENTURE.

#### 3.11.4.1 API specification

The Smart Object Integration component offers ADVENTURE users and other ADVENTURE components the functionality to request (context) data from Smart Objects by using the Smart Object Query.

Smart Object Query

*Parameters*

requestedData: Which (context) data is asked for (e.g., current temperature or acceleration)?

requestedDataFormat: In which format shall the data provided (this does not only include “technical” low-level data formats, but as well for example the differentiation between units of measurement or a request concerning aggregated data)?

*Return Value*

The requested data in the requested format

#### 3.11.4.2 Content Formats and Protocol Definitions

It is envisioned that the communication with the Smart Objects is JSON-based. A JSON-based communication provides the advantages of compactness and easy serialization and deserialization, as well as the possible usage of ultra-lightweight libraries. This is in particular important against the background that the employed hardware of wireless sensors and smartphones possesses significant resource constraints and thus appropriate ways of communication are required. In the following, we present two initial JSON-schemata for querying context parameters from a Smart Object and receiving context parameters from a Smart Object.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>361</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Smart Object Query

```
{
  "description": "This message is used to query an environmental parameter from a sensor node. It triggers a parameter response message.",
  "type": "object",
  "properties": {
    "header": {
      "description": "The common header for all messages",
      "type": "object",
      "properties": {
        "messageType": { "type": "integer" },
        "messageId": { "type": "integer" },
        "srcAddress": { "type": "integer" },
        "dstAddress": { "type": "integer" },
        "srcType": { "type": "integer" },
        "dstType": { "type": "integer" }
      }
    },
    "parameter": {
      "description": "The identifier for the environmental parameter that is to be queried from a sensor node.",
      "type": "integer"
    }
  }
}
```

Smart Object Context Parameter Response

```
{
  "description": "This message is sent by a sensor node to report an environmental parameter value to the user.",
  "type": "object",
  "properties": {
    "header": {
      "description": "The common header for all messages",
      "type": "object",
      "properties": {
        "messageType": { "type": "integer" },
        "messageId": { "type": "integer" },
        "srcAddress": { "type": "integer" },
        "dstAddress": { "type": "integer" },
        "srcType": { "type": "integer" },
        "dstType": { "type": "integer" }
      }
    },
    "parameter": {
      "description": "The identifier for the environmental parameter that is to be reported.",
      "type": "integer"
    },
    "value": {
      "description": "The value of the environmental parameter that was determined by the sensor node.",
      "type": "number"
    }
  }
}
```

```
    },  
    "timestamp":{  
        "description":"The timestamp of the sensor reading that triggered this event.",  
        "type":"integer"  
    }  
}  
  
}
```

### 3.11.5 Summary

The Smart Object Integration component addresses the communication between Smart Objects and the ADVENTURE system. This component supports the IEEE 802.15.4 and the Bluetooth communication standards. As technology platform for this component smartphones are used. In combination with the mentioned communication technologies Android is selected as the smartphone operation system. On the base of these, the logic and the communication requirements will be developed and implemented. The communication between the Smart Objects and the ADVENTURE system will be realized in a push- and pull-based manner, whereby a JSON-based communication is envisioned.

## 4 Conclusion

This deliverable provides the technical specification of the ADVENTURE platform, which followed the functional specification deliverable (D3.2). It is fully aligned with the project's submitted deliverables, specifically, user requirements deliverable (D2.3), global architecture deliverable (D3.1) and functional specification deliverable (D3.2). This deliverable defines the detailed technical design of each of the eleven ADVENTURE components.

Each of the components is elaborated with respect to its major design decisions, technology comparison and selection, technical specification and specification of interfaces, protocols and formats. In technology comparison process, generic and specific parameters were considered in order to select the suitable technology for the specific component. During this selection process, associated licensing issues for the individual components were also considered with respect to possible implementation perspective.

An overall technical characterization of each of the components as defined in the ADVENTURE global architecture is explained in terms of component's functionality and implementation perspective. The required interfaces and communication protocols between the components are detailed with respect to component specifications and internal sequence diagrams. The related API specification of the individual components is technically analyzed and the data exchange between the components is defined with necessary content formats and protocols.

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>364</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			



## Annex A

### *List of Abbreviations*

#### **A**

API — Application Programming Interface

ACL — Access Control List

AJAX — Asynchronous JavaScript and XML  
([http://en.wikipedia.org/wiki/Ajax\\_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming)))

AS2 — ActionScript Applicability Statement 2  
(<http://en.wikipedia.org/wiki/AS2>)

AS3 — ActionScript Applicability Statement 3.03  
([http://en.wikipedia.org/wiki/AS3\\_\(networking\)](http://en.wikipedia.org/wiki/AS3_(networking)))

Active MQ — Active Message Queuing  
(<http://activemq.apache.org/>)

#### **B**

BAM — Business Activity Monitoring  
(<http://www.businessactivitymonitoring.com/>)

BPMN — Business Process Modeling and Notation

BPM — Business Process Management

BPMS — Business Process Management Suite

BPEL — Business Process Execution Language  
(<http://en.wikipedia.org/wiki/BPEL>)

BPMN2.0 — Version 2.0 of Business Process Model and Notation OMG Standard  
(<http://www.bpmn.org/>)

#### **C**

CRUD — Create, Read, Update, Delete  
([http://en.wikipedia.org/wiki/Create,\\_read,\\_update\\_and\\_delete](http://en.wikipedia.org/wiki/Create,_read,_update_and_delete))

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>365</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

CORE ontology — A basic and minimal ontology consisting only of the minimal concepts required

([http://en.wikipedia.org/wiki/Core\\_ontology](http://en.wikipedia.org/wiki/Core_ontology))

CPLEX — IBM ILOG CPLEX Optimizer

(<http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>)

CSV — comma—separated values

([http://en.wikipedia.org/wiki/Comma-separated\\_values](http://en.wikipedia.org/wiki/Comma-separated_values))

## D

DOM — Document Object Model

(<http://www.w3.org/DOM/DOMTR>)

## E

EAV — Entity, Attribute, Value

EAI — Enterprise Application Integration

([http://en.wikipedia.org/wiki/Enterprise\\_application\\_integration](http://en.wikipedia.org/wiki/Enterprise_application_integration))

ebXML — Electronic Business using XML

(<http://en.wikipedia.org/wiki/EbXML>)

eCI@ss — Cross—industry product data standard classification

(<http://www.eclass.de/eclasscontent/standard/overview.html.en>)

ESB — Enterprise Service Bus ([http://en.wikipedia.org/wiki/Enterprise\\_service\\_bus](http://en.wikipedia.org/wiki/Enterprise_service_bus))

EDI — Electronic Data Interchange

([http://en.wikipedia.org/wiki/Electronic\\_data\\_interchange](http://en.wikipedia.org/wiki/Electronic_data_interchange))

EIP — Enterprise Integration Patterns

([http://en.wikipedia.org/wiki/Enterprise\\_Integration\\_Patterns](http://en.wikipedia.org/wiki/Enterprise_Integration_Patterns))

ERP — Enterprise Resource planning

([http://en.wikipedia.org/wiki/Enterprise\\_resource\\_planning](http://en.wikipedia.org/wiki/Enterprise_resource_planning))

ECOLEAD — European collaborative networked organizations leadership initiative

(<http://ecolead.vtt.fi/>)

## F

FOAF — Friend of a Friend

(<http://xmlns.com/foaf/spec/>)

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>366</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

FTP — File Transfer Protocol  
(<http://en.wikipedia.org/wiki/FTP>)

## G

GRAPH —HTTP: Hypertext Transfer Protocol  
(<http://www.w3.org/Protocols/rfc2616/rfc2616.html>)

GUI — Graphical User Interface

GWT — Google Web Toolkit

GoodRelations —Web vocabulary for e-commerce  
(<http://www.heppnetz.de/ontologies/goodrelations/v1.html>)

GPC — Global Product Classification  
(<http://www.gs1.org/gdsn/gpc>)

GDSN — Global Data Synchronization Network  
(<http://www.gs1.org/gdsn>)

Google Product Taxonomy — Google tree of categories that describe product families  
(<http://support.google.com/merchants/bin/answer.py?hl=en-AU&answer=160081>)

## H

HTML — HyperText Markup Language  
(<http://www.w3.org/TR/2011/WD-html5-20110525/>)

HTTP — Hypertext Transfer Protocol

## I

ISIC — International Standard Industrial Classification of All Economic Activities  
(<http://thedatahub.org/dataset/isic-v4>)

IT — Information Technology

## J

JSR — Java Specification Request

jQuery — Cross—browser JavaScript library  
(<http://jquery.com/>)

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>367</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

JavaScript — The scripting language of the Web

(<http://en.wikipedia.org/wiki/JavaScript>)

JSON — JavaScript Object Notation

([http://de.wikipedia.org/wiki/JavaScript\\_Object\\_Notation](http://de.wikipedia.org/wiki/JavaScript_Object_Notation))

JEE — Java Platform Enterprise Edition

jBPM — JBoss Business Process Management suite

(<http://www.jboss.org/jbpm>)

JAAS — Java Authentication and Authorization Service

## K

KPI — Key Performance Indicator

## L

Linked USDL — Remodeled version of USDL that builds upon the Linked Data principles

(<http://linked-usdl.org/>)

LinkedData — Connect Distributed Data across the Web

(<http://linkeddata.org/>)

## M

MSM — Minimal Service Model

([http://lov.okfn.org/dataset/lov/details/vocabulary\\_msm.html](http://lov.okfn.org/dataset/lov/details/vocabulary_msm.html))

Microdata — Is a WHATWG HTML5 specification used to nest semantics within existing content on Web pages.

([http://en.wikipedia.org/wiki/Microdata\\_\(HTML\)](http://en.wikipedia.org/wiki/Microdata_(HTML)))

## N

NEFFICS — Networked Enterprise transFormation and resource management in Future internet enabled Innovation CloudS

(<http://neffics.eu>)

NACE — Nomenclature Générale des Activités Économiques dans les Communautés Européennes (French, EU classification system)

([http://epp.eurostat.ec.europa.eu/cache/ITY\\_OFFPUB/KS-RA-07-015/EN/KS-RA-07-015-EN.PDF](http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-RA-07-015/EN/KS-RA-07-015-EN.PDF))

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>368</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

noSQL — not only Structured Query Language  
(<http://nosql-database.org/>)

## O

OData — Open Data Protocol  
(<http://www.odata.org/>)

OSGi — Open Services Gateway initiative  
(<http://en.wikipedia.org/wiki/OSGi>)

OMG — Object Management Group  
(<http://www.omg.org>)

## P

PMF — OMG Party Management Facility  
(<http://www.omg.org/spec/PARTY/1.0>)

## R

RDBMS — Relational Database Management System  
([http://en.wikipedia.org/wiki/Relational\\_database\\_management\\_system](http://en.wikipedia.org/wiki/Relational_database_management_system))

RDF — Resource Description Framework

RDMS — Relational Database Management System

RDFS — RDF Schema  
(<http://schema.rdfs.org/>)

RDBMS — Relational DataBase Management System  
([http://en.wikipedia.org/wiki/Relational\\_database\\_management\\_system](http://en.wikipedia.org/wiki/Relational_database_management_system))

REST — REpresentational State Transfer

RESTful WebServices — WebServices using REpresentational State Transfer  
(<http://en.wikipedia.org/wiki/REST>)

RTM — Real Time Monitoring

## S

SaaS — Software as a Service  
([http://en.wikipedia.org/wiki/Software\\_as\\_a\\_service](http://en.wikipedia.org/wiki/Software_as_a_service))

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>369</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

Schema.org — A collection of shared Web vocabularies  
(<http://schema.org>)

SOAP — Simple Object Access Protocol

SOA — Service Oriented Architecture  
([http://en.wikipedia.org/wiki/Service-oriented\\_architecture](http://en.wikipedia.org/wiki/Service-oriented_architecture))

SOTA — State Of The Art

SME — Small and Medium sized Enterprises

SEEM — Single Electronic European Market  
(<http://www.seamless-eu.org/>)

SQL — Structured Query Language  
(<http://en.wikipedia.org/wiki/SQL>)

SPARQL — Protocol and RDF Query Language  
(<http://www.w3.org/TR/rdf-sparql-query/>)

SMTP — Simple Mail Transfer Protocol  
([http://en.wikipedia.org/wiki/Simple\\_Mail\\_Transfer\\_Protocol](http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol))

SSL — Secure Socket Layer  
([http://en.wikipedia.org/wiki/Secure\\_Sockets\\_Layer](http://en.wikipedia.org/wiki/Secure_Sockets_Layer))

SVG — Scalable Vector Graphics  
(<http://en.wikipedia.org/wiki/SVG>)

SVN — Apache Subversion

## T

TRADE — OMG's Trading Object Service  
(<http://www.omg.org/spec/TRADE/1.0>)

## U

UI — User Interface

UIDL — User Interface Definition Language

USDL — The Unified Service Description Language  
(<http://www.w3.org/2005/Incubator/usdl/>)

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>370</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

UNSPSC — United Nations Standard Products and Services Code

(<http://www.cs.vu.nl/~mcaklein/unspsc/unspsc84-title.rdfs>)

UTF-8 — UCS Transformation Format-8-bit

URI — Universal Reference Identifier

([http://en.wikipedia.org/wiki/Uniform\\_resource\\_identifier](http://en.wikipedia.org/wiki/Uniform_resource_identifier))

UUID — Universally Unique IDentifier

([http://en.wikipedia.org/wiki/Universally\\_unique\\_identifier](http://en.wikipedia.org/wiki/Universally_unique_identifier))

## V

VAN — Virtual Area Network

([http://en.wikipedia.org/wiki/Value-added\\_network](http://en.wikipedia.org/wiki/Value-added_network))

VFF — Virtual Factory Framework

(<http://www.vff-project.eu/>)

## W

WSRP — Web Services for Remote Portlets

WHATWG — Web Hypertext Application Technology Working Group

(<http://www.whatwg.org/>)

WSDL — Web Services Description Language

(<http://en.wikipedia.org/wiki/WSDL>)

WSO2 — Business Activity Monitor (WSO2 BAM)

(<http://wso2.com/>)

W3C — World Wide Web Consortium

(<http://www.w3.org/>)

## X

XML — Extensible Markup Language

(<http://www.w3.org/XML/>)

XMPP — Extensible Message and Presence Protocol

(<http://xmpp.org/>)

XPath — XML Path Language

(<http://www.w3.org/TR/xpath20/>)

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>371</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			

XEP — XMPP Extension Protocol

([http://en.wikipedia.org/wiki/XMPP\\_Standards\\_Foundation](http://en.wikipedia.org/wiki/XMPP_Standards_Foundation))

## ADVENTURE SPECIFIC ABBREVIATIONS

AID — ADVENTURE Identifier

CSS — Cloud Storage System

DPD — Data Provisioning and Discovery

PD — Process Designer

PMM — Process Model Management/Manager

PME — Process Model Editor

PM — Process Monitoring

SPE — Smart Process Execution

D3.3_Technical Specification_v2.2-final	Author: UVA and Partners	Date:2012-09-24	Page: <b>372</b> / 372
Copyright © ADVENTURE Project Consortium. All Rights Reserved.			