Deliverable D500.4.1

Guidelines for the use of Standards in FIspace

WP 500

Project Acronym & Number:	FIspace – 604 123
Project Title:	FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics
Funding Scheme:	Collaborative Project - Large-scale Integrated Project (IP)
Date of latest version of Annex 1:	03.10.2013
Start date of the project:	01.04.2013
Duration:	24
Status:	Final
Authors:	Christopher Brewster, Andreas Füßler, Scott Hansen, Sabine Kläser, Andrew Josey, Daniel Martini, Esther Mietzsch, Chris Parnel, Tim Sadowski, Angela Schillings-Schmitz, Monika Solanki
Contributors:	Sven Lindmark, Sjaak Wolfert
Document Identifier:	FIspace-D500.4.1-Standards-V012-Final.docx
Date:	19 September 2013
Revision:	012
Project website address:	http://www.FIspace.eu

The FIspace Project

Leveraging on outcomes of two complementary Phase 1 use case projects (FInest & SmartAgriFood), aim of FIspace is to pioneer towards fundamental changes on how collaborative business networks will work in future. FIspace will develop a multi-domain Business Collaboration Space (short: FIspace) that employs FI technologies for enabling seamless collaboration in open, cross-organizational business networks, establish eight working Experimentation Sites in Europe where Pilot Applications are tested in Early Trials for Agri-Food, Transport & Logistics and prepare for industrial uptake by engaging with players & associations from relevant industry sectors and IT industry.

Project Summary

As a use case project in Phase 2 of the FI PPP, FIspace aims at developing and validating novel Future-Internet-enabled solutions to address the pressing challenges arising in collaborative business networks, focussing on use cases from the Agri-Food, Transport and Logistics industries. FIspace will focus on exploiting, incorporating and validating the Generic Enablers provided by the FI PPP Core Platform with the aim of realising an extensible collaboration service for business networks together with a set of innovative test applications that allow for radical improvements in how networked businesses can work in the future. Those solutions will be demonstrated and tested through early trials on experimentation sites across Europe. The project results will be open to the FI PPP program and the general public, and the pro-active engagement of larger user communities and external solution providers will foster innovation and industrial uptake planned for Phase 3 of the FI PPP.

Project Consortium

- DLO; Netherlands
- ATB Bremen; Germany
- IBM; Israel
- KocSistem; Turkey
- Aston University; United Kingdom
- ENoLL; Belgium
- KTBL; Germany
- NKUA; Greece
- Wageningen University; Netherlands
- PlusFresc; Spain
- FloriCode; Netherlands
- Kverneland; Netherlands
- North Sea Container Line; Norway
- LimeTri; Netherlands
- Kühne + Nagel; Switzerland
- University Duisburg Essen; Germany
- ATOS; Spain
- The Open Group; United Kingdom
- CentMa; Germany
- iMinds; Belgium
- Marintek; Norway
- University Politecnica Madrid; Spain
- Arcelik; Turkey
- EuroPoolSystem; Germany
- GS1 Germany; Germany
- Mieloo & Alexander; Netherlands
- OPEKEPE; Greece
- Innovators; Greece

More Information

Dr. Sjaak Wolfert (coordinator)
LEI Wageningen UR
P.O. Box 35
6700 AA Wageningen

e-mail: sjaak.wolfert@wur.nl
phone: +31 317 485 939
mobile: +31 624 135 790
www.FIspace.eu

Dissemination Level

PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Change History

Version	Notes	Date
001	Creation of the document	June 2013
002	Revision and addition	27.06.2013
003	Revision and addition	05.07.2013
004	Revision and addition	15.07.2013
005	First draft for review	17.07.2013
006	Final draft for first review	23.07.2013
007	Commented version from first review	29.07.2013
008	Revision and addition	05.08.2013
009	Revision and addition	23.08.2013
010	Revision and addition	26.08.2013
011	Revision and addition	29.08.2013
012	Final revision	19.09.2013

Document Summary

For fostering and demonstrating potential for innovation of FIspace related to market impact in the food and logistics sector, aspects looked into in FIspace work package 500 on 'Open collaboration and exploitation' range from the exploitation plans of all the consortium partners, over IPR aspects and business models, to policy and regulatory aspects and, last but not least to openness and the standardization approach.

The latter dimension of standardization is covered by FIspace Task 540 of which this document represents the first out of three deliverables covering the investigation on technology standards for cross sectorial system and data integration. It contains on the one hand a compilation of standards relevant to the agri food and logistics sector and enriches this on the other hand with relevant standards for building electronic platforms to apply business applications onto. Hence, this document is intended to guide the use case trials (WP400) concerning the use of standards and related technology to ensure standards are used throughout the FIspace project wherever possible.

This deliverable is partly based on the work of the SmartAgriFood Deliverable 600.2 'Plan for standardisation for large scale experimentation' [1] (The former project SmartAgriFood addressed the food and agribusiness as a use case for the Future Internet.).

Several platform technologies have been identified and described in this document concerning application development, server-side components, event-processing, application store, system and data integration and security. System architecture considerations have been taken into account.

Moreover, agriculture sector specific standards regarding identification, labelling, communication and certification have been identified and described.

As forming a momentous for future internet applications, semantic web technologies have been given extra attention by integrating it into the context of FIspace.

The document will guide the trials of the FIspace project regarding available standards and specifications. It is the first step within an iterative process providing application developers within the FIspace trials with a tool box of what exists today and receive their feedback for improvements where considered necessary.

Abbreviations

AP	Agricultural parcel
API	Application Programming Interface
App	Software Application
BCM	Business Collaboration Module
CP	Cadastral parcel
D	Deliverable
DoW	Description of Work
DDS	Data Distribution Level
EC	European Commission
EDI	Electronic Data Interchange
e.g.	Exempli gratia = for example
EPC	Electronic Product Code
EPCIS	Electronic Product Code Information Services
EPM	Event Processing Module
ESB	Enterprise System Bus
EU	European Union
FB	Farmers' block/ilot
FIA	Future Internet Assembly
FI PPP	Future Internet Public Private Partnership
FOAF	Friend-of-a-Friend Project
GCP	GS1 Global Company Prefix
GDD	Global Data Dictionary
GDSN	Global Data Synchronisation Network
GE	Generic Enabler
GLN	Global Location Number
GML	Geography Markup Language
GPC	Global Product Classification
GRAI	Global Returnable Asset Identifier
GTIN	Global Trade Item Number
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IAC	Integrated Administration and Control System (IACS)
IETF	The Internet Engineering Task Force
ICT	Information and Communication Technology
IDE	Integrated Development Environment
i.e.	id est = that is to say
IP	Intellectual Property
IPR	Intellectual Property Rights
ISO	International Organization for Standardization
JMS	Java Message Service
JSP	JavaServer Pages
JSON	JavaScript Object Notation
JSON-LD	JSON for Linking Data
KPI	Key Performance Indicator
LDAP	Lightweight Directory Access Protocol
LPIS	Land Parcel Identification System
LF	Low Frequency
M	Month
ONS	Object Naming Service
OSGI	Open Services Gateway Initiative
PB	Physical block
RP	Reference Parcel
RP	Relying Parties
RDFa	Resource Description Framework in Attributes
RTD	Research and Technological Development
S/MIME	Secure/Multipurpose Internet Mail Extensions
SAML	Security Assertion Markup Language
SGTIN	Global Trade Item Number
SGLN	Serialized Global Location Number
SME	Small and Medium Sized Enterprise
SMTB	Simple Mail Transfer Protocol
SSCC	Serial Shipping Container Code
SSL	Secure Sockets Layer
ST	Sub-Task
SWT	Semantic Web Technologies

UN/EDIFACT	United Nations Electronic Data Interchange for Administration, Commerce and Transport
USDL	Unified Service Description Language
T	Task
TLS	Transport Layer Security
TCP	Transmission Control Protocol
W3C	World Wide Web Consortium
WGS	World Geodetic System
WP	Work Package
WSDL	Web Services Description Language
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Content and Purpose

To guide the certain trials envisaged within the FIspace project regarding available standards and specifications as base for services, app development and testing, the document covers the spectrum of most relevant aspects concerning application development, server-side components, event processing, application store system and data integration, security, data exchange and semantic web.

It is the first step within an iterative process providing application developers within the FIspace trials with a tool box of what exists today. It should form the equipment to be operated in to validate whether it serves the specific needs. Feedback reported back from the trials will serve as input for FIspace deliverable 500.4.2 to derive recommendations for new or updated standards.

Standards are crucial for efficient cross company interaction and for the development of interoperable applications and platforms. Often they are a prerequisite for systems to be able to communicate at all e.g. for electronic data interchange. They guarantee uniqueness and unambiguity of data which is important for the retrieval of reliable object related data in an open environment and for the attribution of data to an individual object. They also are essential building blocks for providing the functionalities envisioned by FIspace through the provision of an open and extensible framework for deploying cross-sectorial business applications in order to carry out business transactions.

This document deals with existing standards that support FI based business processes and transactions that are enabled by the FIspace platform and technologies. This compilation describes the expected technology and content based standards that will potentially be exploited within FIspace and in some cases extended to support new FIspace capabilities. Production related standards and product quality standards (e.g. organic food or fair-trade certification) are generally considered out of scope in this deliverable.

The set of standards described can roughly be subdivided into communication/transaction or platform, identification and classification standards. Sometimes they have been triggered by European law but in general they are complementary. This document already covers co-operation and interaction between Work Packag 200 and Task 540. This especially applies to contribution to platform related standards (Chapter 2). Apart from this first input on domain specific standards relevant for WP 441 Meat Information Provenance Trial has been considered. By consistently utilizing the relevant standards in the FIspace WP 400 trials transferability of business processes between the trials should be achievable. Beyond this the standard set gives guidance for new requirements evolving from the trials. Within the FIspace project requirements for updates and a way forward towards the most sensible standard mix should become clear and be part of future deliverables within T540.

FIspace as a whole is a continuation and merger of the FI-PPP phase 1 projects SmartAgriFood and FInest. The specific task on standardization of this document builds on the SmartAgriFood Deliverable D600.2 [1] which laid out the existing data exchange standards relevant to the agri-food sector as well as

logistics. Other relevant documents include SmartAgriFood Deliverables D100.4 (Strategic overview) and Deliverable D100.3 (Policy and Regulation Harmonisation) [2].

1.2 Standards organisations, approaches and results

Many standardisation processes in principle follow the same sequential steps (such as requirement analysis, solution development, IPR assessment, etc.), but they do not necessarily generate the same results. This may be due to differences between the nature of organisations, or due to a specific approach (e.g. formal or non-formal) towards standardisation processes. It can also be a result of an organisation's participants aiming at specific standardisation deliverables (e.g. guidelines documents or test specifications).

When considering the various candidate standards involved in FIspace related software development we must consider the differences between types of standards bodies, as well as differences between the standardisation processes they support, and between the standardisation deliverables they produce.

1.2.1 Different types of standards bodies

On a European level, there are three formal standards organisations: CEN[1], CENELEC[2] and ETSI[3]. These are recognised by the European Commission and meet the World Trade Organisation criteria for standards setting. All three have cooperation arrangements in place with their global counterparts: ISO, IEC and ITU. In addition, there are several formal standards bodies working at a national level, which also have wider impact (e.g. DIN[4], ANSI[5] or BSI[6]).

Formal standardisation processes require relatively long periods for approval processes to be completed. However, many aspects of ICT standardisation are covered by industry consortia and trade organisations rather than formal standards bodies. Industry consortia do not primarily aim at producing formal standards, and many times set out to address or resolve only a limited number of specific issues. Despite the less formal character of the industry standards they produce, their strong focus on specific market segments or technical challenges often proves to be an efficient way for generating critical mass among stakeholders, necessary for successfully completing standardisation processes.

As an example GS1 is 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 in the world. Their standards are designed for intersectoral and longterm use, maintenance and development. Having granted ARO[7] status by ISO/IEC Joint Technical Committee they are allowed to be referenced by ISO.

[1] European Committee for Standardization
[2] European Committee for Electrotechnical Standardization
[3] European Telecommunications Standards Institute
[4] German Institute for Standardization
[5] American National Standards Institute
[6] British Standards Institute
[7] Approved Referencing Specification Originator

1.2.2 Different types of standardisation results

The ICT standardisation environment is characterised by a large number of standards bodies, generating an even larger number of standardisation activities. Even with these differences however, the deliverables resulting from these activities can be grouped as follows:

- **Formal standards**, sometimes also referred to as de jure standards, are normative documents from formal standards bodies and have passed through a full and open consensus process. They are implemented on a national level and there is strong pressure to apply them; formal standards have a legal basis and can be made mandatory but considerable time (up to 4 years) is needed for completing the full approval process.
- **Technical or industry specifications** are based on consensus among members of standards bodies, consortia or trade organisations and do not have a formal character or legal basis; they are recommendations and require less time to produce (1-3 years) but when widely accepted and used in practice by relevant market players they can become de facto standards.
- **Workshop Agreements** are industry recommendations developed by interested stakeholders through a short-track process (6-12 months) often facilitated by several formal standards bodies; workshop agreements serve as industrial consensus documents between participating individuals and organisations, and can be revised relatively easily.
- **Conformance, test applications, reference implementations and guidelines** aim to support interoperability between and easy rollout by market players of products and services based on formal standards or industry specifications. They have an informative character and are usually produced in a relatively short timeframe (6-12 months).
- **Technical reports** are informative documents supporting further standardisation work, e.g. by identifying the need for additional technical clarifications in – or between – existing specifications, standards, or guideline documents.

Both formal standards and industry specifications that are developed in an open process and are publicly available under so called Fair, Reasonable and Non-Discriminatory (FRaND) terms, can be regarded as ‘open standards’. Nevertheless, there can be a trade-off between the impact of a formal standard, and the amount of time (and in some cases also resources) it takes to produce.

Although FIspace does not focus on developing new standards, the work undertaken in the project due to its business focus and practical nature will inevitably contribute to the establishment of new or revised industry specifications and thereby more formal standardisation.

1.3 Government role

National governments often play an important role in establishing standards for industry. In some cases governments will mandate the use of specific standards developed by government recognised standards bodies related to safety, communications or other areas having a broad affect on society. Governments also influence standards through procurement where references to standards in government purchasing requirements can often motivate suppliers of products and technologies to comply with standards in order to be eligible for large government contracts for products and services.

While governments most often reference standards specified by recognised standards bodies such as CEN, CENELEC, ETSI or ISO, they sometimes establish standards directly through regulations that mandate specific information be supplied with products or that information concerning goods or materials be provided following specific categorisations or formats. In these cases the data categorisation or formats are defined by the government regulation itself, which provides as part of the mandate for compliance a standard specification for information exchange. For example, in the food sector a standard for the labelling of beef and veal has been specified and mandated across Europe directly by the European Commission rather than referencing a labelling standard from a standards body. The FIspace framework intends to support standards coming from both standards bodies and those directly specified by government mandates, and both types of standards are addressed in this deliverable.

1.4 Emerging standards

The FIspace project is addressing state-of-the-art technologies for an Internet based framework to enable the development and deployment of business components and services, which is a relatively new field of technology development. Some of the most relevant technologies and specifications related to FIspace are *de facto* standards that are recognised more for their widespread usage or acceptance, rather than having passed through a process of industry review and consensus or formal approvals.

These emerging standards often have associated communities of users who create momentum for industry acceptance and adoption even though the standards development process is often no more structured than a website for downloading a specification or an open source reference implementation, and a discussion board where the main contributors to the specifications and industrial and academic users can interact. Nonetheless, there are several examples of successful industry standards that were established through very similar arrangements (e.g. Linux).

A natural progression for many of these emerging standards is that alternative implementations or specifications appear which are driven by different application domain specific requirements. When this occurs and the emerging standards reach a level of maturity, efforts to converge the various alternatives are undertaken. This convergence process functions quite similarly to the consensus process utilised by member based standards bodies, consortia or trade organisations.

The FIspace project expects that many of the specifications and technologies that will eventually be considered as established FIspace standards will follow similar informal paths to industry acceptance and *de facto* standardisation. For this reason, some of the relevant emerging standards related to FIspace have been included in this deliverable.

1.5 Structure of Deliverable

The previous sections enable the reader to differentiate types of standards and to comprehend the roles and responsibilities of parties involved in the standardisation process in order to understand the impact and relevance of standards for the FIspace trials.

Based on this presentation, the following chapters identify the standards considered relevant for being contemplated in the FIspace trials in the context of platform technologies (Chapter 2), sector-specific

standards (Chapter 3) and regarding semantic web technologies (Chapter 4). Two annexes provide deepening information on the standards referred to in Chapter 2 and 3 (Annex 1) respectively an outline on semantic web technology (Annex 2).

2 Platform Standards

2.1 Introduction

The overall aim of the FIspace project is to implement a novel Future Internet enabled cloud-based platform for enabling easy, effective, and seamless collaboration in business networks across organizational borders (see Figure 1 and Figure 2). The FIspace platform provides the generic software infrastructure that enables the envisioned seamless collaboration in business networks and supports the development, provisioning, and consumption of FIspace Apps that provide novel, value-added functionalities for business collaboration in several application domains. The platform is composed of various components that interact to provide a set of services. Some of these services are visible directly as user features, while others are enabling services that are used to implement higher level user features or ensure certain characteristics of the FIspace platform are maintained such as those related to performance levels, security interoperability and many others.

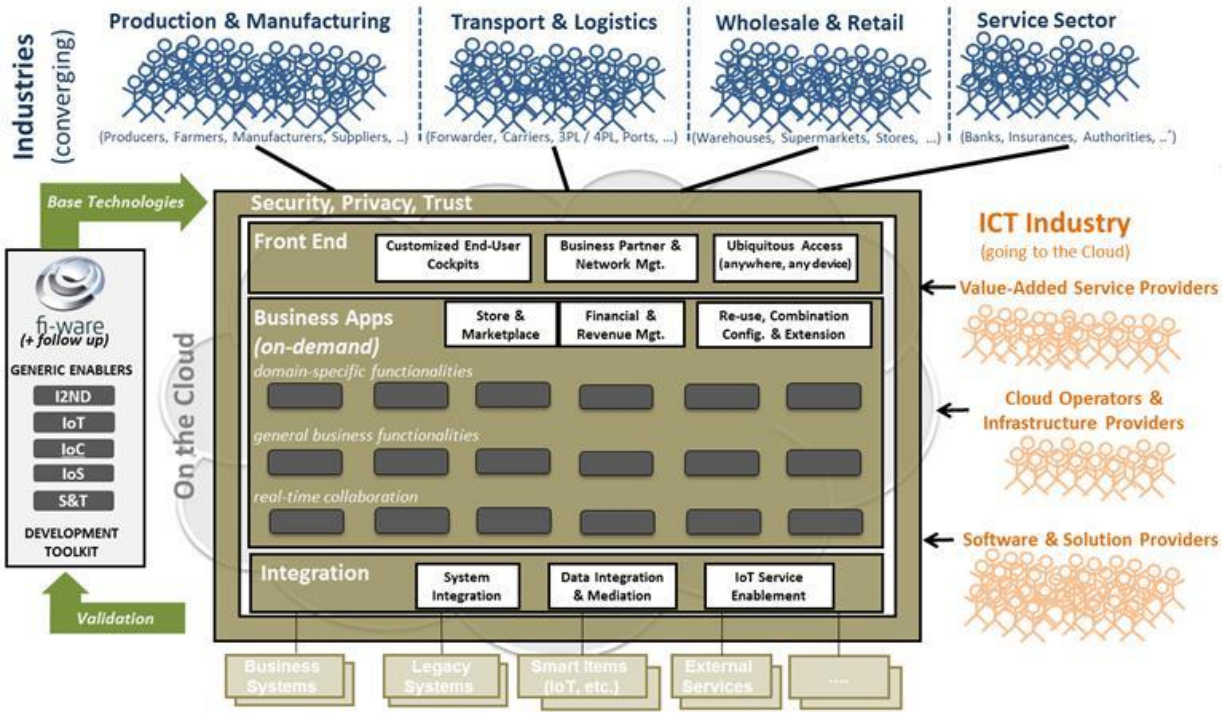


Figure 1: FIspace Overall Vision - A multi-domain Collaboration Space for Business Networks

The main technology areas where there are opportunities to exploit standards within the FIspace platform are the following:

- Application development (see Section 2.2)
- Server-side components (see Section 2.3)
- Event processing (see Section 2.4)
- Application Store (see Section 2.5)

- System and data integration (see Section 2.6)
- Security (see Section 2.7) and
- GS1 System architecture (see section 2.8)

The relationship between these areas is shown in the high level FIspace system architecture in Figure 2. Software developers that wish to exploit the FIspace platform use the platform components and associated services to construct business applications targeting domain-specific needs and requirements. The platform components themselves must interact and operate as an integrated system to deliver required services and must support the exchange of information both internally and with external resources. Finally all of the activities carried out in FIspace must be built upon security technologies that provide required levels of assurance for business users to carry out commercial transactions. For identifying, capturing and sharing of information, the GS1 System architecture provides a standard based environment to cater especially for serving open supply chain needs.

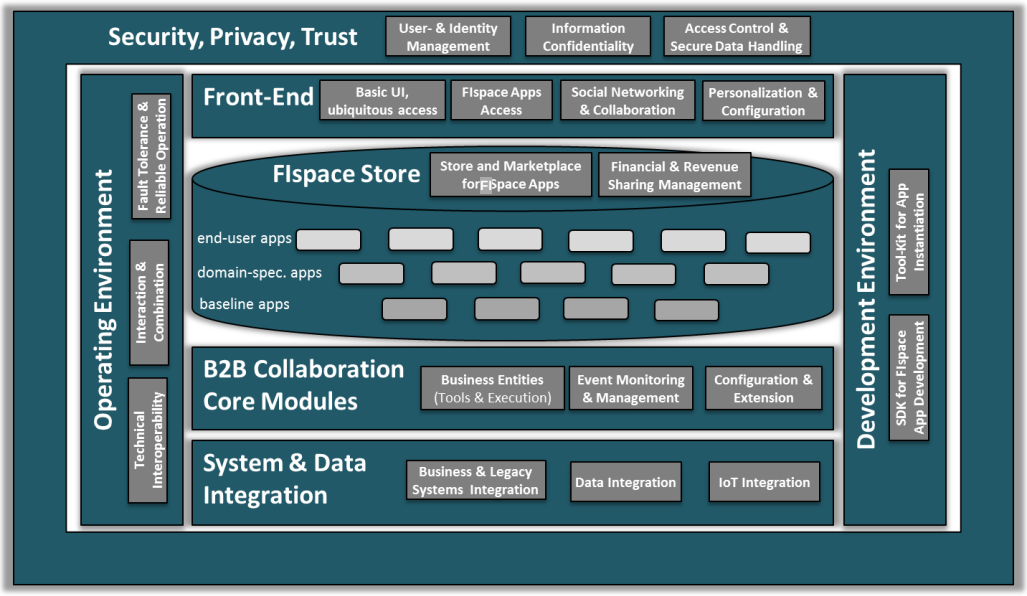


Figure 2: High Level FIspace System Architecture

The following sections describe the opportunities in each of the main platform technology areas to exploit established and emerging standards.

2.2 Application development

The FIspace platform will provide to applications developers a Software Development Kit (SDK) to ease their work during their implementation of applications, providing some specific tools and hiding the complexity of the FIspace platform.

The three core technologies where standards exist that can be used for FIspace application development are:

- **Languages** – used by application developers to implement features in software. These are used to implement application logic, present and manipulate the interface to users, and to implement the protocols to interact with FIspace components

- **Development environments** – provides a set of supporting tools and plug-ins to assist the application developer in automating the design, testing and deployment of applications implemented in one or more programming languages
- **Integration mechanisms** – provides the ability to combine applications to create “mashups” where new custom applications can be easily constructed through composition of existing applications

The FIspace project will likely use the Java programming language standard as the basis for implementing most of the components of the FIspace platform. However, this does not preclude the use of other language standards for the development of FIspace applications. The following describes the candidate standards for programming languages, development environments and integration mechanisms that FIspace is considering to support for use by application developers.

2.2.1 Languages

Candidates for language standards for developing FIspace functionalities and application software are Java, C++, C#, HTML, HTML5, CSS and JavaScript. For a brief decriptions of each of the language standards, please see Annex 1.

2.2.2 Development environments

The following industry recognised standards Eclipse and NetBeans for application development environments are being considered for developer support in creating FIspace applications. They are standards of certain fora or consortia. For their description, see Annex 1.

2.2.3 Integration mechanisms

The application integration features of the FIspace platform targets to provide functionalities to organise and customise a user’s personal environment by combining parts of different FIspace applications. There are several tools available that could be utilised to provide this kind of integration. Some examples are:

- **WireCloud GE** - a reference implementation of the FIWARE “Application Mashup“. It comes with its own graphical wiring editor that allows creating mashup applications out of two or more apps by connecting the output parameters of one app with input parameters of another app.
- **Apache Rave** – an open source social mashup engine developed under the umbrella of the Apache Software Foundation. Published applications can be added to and freely arranged in a user’s personal workspace.
- **JBoss GateIn** – allows a user to create a personal dashboard and populate it with several applications (so-called portlets).

Standards that are used for these types of application integration tools are specification for widgets and portlets, which are parts of applications that can be manipulated within a user working space. Related standards of certain companies, fora or consortia in these areas are Widgets and Java Portlets which are described in Annex 1.

2.3 Server-side components

FIspace provides service interfaces for applications to access additional business logic that is provided on the server-side within the client server model used for FIspace applications. FIspace will include an implementation for server side logic, which also provides the necessary interfaces, and will include a web- or application-server which executes the implemented logic. The most relevant standards for FIspace server-side components are described in Annex 1. They are Java Servlet, JavaServer Pages and WebSockets.

The following are possible candidate server implementations that support all or some subset of the standards mentioned in Annex 1. These implementations have largely become *de facto* standards from their widespread usage:

- **Apache Tomcat** – Tomcat is an open source application server for Java-based applications. It implements the Java Servlet and Java ServerPages specification directly, but also can be extended to support the whole Enterprise Java Stack.
- **Eclipse Jetty** – Jetty is an alternative application server to Tomcat. It also implements the Servlet and ServerPages specification, but provides additionally support for the WebSockets protocol.
- **Spring** – Spring is one of the most popular frameworks for enterprise Java applications. It consists of a variety of different modules, which can be used in very different scenarios and has facilities for linking to social networking sites like Facebook and Twitter.
- **CometD Java Server** – CometD framework allows the technology independent integration of server push mechanisms.

The FIspace project will likely utilise one or more of these server implementations within the FIspace platform.

2.4 Event processing

The FIspace platform will utilise an Enterprise System Bus (ESB) for communications between the two core components enabling business collaboration. The Business Collaboration Module (BCM) takes care of the execution of collaborative business processes among FIspace users, while the Event Processing Module (EPM) is where events are generated to be shared with other users, or where events received from others are processed. The FIspace ESB will use a Publish/Subscribe mechanism. In this interaction pattern, a consumer registers to specific events (subscriptions) and a producer pushes events to the bus (publish) via a Pub/Sub Event API.

Standards that are potential candidates for supporting event processing within the FIspace platform are the following: JSON, JMS and DDS (see also Annex 1 for a brief description).

For FIspace, JSON is a candidate standard for the event processing definitions. In particular the event types and rules to be implemented. The definitions would be processed by the FIspace Complex Event Processing run-time engine from a configuration file in JSON format.

2.5 Application Store

The FIspace Application Store and the necessary platform infrastructure is planned to be developed on top of existing frameworks and technologies; primary candidates are the Generic Enablers from the FI-WARE. In particular, using (Linked) USDL as the basis for the Application Description Language, using the Service Repository as basis for the FIspace App repository and the Marketplace GE as basis for features of the Consumption Support and Provisioning components.

Standards that could potentially be used for the implementation of the Application Store are WSDL and USDL, further described in Annex 1.

2.6 System and Data Integration

One of the roles of the FIspace platform is to provide a robust and scalable infrastructure that enables seamless integration of external legacy systems/IoT systems and applications deployed on it. To facilitate the implementation of web based, FIspace applications, unifying data models, data mediation tools and system integration APIs will be utilised.

REST or Representational State Transfer based interfaces often appear within the context of system or component integration. We do not specifically identify REST as a standard within the context of this deliverable as REST is an architecture style for designing networked applications. The idea is that, rather than using complex mechanisms such as SOAP to connect machines, the standard Hypertext Transfer Protocol (HTTP) protocol is used to provide the fundamental mechanisms to post data (create and/or update), read data (e.g., make queries), and delete data. RESTful protocols do not follow a prescribed standard except for the underlying use of HTTP, and it is very likely the FIspace project will create and utilise several RESTful APIs specific to FIspace for integrating FIspace components and other third party systems in accordance with industry best practices.

Candidate standards related to system and data integration are SOAP, NGSI, OSGi (see also Annex 1). Standards that will enable flexible system and data integration of FIspace components and likely be used in developing the FIspace platform will be the Java Dependency Injection API, the JAX-RS - Java API for RESTful Web Services, and the JAX-WS Java API for XML based Web Services (also described in Annex 1).

2.6.1 SOAP

SOAP (Simple Object Access Protocol) is based on XML (see [3]) and is a communication protocol for exchanging information between applications by sending messages. SOAP provides a standard format for sending messages that is platform and language independend. SOAP is used more often within larger industrial organisations for managing system and data integration. It may be necessary to utilise simpler mechanisms or to create custom RESTful protocols within FIspace for addressing the needs and limitations of smaller supply chain actors.

2.6.2 NGSI

The Context Management component of FIspace will likely use the NGSI-9 (see [4]) and NGSI-10 (see [5]) interfaces from the Open Mobile Alliance (OMA) to manage Context Information (for the underlying OMA specifications see [6]). Through these interfaces, a Context Management component will provide its context management services to actors outside of a single network. These actors can:

- provide Context Information (update operations)
- consume Context Information (query and subscribe/notify operations)
- discover context entities through query or notifications (register and discover operations)

The basic elements of the NGSI Context Management Information Model are the following:

- **Entities**
  The central aspect of the NGSI-9/10 information model is the concept of entities. Entities are the virtual representation of all kinds of physical objects in the real world. Examples for physical entities are tables, rooms, or persons. Virtual entities have an identifier and a type. For example, a virtual entity representing a person named “John” could have the identifier “John” and the type “person”.
- **Attributes**
  Any available information about physical entities is expressed in the form of attributes of virtual entities. Attributes have a name and a type as well. For example, the body temperature of John would be represented as an attribute having the name “body_temperature” and the type “temperature”. Values of such attributes are contained in value containers. This kind of container does not only consist of the actual attribute value, but also contains a set of metadata. Metadata is data about data; in in our body temperature example this metadata could represent the time of measurement, the measurement unit, and other information about the attribute value.
- **Attribute Domains**
  There also is a concept of attribute domains in OMA NGSI 9/10. An attribute domain logically groups together a set of attributes. For example, the attribute domain "health_status" could comprise of the attributes "body_temperature" and "blood_pressure". The interfaces defined in NGSI 10 include operations to exchange information about arbitrary sets of entities and their attribute values, while NGSI-9 consists of functions for exchanging information about the availability of information about entities.
- **Context Elements**
  The data structure used for exchanging information about entities is context element. A context element contains information about multiple attributes of one entity. The domain of these attributes can also be specified inside the context element; in this case all provided attribute values have to belong to that domain. Formally, a context element contains the following information
  - an entity id and type
  - a list of triplets <attribute name, attribute type, attribute value> holding information about attributes of the entity
  - (optionally) the name of an attribute domain

- (optionally) a list of triplets <metadata name, metadata type, metadata value> that apply to all attribute values of the given domain

OMA NGSI defines two interfaces for exchanging information based on the information model. The interface OMA NGSI-10 is used for exchanging information about entities and their attribute, i.e., attribute values and metadata. The interface OMA NGSI-9 is used for availability information about entities and their attributes. Here, instead of exchanging attribute values, information about which provider can provide certain attribute values is exchanged.

2.6.3 OSGi

Open Services Gateway Initiative (OSGi) is the most standardized specification for modularization in the Java programming language. Roughly it consists of a runtime container and modules. The modules are deployable in the runtime container. Each module has a description with it public packages and dependencies on other modules. The runtime container enforces module class loader isolation.

2.6.4 Dependency Injection

Dependency Injection provides a mechanism to allow greater flexibility in developing and deploying systems that are comprised of multiple interacting components. The protocol provides the ability to annotate a component's dependency on other system components or components that might be external to the system. This allows greater flexibility to modify components that other components rely on or to reconfigure what components provide what services within the system.

2.6.5 JAX–RS

The JAX-RS API provides developers with the ability to quickly develop Java based Web applications and components that uiltise RESTful APIs. The JAX-RS API provides a high level easy-to-use API for developers to write RESTful web services independent of the underlying technology. The API reduces the development effort in utilising RESTful Web services using the Java Platform by reducing or eliminating the requirement of using low-level APIs like Java Servlets (see Annex 1 for more information on Java Servlets).

2.6.6 JAX–WS

The JAX-WS API is the de facto standard API for building Web Services in Java. It provides a standardised facility for webservice publication and invocation using using XML as the basis for communication between Web-based services and clients. SOAP is an example of an XML based message format and WSDL uses an XML format for describing network services. The JAX-WS API makes enables developers to easily utilise XML based Web Services frameworks.

2.7 Security

The FIspace platform aims to provide secure and reliable exchange of confidential business information and transactions using secure authentication and authorization methods that meet required levels of se-

curity assurance. Key areas to be addressed are authentication, access control and secure communications.

Well-established standards (see Annex 1 for their description) exist in each of these areas. They are

- for authentification
  - SAML
  - OpenID
  - OAuth
- for access control
  - XACML
  - LDAP
- for secure communications
  - S/MIME
  - TLS

2.8 GS1 System Architecture Considerations

The business needs of supply chain participants determine the types of interfaces definitions that require standardisation, and in particular lead to a portfolio of standards that are concerned with *identification* of real-world entities (both physical and virtual), *capture* of identification and other data from physical objects, and *sharing* of information concerning real-world entities among the participants of the supply chain [7].

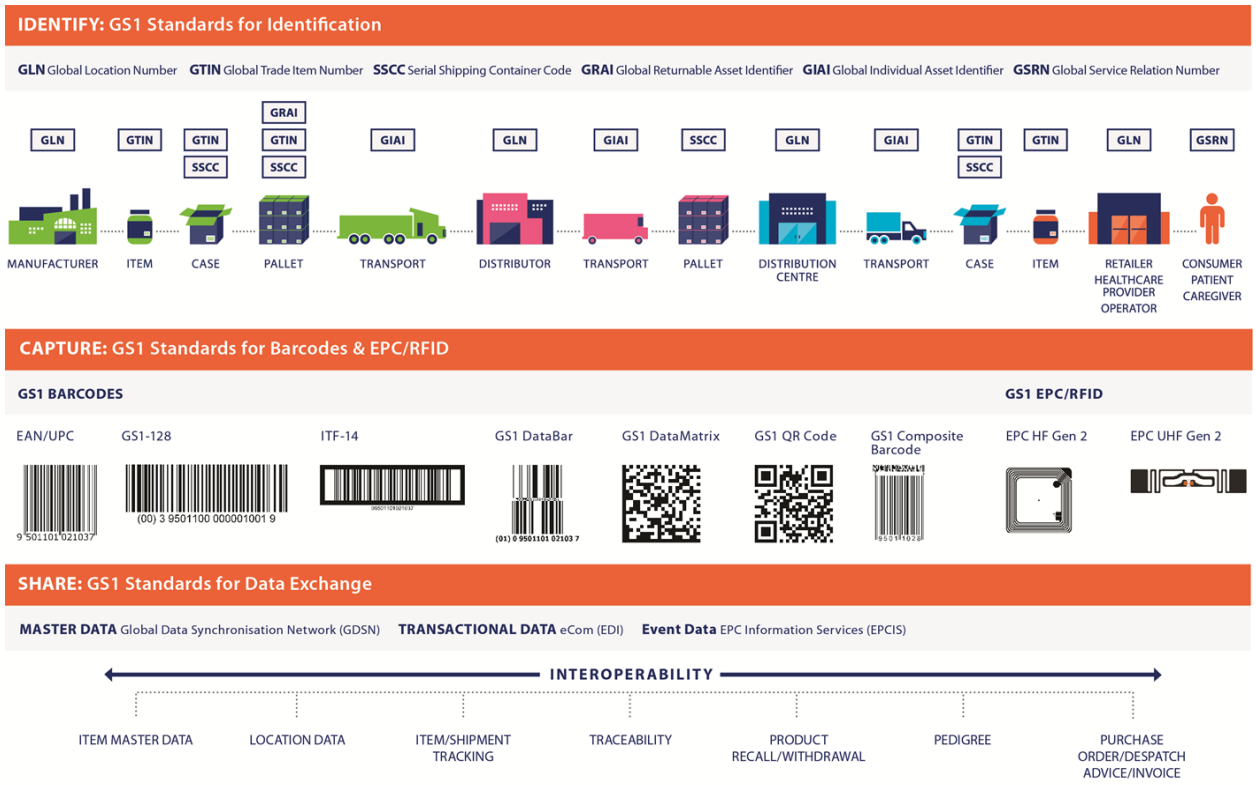


Figure 3: Identify – Capture – Share approach illustrated for the GS1 System

Supply chains can be divided into open and closed ones.

- Open Supply Chains

  An open supply chain is one in which the complete set of trading partners is not known in advance and which changes continually. This has great significance for the architecture of information systems.

  Open supply chains require that interfaces be negotiated and implemented outside the context of any particular trading relationship, and be adhered to by all parties so that interoperability will be achieved despite the fact that the users on each side of the interface are not able to negotiate in advance. It leads to the definition of broadly accepted standards, in which the emphasis is placed on interoperability, maximum applicability to a broad range of business contexts, and minimisation of choices that require pre-coordination between interfacing parties. These are precisely the principles that underlie GS1 Standards and the Semantic Web standards.

- Closed Supply Chains

  In a closed supply chain, a fixed universe of partners is known in advance, and so interfaces can be negotiated in a controlled, coordinated way, and change management is simplified because all parties can agree to make changes simultaneously.

The growth of global supply chains with an ever greater set of potential actors, across many countries and jurisdictions, makes the need for standards for open supply chains ever more pressing. There are a very large number of suppliers, traders, aggregators and transporters in the food and agriculture supply chain, and flexible and dynamic open supply chains depend on interoperability.

2.8.1 Entity Identification

Standards in entity identification provide the means to identify real-world entities so that they may be a subject of electronic information that is stored and/or communicated by end users. An entity may be:

- Physical: A tangible object in the real world made of matter. In particular, a physical object is something to which a physical data carrier (bar code or RFID tag) may be physically affixed.
- Abstract: A virtual object or process, including legal abstractions (e. g. a party), business abstractions (e. g. a class of a trade item) and so on.

An attribute is a piece of information associated with an entity.

Information systems refer to a specific entity by means of a key. It serves to uniquely identify that entity, within some specific domain of entities. An information system uses a key as a proxy for the entity itself.

A primary key is a simple or compound attribute that is used consistently by an application as the key to refer to an entity within a specific domain.

2.8.1.1 GS1 Identification Keys

A GS1 Identification Key is an identifier defined by the GS1 Standards that is usable and intended as to refer to a specific business entity. All GS1 Keys are

- Unique
- Non-significant
- International: GS1 Identification Keys may be used in all countries and all sectors
- Secure: GS1 Identification Keys have a defined structure and most include Check Digits

2.8.1.2 Identifier Syntax: 'Plain', GS1 Element String, Electronic Product Code (EPC)

When a GS1 Identification Key or other identifier is used in an information system, it is necessarily represented using a specific concrete syntax. The syntax that is used may depend on the medium in which the identifier exists; for example, an XML message is text-oriented, while a memory of an RFID tag is binary-oriented. The syntax may also depend upon the context in which the identifier appears.

GS1 Standards provide three different syntaxes for identifiers that support progressively broader application contexts:

- "Plain"

  This syntax is just the GS1 Identification Key with no additional characters or syntax features (see examples in Annex 1 regarding GTIN, GRAI, GLN, SSCC, etc.).

- GS1 Element String

  This syntax consists of a short (2-4 character) "application identifier" that indicates what type of GS1 Identification Key follows, followed by the key itself (see examples of section "Application Identifier System").

- Electronic Product Code (EPC) URI

  This syntax is an Internet Uniform Resource Identifier (URI), specifically a Uniform Resource Name (URN) beginning with urn:epc:id:... and the remainder having a syntax defined by the GS1 EPC Tag Data Standard (see also Section 2.8.1.3 on EPC below)

2.8.1.3 GS1 Application Identifier System

GS1 Application Identifiers are used in all symbologies encoding data beyond GTIN. Every data element is denominated by an Application Identifier stating the content and the structure of the information. For example the batch/lot number is announced by AI 10 and may have up to 20 alphanumeric characters. The BBD is announced by AI 15. Its structure is YYMMDD (YearYearMonthMonthDayDay). For certain data demanded by law such as the eartag number (see section 3.1.1) have dedicated data elements. Further examples can be derived from Annex 1 on the Electronic Product Code (EPC)

An EPC is a unique, individual meta key for different types of business objects (i. e. instances of articles, returnable transport items (RTI), shipments, etc.). The EPC is used in information systems that need to track or otherwise refer to business objects. Large subsets of applications that use the EPC rely upon

RFID Tags as a data carrier. However, it is vital to understand that RFID is not necessarily needed in order to utilize the EPC standard(s).

The following table displays four of the most relevant EPC schemes for the product domain as specified in the EPC Tag Data Standard: Serialized Global Trade Item Number (SGTIN), Serial Shipping Container Code (SSCC), Global Returnable Asset Identifier (GRAI), and Global Location Number with optional extension (SGLN). Apart from that, the EPC can also be used to identify service relations (patients, e.g.), documents (certificates, tenders, eCoupons, etc.) as well as assets (farm machines, etc.).

Depending on its field of application, an EPC can be displayed in three different forms:

- EPC Pure Identity (as used in application systems and EPCIS)
- EPC Tag URI (as used in RFID middleware systems)
- EPC binary code (as used on RFID transponders)

The latter two are related to RFID only. Thus, the most important EPC representation related to FIspace is the EPC Pure Identity one. Table 1 below also displays that an EPC can be converted into the corresponding (serialized) GS1 key and vice versa. Taking the example of the GTIN (Application Identifier '01') and a serial number (AI '21'), the figure displays the conversion process into the three different (SGTIN) EPC representation forms as indicated above.

EPC scheme	Area of application	Example (URI form)
SGTIN	Trade items	urn:epc:id:sgtin:4012345.066666.12345
SSCC	Shipments; logistics unit loads	urn:epc:id:sscc:4012345.1234567891
GRAI	Returnable/ reusable items	urn:epc:id:grai:4012345.77777.678
SGLN	Locations	urn:epc:id:sgln:4012345.66666.5

Table 1: Examples for EPC schemes

Whereas the EPC schemes mentioned above refer to serialized GS1 Keys a Iot related identification of GTIN as another identifier displayable as EPC, that is very important in the field of food and food ingredients, is under development.

2.8.2 Data Capture

The capture standards in the GS1 System are standards for automatically capturing identifying information and possibly other data that is associated with a physical object. The industry term Automatic Identification and Data Capture (AIDC) is sometimes used to refer to the standards in this group.

A principle of GS1 Standards is that data elements are defined in a data carrier neutral way so that their semantics is the same regardless of what data carrier is used to affix them to a physical object (and also the same outside of a physical data carrier, such as an electronic message).

One can differentiate the following:

- Application Data

These are GS1 data elements defined in a data-carrier neutral way. A business application sees the same data regardless of which type of data carrier is used.

- Transfer Encoding

  This is the representation of data used in the interface between a capturing application and the hardware device that interacts with the data carrier (bar code scanner or RFID interrogator). The transfer encoding provides access to control information and carrier information and therefore is different for different data carrier types.

- Carrier Internal Representation

  This is the representation of data in the data carrier itself. In a bar code, this is the pattern of light and dark bars or squares. In an RFID tag, this is the binary data stored in the digital memory of the RFID chip.

The GS1 System provides several types of bar code for use by GS1 members depending on the application. The reasons for this vary because different bar code types have different strengths. GS1 selects the bar code that best fits the application. The bar codes used by GS1 include EAN/UPC, GS1 DataBar, GS1-128, ITF-14, GS1 DataMatrix, Composite Component and GS1 QR Code.

In the food sector EAN/UPC and from 2014 GS1 DataBar are deployed in an open environment. Whereas EAN/UPC barcodes, encoding GTIN and Restricted Circulation Numbers only have been used on consumer units for decades GS1 DataBar encodes additional information such as a batch, best before date, serial number or net weight.

On logistic units GS1-128 is commonly used encoding data such as GTIN, gross weight, net weight, batch, best before date or the SSCC (see example in Annex 1). Especially in logistic environments GS1 Keys can be encoded in an EPC/RFID tag.

In the context of internet applications these data carrier standards become increasingly important for mobile applications (e. g. reading of a barcode by creating an image of the code via smart phone camera, decoding the information and executing defined actions accordingly).

Examples for such mobile applications are ensuring trusted source of data, mobile couponing and mobile payment.

2.8.3 Data Exchange

The use of standards in data exchange provides a predictable structure of electronic business messages, enabling business partners to communicate business data in an automated way, efficiently and accurately, irrespective of their internal hardware or software types. Business partners do not have to align the format and structure of messages - they can use the standard, readily-available format instead.

GS1 communication standards are part of the wide portfolio of standards and solutions that form the GS1 System and are integrated with other supply chain management tools offered by GS1 [8]. GS1 Identification Keys are used in all GS1 Standards, both in physical flow of goods and in information flow of business data and also in XML messages for the identification of (see also Annex 1):

- products (GTIN) [9],
- locations and parties (buyer, seller, and any third parties involved in the transaction, GLN [10]), and
- logistic units (SSCC) [11],

The same identification keys are bar coded on the product packaging in the retail (at the trade item or case level) and on logistic units. Bar code scanning provides direct access to information received in electronic messages (e.g. Order or Despatch Advice) in the data base, without any additional mapping and complex cross-referencing.

Within the FIspace project distinction is made between internal and external data exchange. Whereas in FIspace communication is based on FI technology there will always be classical communication technology in place based on existing business relations.

2.8.3.1 EPCIS

EPCIS (Electronic Product Code Information Services) [12] is a standard for the capture and exchange of visibility data of objects identified with an EPC (Electronic Product Code). Examples for objects relevant for the agri-food sector encompass products, animals, shipments, documents, locations, returnable transport items as well as assets. It is important to comprehend that EPCIS is data carrier agnostic. Thus, EPCIS does not necessarily require RFID technology. Objects referred to are identified by an EPC.

Each time an EPC is read, an event is generated containing fine-grain visibility data encompassing four dimensions: *what* (uniquely identified objects), *where* (location and read point), *when* (time of event) and *why* (status and business process).

The events are stored in decentralized databases (EPCIS repositories). An EPCIS repository has a capture interface for storing as well as a query interface for retrieving event data. The transfer of data via the capture interface is via HTTP, the query interface uses SOAP, XML over AS2 and XML over HTTP(S). All message protocols must be able to use authentication and authorization.

Apart from the Object Name Service (ONS) and EPC Discovery Services, EPCIS is the most important of the three major components of the EPCglobal network. Their interaction is as follows: The ONS (Object Name Service) can be used to provide a lookup service for delivering the network address (URL) of an EPCIS system. In contrast to that, the EPC Discovery Services will serve as a search engine for obtaining information about specific EPCs. However, the latter is still under development.

EPCIS served as the basic communication backbone in the three SmartAgriFood pilots “Fresh Food and Vegetables, “Plants and Flowers” and “Tracking & Tracing for Awareness of Meat” [13] and will continue to form part of trials in FIspace as well.

A complete free and open source implementation of the EPCIS specification including repository as well as query/ capture clients and interfaces – Fosstrak – has been developed by the Auto-ID Labs [14] (see example in Annex 1).

The following four event types are already specified: aggregation event, object event, quantity event, and transaction event. A new fifths event type is currently under development. It is the transformation event which is usable for instance in case of irreversible processes like producing or deboning.

Enterprises commonly use “message bus” technology for interconnection of different distributed system components (see section 2.4). A message bus provides a reliable channel for in-order delivery of messages from a sender to a receiver. (The relationship between sender and receiver may be point-to-point (a message “queue”) or one-to-many via a publish/subscribe mechanism (a message “topic”).) A “message-queue type binding” of the EPCIS Capture Interface would simply be the designation of a particular message bus channel for the purpose of delivering EPCIS events from an EPCIS Capture Application to an EPCIS Repository, or to an EPCIS Accessing Application by way of the EPCIS Query Callback Interface. Each message would have a payload containing one or more EPCIS events (serialized through some binding at the Data Definition Layer; e.g., an XML binding). In such a binding, therefore, each transmission/delivery of a message corresponds to a single “capture” operation.

2.8.3.2 EDI

EDI stands for Electronic Data Interchange and allows rapid, efficient and accurate automatic electronic transmission of agreed business data between trading partners. GS1 mainly provides two different types of EDI languages:

- GS1 EANCOM®
- GS1 XML Interface Description

Both are implemented in parallel by different users. Although XML is a newer technology than the EANCOM®, the latter has a large and constantly growing number of users. GS1 is going to continue supporting both syntaxes for as long as it is necessary. Any new developments, however (e.g. messages for sectors new to GS1) by default will be done only in GS1 XML. New EANCOM® developments in new domains will be done only if there is a justified business reason.

For further information on EANCOM® and GS1 XML, please see Annex 1.

2.8.4 Business Process Standards and Certification Programmes

2.8.4.1 Traceability & Recall

Food safety, product tracing, and product recalls are currently at the forefront of both government regulations and industry concerns around the world. Companies are facing numerous track and trace requirements, not always easily reconcilable. Moreover technology offers various ways of achieving traceability and many solutions exist for national, regional and global supply chain participants. As a result, to facilitate trade today, it is necessary to implement international standards and ensure interoperability of traceability systems.

GS1 offers traceability and recall standards and support their implementation to enhance companies’ business processes supporting visibility, quality and safety in the supply chain.

- The GS1 Traceability Standard [15] defines business rules and minimum requirements to be followed when designing and implementing a traceability system. They are clustered around a matrix of roles and responsibilities for each step of the traceability process.

  - The **Plan and Organize** sub-process determines how to assign, collect, share and keep traceability data. Furthermore, it determines how to manage links between inputs, internal processes, and outputs.
  - The **Align Master Data** sub -process determines how to assign identifications to the parties and physical locations, trade items and if appropriate to assets. It also determines how to exchange Master Data with trading partners.
  - The **Record Traceability Data** sub-process determines how to assign, apply and capture traceable items identification and how to collect, share and store traceability data during the physical flow.
  - The **Request Trace** sub-process determines how to initiate and respond to a traceability request.
  - The **Use Information** sub-process enables the use of the previous processes to take appropriate action as required by legal and business requirements.
- The GS1 Product Recall standard [16] serves as a common-sense blueprint enabling all supply chain stakeholders to implement more effective product recall processes and notifications. The standard defines, standardizes and harmonizes the critical attributes to be captured and shared among trading parties and regulators during a product recall alerting and messaging process. It is a GS1 XML business message standard.

2.8.4.2 Business Processes Modelling Standards

Other standards will be utilised to support business processes within the Business Collaboration Module (BCM) of the FIspace platform. For business processes there are a few standards, maybe the most popular is BPMN (see [17]), which provides businesses with the capability of understanding their internal business procedures using a graphical notation. In FIspace it is likely that a data-centric approach which is a different programming model will be uitilised. Though this approach doesn't yet have an associated standard approach the underlying model for the Case Management Model and Notation (CMMN) proposed standard from OMG (see [18]) currently in beta provides a basis for the approach that will be utilise in FIspace for developing the BCM.

3 Sector-specific Standards

In contrast to multi-industry standards, sector-specific standards lead immanentely to a higher variety of single shapes of standards (at least one per sector). As the identification of an item is essential for almost any process step the industry-specific identification specifications form major part of this chapter (see Chapter 3.1).

Chapter 3 is completed by some additional sector-specific aspects of relevance in certain business process steps regarding electronic data interchange, labelling and certification.

3.1 Industry-specific identification specifications

To the greatest extent possible, the components of the GS1 System are designed to be broadly applicable across all sectors and geographic regions. However, there are often needs that exist only within a particular industry but where all industry participants must still rely upon normative standards in order to achieve interoperability and/or economies of scale in meeting those needs. Such sector specific or regional standards are either used on their own or in case of a seamless integration with global standards by mapping with global/cross-sector solutions. The sector specific standards described in this document refer mainly to information needed in the food chain (“from farm to fork”) as will be used in the related FIspace trials (Trial 432 Fruit & Vegetables Quality Assurance, Trial 441 Meat Information Provenance, Trial 431 Fish Distribution and (Re-) Planning).

3.1.1 Animal identification

The identification of farm animals within Europe is regulated by European legislation. The main goal of these legal regulations is to prevent the spread of diseases and to ensure food safety. In addition, the individual identification of animals gives the farmer access to indivual performance data and enables him to implement an effective herd management. The regulations apply for cattle, pigs, ovines and caprine animals (goats, sheep, etc.) and for horses and other equine animals. The latter are not considered in this document, as they are not used as food in most countries. Examples are given for those countries which are most relevant for livestock farming of the respective animal [19]. Information for each member state can be found at [20].

The main methods of animal identification are RFID transponders and eartags with a human-readable number and/or a barcode. The use of RFID transponders has been analyzed in [21]. LF (low frequency, < 135 kHz) identification has been found suitable for animal application because LF signals are not influenced by body tissue and the achievable reading distances meet requirements. Both E-ear tags and boluses can be used as transponders for ruminants (such as cows). Electronic ruminal bolus transponders are transponders placed into a high specific gravity container able to be orally administered to ruminants, which remain permanently in the fore stomach. The use of injectable transponders is considered to be problematic in relation to the slaughterhouse recovery.

3.1.1.1 The international standard "ISO 11784: Radio frequency identification of animals - code structure" describes the code structure of the transponders (for details, see Annex 1).Identification and registration of pigs

Pigs must be **identified** and **registered** so as to trace the original or transit holding, centre or organisation for intra-Community trade or the movements of animals within national territory [22]. This implies that pigs are not individually identified, but can only be traced back to their holding respectively last stage of fattening. For details, see Annex 1.

3.1.1.2 Identification and registration of ovines and caprines

A system of individual traceability enables each ovine and caprine animal to be traced from the moment of birth and through any intra-Community trade in which it is involved.

All ovine and caprine animals born on holdings within the Community [23] are identified by:

a **first means of identification**, consisting of an electronic transponder or an eartag approved by the competent national authorities, affixed to one ear, made of non-degradable material, tamper-proof and easy to read throughout the lifetime of the animal without causing it any discomfort. The mark must be easily visible at a distance and include a code for the Member State where the holding is located and an individual code;

a **second additional means of identification**, which can be an electronic transponder for animals that already have an eartag as a first means of identification an eartag, a mark on the pastern or a tattoo for animals that already have an electronic transponder as the first means of identification (the tattoo may not be used for animals involved in intra-Community trade).

The code structure for eartags can be looked up in Annex 1.

3.1.1.3 Identification and labelling of bovine animals

In the wake of the crisis over mad cow disease (or bovine spongiform encephalopathy), the European Union has established a system of identification and labelling of beef and veal since the year 2000 [24].

Every Member State must set up a cattle identification and registration system. This system must comprise the following elements [25]:

- ear tags to identify animals individually;
- computerised databases;
- animal passports;
- individual registers kept on each holding.

In Annex 1, detailed examples are given for implementations in France, Germany and UK.

3.1.1.4 Identification of fish

The new European fisheries control system aims at ensuring compliance with the rules of the common fisheries policy (CFP) throughout the production chain – i.e. from the boat to the retailer [26]. Member States are to carry out inspections of activities throughout the production chain for fishery products, in particular landing, processing, transport and marketing. The use of modern inspection technologies such

as the Satellite-based Vessel Monitoring System (VMS), electronic logbooks and the electronic notification of catch data has been extended. The collection, processing and analysis of fishing data have been considerably enhanced. A systematic catch weighing system has been introduced. A new system of traceability for fishery products will allow fishery products to be monitored from the vessel to the retailer.

According to Article 14 of Council Regulation (EC) No 1224/2009 [27] a fishing logbook has to be kept by the master of each fishing vessel. The containing data is illustrated in Annex 1.

3.1.2 Identification of holdings

European regulations have also established a system for the identification of farms and other agricultural enterprises.

In Germany, all holdings keeping farm animals (cattle, pigs, sheep, poultry etc.) have to register at the local veterinary office [28]. The farm is then provided with a 12-digit registration number (2 digits: federal state, 3: administrative district, 3: municipality, 4: ID). Similar regulations apply in other countries, but the format of the registration number will be be different.

In England, CPH numbers are used to identify agricultural holding(s) and any premises where cattle, sheep, goats and pigs are kept. A County Parish Holding number (CPH) is a 9 digit number [29]. The first 2 digits relate to the county the animals are kept in, the next 3 digits relate to the parish and the last 4 digits are a unique number to the keeper.

For laying hens, each production site is registered and has a unique identification number according to Commission Directive 2002/4/EC of 30 January 2002 on the registration of establishments keeping laying hens [30]. The identification number contains one digit for the farming method (eg caged, barn, free range or organic). The register holds information on the name and address of the establishment, maximum capacities, the name and address of the keeper and owner if different, and the registration number of any other establishment kept or owned by the keeper and owner.

In addition to these regulations, EU Member States' competent authorities must give approval to establishments handling, preparing or producing products of animal origin for which requirements are laid down in Regulation (EC) 853/2004. Member States are required to publish lists of these establishments under sixteen separate sections each corresponding to a different food sector [31]. The approval number of the facility has to be given on the packaging [32].

3.1.3 Field identification

Each agricultural area within the EU can be identified by the Land Parcel Identification System which is part of the Integrated Administration and Control System (IACS).

IACS is the most important system for the management and control of payments to farmers made by the Member States in application of the Common Agricultural Policy [33]. It provides for a uniform basis for controls and, among other requirements, it covers the administrative and on-the-spot controls of applications and the IT system which supports the national administration in carrying out their functions.

IACS is operated in the Member States by accredited paying agencies. It covers all direct payment support schemes as well as certain rural development measures. Furthermore, it is also used to manage the controls put in place to ensure that the requirements and standards under the cross-compliance provisions are respected.

The legal requirements concerning IACS are laid down in Council Regulation (EC) No 73/2009 [34] establishing common rules for direct support schemes for farmers and in Commission Regulation (EC) No 1122/2009 [35] laying down the implementing rules.

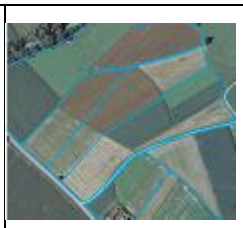
In physical terms, IACS consists of a number of computerized and interconnected databases which are used to receive and process aid applications and respective data. Thus it provides for:

- a unique identification system for farmers;
- an identification system covering all agricultural areas called Land Parcel Identification System (LPIS);
- an identification system for payment entitlements;
- a system for identification and registration of animals (in Member States where animal-based measures apply).

The system ensures a unique identification of each farmer as well as of all agricultural parcels of land and, if needed, of animals. The system covers also the processing of the aid applications.

The approaches used for the creation of the LPIS differs from country to country [36]. They depend on the reference parcel defined, the reference data available (orthophoto, cadastre, topomaps), local agriculture conditions, etc.

The statement of Art 6.1 of the Comm Reg No 796/2004 [37] created the opening for a diverse practice among Member States of 'reference parcel' (RP) representations as: **Cadastral parcel** (CP), **Agricultural parcel** (AP), **Farmers' block/ilot** (FB) and **Physical block** (PB). The cadastral parcel is based on ownership, whilst the other LPIS reference parcels are based on land cover delineated by topographical boundaries and/or agricultural land use. The latter representations (see [38]) correspond either directly to single Agricultural parcel or indirectly to an association of one or more agricultural parcels into 'blocks' according to production pattern or physical (topographic) boundaries of agricultural land use.

Examples				
Type of RP	**= Agricultural parcel**	**< Farmer block/ilot**	**< Physical block**	**Cadastral parcel**
land use for aid scheme	one single crop group	one or several crop groups	one or several crop groups	do not match agricultural pattern
applicants	single farmer	single farmer	one or several farm-	one or several farmers

			ers	
temporal aspect	annual	multi-annual	semi-permanent	land tenure cycle
main data source	farmer's application	farmer's survey	administration survey	land register/cadastre

Table 2: Types of reference parcels

In most member states, but not e.g. in Germany and the Netherlands, LPIS data are available to external organisations or users [36]. The indivual land parcel identifier could therefore be used to trace the origin of field crops. In Germany, the "Flächenidentifikator" (FLIK, parcel unique identifier) consists of 16 alpha-numeric characters [39].

3.1.4 Geo data

The EU member states have to provide their IACS reference parcels in the Geography Markup Language (GML) format [40]. GML is also the default encoding for INSPIRE [41]. The INSPIRE (Infrastructure for Spatial Information in the European Community) directive aims to create a European Union (EU) spatial data infrastructure. This will enable the sharing of environmental spatial information among public sector organisations and better facilitate public access to spatial information across Europe [42].

The GML is an XML grammar defined by the Open Geospatial Consortium (OGC) for expressing geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. As with most XML based grammars, there are two parts to the grammar – the schema that describes the document and the instance document that contains the actual data. A GML document is described using a GML Schema. This allows users and developers to describe generic geographic data sets that contain points, lines and polygons [38]. It is also an ISO standard [43].

ISO 19136:2007 defines the XML Schema syntax, mechanisms and conventions that:

- provide an open, vendor-neutral framework for the description of geospatial application schemas for the transport and storage of geographic information in XML;
- allow profiles that support proper subsets of GML framework descriptive capabilities;
- support the description of geospatial application schemas for specialized domains and information communities;
- enable the creation and maintenance of linked geographic application schemas and datasets;
- support the storage and transport of application schemas and data sets;
- increase the ability of organizations to share geographic application schemas and the information they describe.

3.2 Industry-specific Communication Protocols

The electronic communication between implements, tractors and computers is standardized in the international ISO 11783 standard – "Tractors and machinery for agriculture and forestry - Serial control and communications data network" [44]. Its parts specify all layers of communication: the pyhsical layer, data

link layer, network layer, management of addresses, message application layer, task-controller applications layer, the message format, virtual terminal, tractor electronic control unit and the file server. ISO 11783-11:2011 provides a detailed Data Dictionary. It is the basis for precision farming and enables steering of different farming devices from variable companies.

3.3 Global G.A.P.

“GLOBALG.A.P. is a private sector body that sets voluntary standards for the certification of production processes of agricultural (including aquaculture) products around the globe.

The GLOBALG.A.P. Standard is primarily designed to reassure consumers about how food is produced on the farm by minimising detrimental environmental impacts of farming operations, reducing the use of chemical inputs and ensuring a responsible approach to worker health and safety as well as animal welfare.” [45].

3.4 Orgainvent

Similar to Global G.A.P. Orgainvent is a private sector body and the largest German labelling organisation in the beef sector. In addition to this officially approved labelling system, they also provide voluntary proof of origin for pork and poultry meats. Information about this can be found in the "Labelling Systems / Traceability" section [46].

4 Semantic Web Technologies

4.1 Introduction

This section discusses on the relevance, applicability and potential of Semantic Web technologies, specifically those technologies which have been developed by Tim Berners-Lee and his colleagues since the late 1990s, [48], [49], for the B2B scenarios in FIspace and in particular the agri-food sector including logistics. Semantic technologies are concerned with the integration and interoperability of data. There has already been extensive use of semantic technologies in intra-organisational scenarios [50], and there is growing awareness of the applicability of this technology stack in inter-organisational data integration [51]. The fundamental idea behind the Semantic Web has been to create a 'web of data', of machine readable data in parallel to the 'web of documents' which already exists. Berners-Lee's vision is a world where vast quantities of data are published by the public and other organisations, where the format follows certain standard rules, and uses a variety of standard vocabularies or 'ontologies', and where different, disparate data can be interlinked to answer questions and to connect both people and knowledge in a manner that was impossible before. For the technical background on the semantic technology stack, please refer to [52], [53], [54], [55].

As is clear from a wide range of research, including the deliverables developed in SmartAgriFood, in the agri-food sector, information and knowledge in any form is a "high cost" item, i.e. costly to gather by any actor, costly to communicate, costly to certify and thereby provide assurance to subsequent actors on the supply chain, and costly to deliver to the information user. Semantic technologies provide a fundamental approach, and a set of solutions that can address many of the challenges arising from the complexity and heterogeneity of the agri-food supply chain. There are a number of existing and successful standards focussed largely on product identification (i.e. GS1 standards cf. Chapter 2.8.1 above). The GS1 standards could form in combination with onthologies (see section 4.3) an important brick to make significant inroads to the challenge of interoperability (and 4.4). Semantic technologies provide a set of ad hoc standards for data capture, for data publishing and for data consumption. They provide flexibility, rapid adaptation to new requirements, and greater functionalities.

There are already a number of excellent examples where semantic technologies have played a major role in inter-enterprise data integration. The most widely cited one is the use made by the UK's BBC of semantic technologies to link data across different departments and also to sub-scribe to external data (cf. [56]). For example, all BBC music content is consumed from Musicbrainz and Wikipedia, and BBC editors edit these websites rather than any internal BBC site. Equally, BBC Nature derives a large part of its content from external websites such as WWF Wildfinder and the Animal Diversity Website. Another example is the European project which integrates descriptive metadata from hundreds of museums and galleries around Europe marked up with standard vocabularies and ontologies (cf. [57]). In the supply chain arena, companies like Cambridge Semantics are offering solutions for pharmaceutical companies to track and manage their supply chain in a scalable integrated manner.

Semantic technologies could not have an impact on business and society, on the world in general, unless there is sufficient uptake and adoption. A sufficient number of people need to be using the technologies and above all publishing data so that the technology is useful. To this end Berners-Lee proposed a set of principles to facilitate the publishing of data on the web so that a global data space would be created, the so called “Linked Data principles”. Linked Data is data published on the web that is “is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets” [58]. Ordinary links in web pages allow different websites to be connected. Links in Linked Data allow formal statements to be made that link arbitrary things in the world. Linked Data is “a style of publishing structured data on the Web in which all elements of an ontology (properties, classes, and value vocabularies), as well as things described by the ontology (publications, events, people), are identified by Uniform Resource Identifiers (URIs), allowing data to be extensively cross-referenced (“linked”) with other data sources” [59].

Following the declaration of the principles, there has been an explosion of data available on the web. The Linking Open Data project brought researchers, universities and small companies together, and these were quickly followed by large organisations such as the BBC, Thompson Reuters and the Library of Congress. One of the key developments was the transformation of the publicly edited Wikipedia into a structured set of data called DBpedia [60]. This has acted as a central hub to which many other data sets link, a dizzying array of different types of data mostly freely available for organisations and people to use. Data has been added concerning geographical locations, scientific publications, music, programmes on television and radio, all kinds of life science data including proteins, genes, metabolic pathways, drugs and clinical trials, political and historical data and statistical and census data. The FAO ontology AGROVOC has also been integrated (cf. same chapter below).

Semantic technologies address key challenges in the agri-food sector. Here we enumerate some of these challenges and discuss the corresponding potential solution:

- Heterogeneity of actors. Different actors along the supply chain can publish data using standard vocabularies, and only publish the data they feel comfortable with making public. The low cost of entry for publishing data makes this an option available to small and medium size enterprises at all stages of the supply chain.
- Heterogeneity of data. Different actors will publish data using different vocabularies but the ability to create mapping ontologies makes the integration of disparate data perfectly feasible.
- Integration of intra-organisation data with external data. Recent developments such as AgroXML make the integration of on-farm data with farm machinery and sensors a near future reality. External data e.g. from meteorological services that provide data following semantic standards can be seamlessly integrated.
- Lack of supply chain integration. One of the key challenges in agri-food remains the lack of supply chain integration. Semantic technologies are specifically designed for data integration, and thus provide the fundamental glue for achieving this goal.

In this approach of the supply chain (or rather supply web), every actor can publish linked data about themselves, about the entities they produce (tomatoes), process, transport or sell. Currently Google and

other search engines encourage websites to include “rich snippets”, an initiative that has now metamorphosed into the http://www.schema.org project, and which enables search engines to “harvest” data for indexing. In a similar manner, we envisage all types of food producers, processors and retailers, to publish data about their products in sufficient detail to be of use to data “consumers”. Data might come from multiple sources, some static, some dynamic. These will include websites, handheld devices, smartphones, sensors, and a variety of smart devices which are increasingly being used along the supply web. Data can be in structured formats following formal vocabularies/ontologies (such as GoodRelations, AgroXML, or Agrovoc) so as to be semantically correct and machine readable. The use of standard vocabularies can ensure both interoperability and absence of ambiguity. Semantic Web services [61] can be provided by third parties to enable data aggregation, reasoning in some cases, and publishing in specific formats e.g. semantic mash-ups or end-user smartphone apps. Actors can easily enter the agri-food data market (whether as read world actors or as added-value data processors) allowing small farmers or retailers to easily integrate into the over-arching agri-food supply web. New players would be able to come up with new services. These might include un- foreseen data integration applications or facilities to order products from hyper-local food providers. This would have a number of advantages:

- Data is available to both the “next” and the “previous” actor in the chain, but also to every other interested actor in the supply web.
- Given the use of URIs for the relevant level of granularity (shipment, pallet, package, individual can or tomato) information can be updated appropriately in real time
- The class hierarchies allow the propagation of data through different level e.g. if a pallet of processed tomatoes goes from A to B, then reasoning services can conclude this is true for individual cans of tomatoes.
- External data providers (e.g. certification bodies) can publish their data independently and yet have it successfully integrated into the information flow.
- The flexibility of semantic technologies allows for the addition of new data types easily without the restructuring or redesign of databases and communication protocols. Thus if suddenly there was a need to know (for example) exactly how much water is used in a specific crop, such an additional ‘field’ can be added with almost no overhead.
- The integration of crowd-sourced data (such as that collected by MusicBrainz in the domain of music) becomes possible in an efficient manner.

As noted by Hyland, “Linked Data builds on the Web foundation, but is emphatically geared towards cooperative decentralization” [62] and this is exactly what is needed in the case of the agri-food supply web. The current status of these technologies is that a great deal of the background work has already been undertaken. There exist a number of key ontologies, some of which have already been mentioned above, and there exist a number of enabling tools ranging from triple stores, semantically enabled CMSs (e. g. Drupal), and integration with lower level data sources such as sensors and more generally the Internet of Things [63].

4.2 Technology Standards

Semantic Web Technology standards include a series of WC3C standards covering the following aspects:

- Uniform Resource Identifiers [64]
- Resource Description Framework (RDF) [65]
- RDF Schema (RDFS) [66]
- The Web Ontology Language (OWL) [67] and OWL 2 [68]
- SKOS Simple Knowledge Organization System [69]
- RDFa [70]
- SPARQL query language [71]

For a more technical introduction to the Semantic Technology stack of standards and corresponding technologies, refer to Appendix 2, as well as [52], [53], [54], [55].

There is a large collection of tools and libraries which implement these standards and which can be integrated into the FIspace environment or the applications developed for the FIspace platform. Some are commercial offerings and others are open source. These include:

- Triple Stores i.e. graph based data base systems for storage, retrieval and querying of large RDF data sets, e.g. Apache Jena [72], 4store [73], OWLIM [74], OpenLink Virtuoso [75], Sesame [76], Stardog [77].
- Reasoners i.e. tools for deriving inferences over the data in the triplestores. Research systems include Pellet [78] and FACT++ [79]. Several of the triplestore systems provide reasoning capabilities of different degrees of sophistication (Jena, Stardog, OWLIM, etc.).
- Ontology Editors. The main editor used by the community is Protege [80], although there are others such as Topbraid Composer [81], the Neon Toolkit [81]
- Browsers. Speciality browsers for visualising RDF data also exist including OpenLink Data Explorer (a browser extension [82], Tabulator [83], Sigma [84].

4.3 Data Exchange Standards / Ontologies

In the preceding work undertaken during SmartAgriFood, our main concern was to map the landscape with respect to data exchange standards (as opposed to technology standards in general), and this was largely undertaken in D600.2 of SmartAgriFood (see [1]), where more details can be found for several of the following ontologies. A number of vocabularies or ontologies exist in the agri-food domain with less work having been undertaken in the pure logistics space.

4.3.1 Agriculture

AgroXML/agroRDF (see [85]): agroXML is an XML dialect for representing and describing farm work. It provides elements and XML data types for representing data on work processes on the farm including accompanying operating supplies like fertilisers, pesticides, crops and the like. It can be used within farm management information systems as a file format for documentation purposes but also within web services and interfaces between the farm and external stakeholders as a means to exchange data in a structured, standardised and easy to use way.The main purposes of agroXML and agroRDF are:

- exchange between on-farm systems and external stakeholders
- high level documentation of farming processes
- data integration between different agricultural production branches
- semantic integration between different standards and vocabularies
- a means for standardised provision of data on operating supplies

agroRDF is an accompanying semantic overlay model that is at the moment still under heavy development. Using the resource description framework (RDF) of the W3C, a set of small, modular ontologies called agroRDF based on the agroXML schemas has been created. They currently cover processes and associated data items in agriculture like harvest, seeding, machines, etc. agroRDF has been mapped to the XML schemas using additional schema attributes from the semantic annotations to WSDL and XML schema (SAWSDL) recommendation of the W3C. That way, data sets can be related to the higher level semantic model providing additional information like class-superclass relations, properties etc. that cannot easily be conveyed through the XML schema alone. Wherever possible, classes and properties in agroRDF refer to existing ontologies like the QUDT ontology for quantities and dimensions and the vCard and FOAF ontologies for data on people. A mapping into the AGROVOC thesaurus of the FAO has been created allowing e. g. applications like multilingual search in data sets. A prototypical web service showing application of these semantic web technologies to agricultural machinery data has been created showing flexibility in extensions however covering only a single data source. Recently, work is conducted to build a set of web services for livestock farming that demonstrate integration of several data sources and standards using agroRDF as a facilitator. As such, agroRDF serves a broader purpose than the agroXML schemas alone in that it provides mechanisms for inter-standards-integration and interoperability.

For further details on agroRDF consult SAF D600.1 Section 2.4 [2].

4.3.2 Food Chain

Agrovoc [86]: The AGROVOC thesaurus by the Food and Agricultural Organization of the United Nations (FAO) is nowadays the most comprehensive multilingual thesaurus and vocabulary for agriculture. Originally, it was devised for indexing of literature, but it is increasingly used also in facilitating knowledge sharing and exchange through electronic media and machine-readable data formats. The AGROVOC thesaurus contains more than 40.000 concepts in up to 21 languages covering topics related to food, nutrition, agriculture, fisheries, forestry, environment and other related domains [87].The vocabulary is provided in standard RDF and SKOS and concepts are identified by URLs. Therefore, it is easy to reference these concepts or create mappings to other vocabularies. Apart from several agricultural ontology relations (for a complete list see [88]), AGROVOC uses common thesauri relationships like “broader term”, “narrower term”, “related term”.

The multilingual character of Agrovoc is one of its major advantages particularly for real world applications at the user end. While it is entirely possible to code meta-data from different languages in English (the dominant language for ontologies and formal vocabularies), in the food and agriculture domain it is particularly useful to be able to have access to local language labels for different concepts.

Further details on Agrovoc are available in SAF deliverable D600.2 Section 2.3 [1]

Network of Fisheries Ontologies [89]: This is a network of ontologies developed by the NEON project [90] to cover the fishing domain as this was a pilot area. The ontologies cover species, water areas, aquatic resources, vessel types, gear types, etc. The network integrates a number of links into Agrovoc and other ontologies. Full details are available at the link cited above. The work is marked as draft so the final status of this network would need some evaluation.

4.3.3 Generic standards relevant to the domain

Here we mention some more general ontologies/vocabularies which can be used for the agri-food and logistics domains but actual have wider application. Many are official standards (i.e. produced by an official body) but others are de facto standards (i.e. are widely used with official definition).

Observations & Measurements [91]: This provides general models and schema for supporting the packaging of observations from sensor system and sensor-related processing. The model supports metadata about the Observation, as well as the ability to link to the procedure (i.e. sensors plus processing) that created the observation, thus, providing an indication of the lineage of the measurements.

For further details consult SAF Deliverable D600.2 Section 2.26 [1]

The OM standard has been encoded in XML [92] and is also available in an experimental OWL format [93].

Sensor Model Language (SensorML) [94]: This provides standard models and an XML encoding for describing any process, including the process of measurement by sensors and instructions for deriving higher-level information from observations. Processes described in SensorML are discoverable and executable. All processes define their inputs, outputs, parameters, and method, as well as provide relevant metadata. SensorML models detectors and sensors as processes that convert real phenomena to data. For further details consult SAF Deliverable D600.2 Section 2.29 [1]

Closely related to this standard is the W3C initiative which developed the Sensor and Sensor Network ontology which “answers the need for a domain-independent and end-to-end model for sensing applications by merging sensor-focused (e.g. SensorML), observation-focused (e.g. Observation & Measurement) and system-focused views. It covers the sub-domains which are sensor-specific such as the sensing principles and capabilities and can be used to define how a sensor will perform in a particular context to help characterise the quality of sensed data or to better task sensors in unpredictable environments. Although the ontology leaves the observed domain unspecified, domain semantics, units of measurement, time and time series, and location and mobility ontologies can be easily attached when instantiating the ontology for any particular sensors in a domain. The alignment between the SSN ontology and the DOLCE Ultra Lite upper ontology has helped to normalise the structure of the ontology to assist its use in conjunction with ontologies or linked data resources developed elsewhere.” cf. [95]

The SSN ontology thus can be seen as including the OGS standard and provides a framework for application development.

Geography Markup Language [96] and GeoSPARQL [97]: The OpenGIS® Geography Markup Language Encoding Standard (GML) The Geography Markup Language (GML) is an XML grammar for expressing geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. As with most XML based grammars, there are two parts to the grammar – the schema that describes the document and the instance document that contains the actual data. A GML document is described using a GML Schema. This allows users and developers to describe generic geographic data sets that contain points, lines and polygons.

GeoSPARQL is a standard built upon GML providing for the representation and querying linked geospatial data. It defines a small ontology using GML and WKT, as well as defining a SPARQL query interface. For more details consult the links mentioned above and [98]. GeoSPARQL is assumed to be replacing the Ordinance Survey Vocabulary the UK OS have used until now to encode their geospatial data as linked data.

FOAF (http://xmlns.com/foaf/spec/): “FOAF is a project devoted to linking people and information using the Web. Regardless of whether information is in people's heads, in physical or digital documents, or in the form of factual data, it can be linked. FOAF integrates three kinds of network: social networks of human collaboration, friendship and association; representational networks that describe a simplified view of a cartoon universe in factual terms, and information networks that use Web-based linking to share independently published descriptions of this inter-connected world. FOAF does not compete with socially-oriented Web sites; rather it provides an approach in which different sites can tell different parts of the larger story, and by which users can retain some control over their information in a non-proprietary format.”

FOAF is a widely used standard for describing people and their relationships. It is used in agroRDF.

VCARD Ontology [99]: A widely used ontology for describing addresses for people and organisations. The ontology is compatible with the widely used VCARD format used in address book applications. It is used in agroRDF.

GoodRelations Ontology [100]: Goodrelations is a widely used ontology for the markup of products and organisations offering products on the internet. Its prime focus is e-commerce. The standard approach is to use GoodRelations in RDFa to mark up information in HTML webpages thereby allowing web-crawlers to collect RDF format data for whatever indexing or aggregation purposes. GoodRelations has had considerable success and is used in a number of major websites around the world. The best known examples include the adoption of Goodrelations by the US retailers Bestbuy.com which saw a 30% increase in their search results and a 15% increase in click throughs (Sheldon, 2011). Over 600,000 products were marked up on their website with this ontology in RDFa. Other organisations include O’Reilly books in the US, both Renault and Volkswagen in the UK and Arzneimittel.de in Germany. One of the more interesting projects has involved the markup of nearly all the retail outlets in the German town of Ravensburg and the creation of a corresponding smartphone app with information on address and opening hours. The most common current usage of the GoodRelations ontology is to ensure the best possible Search Engine Optimisation, but the design and intent from the start was much greater.

The GoodRelations Ontology has now become a de facto standard and all retail and consumer facing publishing of data would be well advised to use it as the basis for their mark up and metadata.

4.4 Relationship between Semantic Web Technologies and GS1 standards

Semantic Web standards and linked data technologies can contribute significantly to increase the interoperability among systems and processes in organisations and industries implementing GS1 standards, thereby facilitating increased collaboration and sharing of information between trading partners in end-to-end supply chains. The benefits of their uptake can be realised both within B2B and B2C setups.

The key to achieving interoperability is unambiguous identification of resources to be shared and representation of the resources in a standardised and machine understandable format. GS1 provides a set of information standards for uniformly exchanging the attributes of products – commonly referred to as the product master data. Within B2B settings, master data is currently shared via the data pools in the Global Data Synchronization Network (GDSN) and products identified using Global Trade Identification Number (GTIN) (see 2.8.1.1). With the EPC concept, serialised GTINs can be represented in URN format(e.g. urn:epc:id:sgtin:4012345.066666.12345). Used in the context of self resolving and authoritative HTTP URIs and representing product master data as linked data in one of the several RDF serialisations, not only can the sharing of information among trading partners (B2B) be streamlined with fewer intermediary services, but the master data can be enriched by interlinking it with externally available, open, crowdsourced and authoritative datasets. A subset of this data could also be exposed within consumer facing, i.e., B2C based Web applications using RDFa and Microdata. Several existing vocabularies could be exploited for providing metadata to product descriptions.

GS1 provides GPC bricks with identifiers for very broad classes of products. Currently GPC bricks are not linked with externally available, enriched product catalogues. Linking GPC bricks identified using HTTP URIs, with these catalogues, e.g., via the SKOS vocabulary, will lead to a richer semantic user search experience when combined with linked datasets of product master data. Some other GS1 artefacts that could benefit from the Semantic Web/Linked data paradigm of knowledge representation include the GDD (Global Data Dictionary) [101], especially when enriching it with multilingual support and EPCIS, for representing and sharing event based tracking and tracing information for products on the Web of data.

4.5 Semantic Web Technologies in FIspace

As outlined above, in FIspace significant benefit could be drawn from the combination of semantic standards and GS1 standards. To the extent that applications (apps) on the FIspace platform will make data available to third parties, it will be advisable to have that data in a standard format BOTH syntactically and semantically. This means in practice if it is data whose semantics is specified by GS1 standards these are the ones to follow. If the data points are ones which do not have existing GS1 encodings, then suitable ontologies/vocabularies should be used from the ones mentioned above. This is obviously true on and within the farm where agroRDF should be the accepted standard.

As we have stressed elsewhere (e.g. in SAF Deliverable D100.4 Section 9.7 [2]), the use of semantic technologies is to be seen as a complement to the existing GS1 standards and other standards. Ontolo-

gies can act as standards but can also provide mappings between standards so they can enable flexibility and adaptation to the changes needed as industry, commerce and technology evolve. One example of this is the replacement of the Ordinance Survey vocabulary (for geographical linked data) by the GeoSPARQL standard.

5 Conclusion and Follow-up activities

Within the FIspace project, a report will be prepared for M22 entitled "Recommendations for new or updated Standards". Between now (M4) and then the partners involved in Task 540 will continue an existing dialogue with the Trials undertaken in Work Package 400 so as to closely monitor how and where standards are relevant to those activities. This is especially but not only relevant to the App development in T450. Out of this dialogue, work on the trials will be encouraged to use standards where appropriate, and identify adaptations to existing standards where needed or additional standards where appropriate. It is not the remit of FIspace to develop or establish new standards but we can identify the requirements where appropiate.

Since there is a mutual dependency between FIspace Platform development and recommendation on technical standards co-operation and interaction with WP 200 (FIspace Development) will be continued in future WP 540 deliverables.

It is obvious that many of the standards discussed and identified as most relevant in this report (and the accompanying SmartAgriFood and FInest deliverables) are of importance to the wider Future Internet initiative and the FI-PPP. For this reason, we plan to establish a wider dialogue with the FI-PPP Standards Committee, both to learn from them and to influence any initiatives they may take and ensure they are informed by the needs and requirements of our domain pilots.

In parallel with the core development activity, we are also undertaking outreach activities and supply chain integration research. Included in the former category is the FIspace sponsored tutorial on "Semantic Web and Ontologies in Agri-Food".

At HAICTA, a follow on from the Workshop held on that theme at SmartAgriMatics 2012. In the latter category, is research undertaken to integrate more effectively GS1 technologies with the opportunities offered by Semantic Web technologies (cf. [102] and [103]).

6 References

[1] http://www.smartagrifood.eu/sites/default/files/content-files/downloads/SAF-D600.2-PlanForStandardiation-FINAL-2.pdf
[2] Intendet to be made publically available soon.
[3] http://www.w3.org/TR/soap/
[4] http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_NGSI-9_Open_RESTful_API_Specification
[5] http://forge.fi-http://www.openmobilealliance.org/Technical/release_program/docs/CopyrightClick.aspx?pck=NGSI&file=V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdfware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_NGSI-10_Open_RESTful_API_Specification
[6] http://www.openmobilealliance.org/Technical/release_program/docs/CopyrightClick.aspx?pck=NGSI&file=V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdf
[7] http://www.gs1.org/docs/gsmp/architecture/GS1_System_Architecture.pdf
[8] http://www.gs1.com/barcodes/technical/id_keys
[9] http://www.gs1.com/barcodes/technical/idkeys/gtin
[10] http://www.gs1.com/barcodes/technical/idkeys/gln
[11] http://www.gs1.com/barcodes/technical/idkeys/sscc
[12] EPCglobal Inc. (2007): EPC Information Services (EPCIS) Version 1.0.1 Specification http://www.gs1.org/gsmp/kc/epcglobal/epcis/epcis_1_0_1-standard-20070921.pdf
[13] http://www.smartagrifood.eu/pilots
[14] https://code.google.com/p/fosstrak/wiki/EpcisMain
[15] www.gs1.org/gsmp/kc/traceability
[16] www.gs1.org/gsmp/kc/recall
[17] http://www.bpmn.org/
[18] http://www.omg.org/spec/CMMN/1.0/Beta1/
[19] http://epp.eurostat.ec.europa.eu/portal/page/portal/statistics/search_database: apro_mt_lscatl and apro_mt_lspig
[20] http://ec.europa.eu/food/animal/identification/bovine/id_bovine_animals_en.htm
[21] European Commission Directorate General for Health and Consumers: Study on the introduction of electronic identification (EID) as official method to identify bovine animals within the European Union Final Report, 25.04.2009 http://ec.europa.eu/food/animal/identification/bovine/docs/EID_Bovine_Final_Report_en.pdf
[22] Identification and registration of pigs http://europa.eu/legislation_summaries/food_safety/veterinary_checks_and_food_hygiene/sa0001_en.htm
[23] Identification and registration of ovines and caprines http://europa.eu/legislation_summaries/food_safety/veterinary_checks_and_food_hygiene/l67005_en.htm
[24] Regulation (EC) No 1760/2000 of the European Parliament and of the Council of 17 July 2000 establishing a system for the identification and registration of bovine animals and regarding the labelling of beef and beef products and repealing Council Regulation (EC) No 820/97. (http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32000R1760:EN:NOT)
[25] Identification and labelling of beef and veal http://europa.eu/legislation_summaries/food_safety/veterinary_checks_and_food_hygiene/l12064_en.htm
[26] http://europa.eu/legislation_summaries/maritime_affairs_and_fisheries/fisheries_resources_and_environment/pe0012_en.htm
[27] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32009R1224:DE:NOT
[28] Viehverkehrsverordnung in der Fassung der Bekanntmachung vom 3. März 2010 (BGBl. I S. 203): http://bundesrecht.juris.de/viehverkv_2007/__26.html
[29] http://archive.defra.gov.uk/foodfarm/farmanimal/movements/pigs/documents/new_owner_guide.pdf
[30] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0004:EN:NOT
[31] http://www.food.gov.uk/enforcement/sectorrules/#.Ud6Khm2K4cs
[32] http://en.wikipedia.org/wiki/Health_mark

[33] Integrated Administration and Control System (IACS) http://ec.europa.eu/agriculture/direct-support/iacs/index_en.htm
[34] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32009R0073:en:NOT
[35] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32009R1122:en:NOT
[36] LPIS - Wikicap - European Commission http://marswiki.jrc.ec.europa.eu/wikicap/index.php/LPIS
[37] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32004R0796:EN:NOT
[38] Geography Markup Language http://www.opengeospatial.org/standards/gml
[39] Verordnung über die Durchführung von Stützungsregelungen und des Integrierten Verwaltungs- und Kontrollsystems (http://www.gesetze-im-internet.de/invekosv/index.html#BJNR319410004BJNE000404377)
[40] http://marswiki.jrc.ec.europa.eu/wikicap/index.php/Tools_2012
[41] INSPIRE Infrastructure for Spatial Information in Europe, D2.8.III.9 Data Specification on Agricultural and aquaculture facilities – Draft Technical Guidelines http://inspire.jrc.ec.europa.eu/documents/Data_Specifications/INSPIRE_DataSpecification_AF_v3.0rc3.pdf
[42] http://inspire.jrc.ec.europa.eu/index.cfm
[43] ISO 19136:2007, Geographic information -- Geography Markup Language (GML) http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=32554&commid=54904
[44] ISOBUS standards (ISO 11783); http://dictionary.isobus.net/isobus/
[45] www.globalgap.org
[46] www.orgainvent.de
[47] Berners-Lee, T.; Hendler, J. & Lassila, O., (2001) The Semantic Web, Scientific American, , 30-37, http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21
[48] Shadbolt, N.; Berners-Lee, T. & Hall, W., (2006) The Semantic Web Revisited, IEEE Intelligent Systems, , 21, 96-101, http://doi.ieeecomputersociety.org/10.1109/MIS.2006.62
[49] Berners-Lee, T.; Hall, W.; Hendler, J. A.; O'Hara, K.; Shadbolt, N. & Weitzner, D. J., (2006) A Framework for Web Science, Foundations and Trends in Web Science, , 1, 1-130
[50] Wood, D. (ed.)(2010) Linking Enterprise Data, Springer, http://3roundstones.com/led_book/led-contents.html
[51] Alves, B. & Schumacher, M., (2012) Fairtrace - Tracing the textile industry , Proceedings of the ISWC 2012 Industry Track
[52] Allemang, D. & Hendler, J.,(2011) Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL, Morgan Kaufmann
[53] Hitzler, P.; Krötzsch, M. & Rudolph, S., (2009) Foundations of Semantic Web Technologies, Chapman & Hall/CRC
[54] Segaran, T.; Evans, C. & Taylor, J., (2009) Programming the Semantic Web, O'Reilly
[55] Heath, T. & Bizer, C., (2011) Linked Data: Evolving the Web into a Global Data Space, Morgan Claypool, http://linkeddatabook.com/editions/1.0/
[56] http://www.w3.org/2001/sw/sweo/public/UseCases/BBC/
[57] http://www.w3.org/2001/sw/sweo/public/UseCases/Europeana/
[58] Berners-Lee, T.,(2006) Linked Data - Design Issues, http://www.w3.org/DesignIssues/LinkedData.html
[59] Baker, T. & Keizer, J., (2010) Linked Data for Fighting Global Hunger:Experiences in setting standards for Agricultural Information Management, in Wood, D. (ed.) Linking Enterprise Data, Springer US, , 177-201, http://3roundstones.com/led_book/led-baker-et-al.html
[60] Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R. & Ives, Z., (2007) Aberer, K.; Choi, K.-S.; Noy, N.; Allemang, D.; Lee, K.-I.; Nixon, L.; Golbeck, J.; Mika, P.; Maynard, D.; Mizoguchi, R.; Schreiber, G. & Cudré-Mauroux, P. (ed.)
DBpedia: A Nucleus for a Web of Open Data, The Semantic Web: Proceeding of the International Semantic Web Conference 2007 and the Asian Semantic Web Conference 2007 (ISWC2007+ASWC2007), Springer, , 722-735, http://richard.cyganiak.de/2008/papers/dbpedia-iswc2007.pdf
[61] Cabral, L.; Domingue, J.; Motta, E.; Payne, T. R. & Hakimpour, F., (2004) Bussler, C.; Davies, J.; Fensel, D. & Studer, R. (ed.) Approaches to Semantic Web Services an Overview and Comparisons., ESWS, Springer, , 3053, 225-239, http://springerlink.metapress.com/openurl.asp?genre=article{&}issn=0302-9743{&}volume=3053{&}spage=225
[62] Hyland, B., (2010) Preparing for a Linked Data Enterprise, in Wood, D. (ed.) Linking Enterprise Data, Springer US, , 51-64, http://3roundstones.com/led_book/led-hyland.html
[63] Barnaghi, P.; Wang, W.; Henson, C. & Taylor, K., (2012) Semantics for the Internet of Things: Early Progress and Back to the Future, International Journal on Semantic Web and Information Systems

(IJSWIS), International Journal on Semantic Web and Information Systems (IJSWIS), IGI Global, , 8, 1-21, http://knoesis.org/library/download/IJSWIS_SemIoT.pdf
[64] http://www.w3.org/Addressing/URL/uri-spec.html
[65] http://www.w3.org/RDF/
[66] http://www.w3.org/TR/2004/REC-rdf-schema-20040210/
[67] http://www.w3.org/TR/2004/REC-owl-features-20040210/
[68] http://www.w3.org/TR/2012/REC-owl2-overview-20121211/
[69] http://www.w3.org/TR/2009/REC-skos-reference-20090818/
[70] http://www.w3.org/TR/2012/REC-rdfa-core-20120607/
[71] http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
[72] http://jena.apache.org/
[73] http://www.4store.org/
[74] http://www.ontotext.com/owlim
[75] http://virtuoso.openlinksw.com/
[76] http://www.openrdf.org/
[77] http://stardog.com/
[78] http://clarkparsia.com/pellet
[79] http://owl.man.ac.uk/factplusplus/
[80] http://www.topquadrant.com/products/TB_Composer.html
[81] http://neon-toolkit.org/
[82] http://ode.openlinksw.com
[83] http://www.w3.org/2005/ajar/tab
[84] http://sig.ma
[85] www.agroxml.de
[86] http://aims.fao.org/standards/agrovoc/about
[87] http://aims.fao.org/website/AGROVOC-Thesaurus/sub
[88] http://aims.fao.org/website/Ontology-relationships/sub
[89] http://aims.fao.org/network-fisheries-ontologies
[90] http://www.neon-project.org/
[91] http://www.opengeospatial.org/standards/om
[92] http://schemas.opengis.net/om/2.0/
[93] http://def.seegrid.csiro.au/static/isotc211lax/iso19156/2011/
[94] http://www.opengeospatial.org/standards/sensorml
[95] http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/
[96] http://www.opengeospatial.org/standards/gml
[97] http://www.opengeospatial.org/standards/geosparql
[98] http://en.wikipedia.org/wiki/GeoSPARQL
[99] http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.doc
[100] http://www.heppnetz.de/projects/goodrelations/
[101] GDD: http://apps.gs1.org/gdd/SitePages/Home.aspx
[102] Monika Solanki and Christopher Brewster; Representing Supply Chain Events on the Web of Data. Proceedings of the 3rd International Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2013), , held at the International Semantic Web Conference (ISWC 2013), 21-25 October 2013.
[103] Monika Solanki and Christopher Brewster; Consuming Linked data in Supply Chains: Enabling data visibility via Linked Pedigrees. Proceedings of the Fourth International Workshop on Consuming Linked Data (COLD2013), held at the International Semantic Web Conference (ISWC 2013), 21-25 October 2013
[104] ”White Paper - The Java Language Environment”, Oracle, http://www.oracle.com/technetwork/java/langenv-140151.html
[105] “Overview of C++” - Bjarne Stroustrup, AT&T Laboratories; http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.26.9545
[106] http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29
[107] http://www.rfc-base.org/rfc-1866.html
[108] http://dev.opera.com/articles/view/12-the-basics-of-html/
[109] http://en.wikipedia.org/wiki/HTML5
[110] http://en.wikipedia.org/wiki/Cascading_Style_Sheets
[111] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/JavaScript_Overview
[112] http://www.eclipse.org/org/
[113] http://en.wikipedia.org/wiki/NetBeans
[114] www.w3.org/TR/widgets/
[115] http://www.jcp.org/en/jsr/detail?id=286

[116] http://jsr286tutorial.blogspot.de/p/portlet_21.html
[117] http://rajeevalochanabr.wordpress.com/category/introduction
[118] http://en.wikipedia.org/wiki/Java_Servlet
[119] http://en.wikipedia.org/wiki/JavaServer_Pages
[120] http://en.wikipedia.org/wiki/WebSocket
[121] www.json.org
[122] http://en.wikipedia.org/wiki/Java_Message_Service
[123] http://en.wikipedia.org/wiki/Data_distribution_service
[124] http://en.wikipedia.org/wiki/Web_Services_Description_Language
[125] www.internet-of-services.com/index.php?id=288&cmd=infomail&backURL=index.php%3Fid%3D288
[126] www.w3.org/2005/Incubator/usdl/charter
[127] http://en.wikipedia.org/wiki/SOAP
[128] http://gravity.sourceforge.net/servicebinder/osginutshell.html
[129] http://jcp.org/en/jsr/detail?id=330
[130] http://jcp.org/en/jsr/detail?id=311
[131] http://jcp.org/en/jsr/detail?id=224
[132] http://en.wikipedia.org/wiki/Security_Assertion_Markup_Language
[133] http://openid.net/specs/openid-authentication-2_0.html
[134] http://en.wikipedia.org/wiki/OpenID
[135] OAuth 1.0 Protocol: http://tools.ietf.org/html/rfc5849;
OAuth 2.0 Authorization Framework: http://tools.ietf.org/html/rfc6749;
OAuth 2.0 Authorization Framework: Bearer Token Usage: http://tools.ietf.org/html/rfc6750
[136] http://en.wikipedia.org/wiki/OAuth
[137] www.ietf.org/rfc/rfc5849.txt
[138] http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.doc
[139] http://en.wikipedia.org/wiki/XACML
[140] http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol
[141] http://en.wikipedia.org/wiki/S/MIME
[142] http://en.wikipedia.org/wiki/Transport_Layer_Security
[143] http://www.w3.org/standards/techs/widgets#w3c_all
[144] http://www.jcp.org/en/jsr/detail?id=315
[145] http://tools.ietf.org/html/rfc4627
[146] http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.doc
[147] http://www.gs1.com/barcodes/technical/idkeys/grai
[148] GS1 General Specifications: http://www.gs1.org/docs/gsmp/barcodes/GS1_General_Specifications.pdf
[149] http://www.gs1.org/gsmp/kc/ecom/eancom
[150] http://www.w3.org/TR/REC-xml/
[151] 2000/678/EC: Commission Decision of 23 October 2000 laying down detailed rules for registration of holdings in national databases for porcine animals as foreseen by Council Directive 64/432/EEC http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32000D0678:EN:NOT
[152] Verordnung zum Schutz gegen die Verschleppung von Tierseuchen im Viehverkehr (http://www.gesetze-im-internet.de/viehverkv_2007/index.html" \l "BJNR127400007BJNE004101377)
[153] Council Directive 2008/71/EC of 15 July 2008 on the identification and registration of pigs. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32008L0071:EN:NOT
[154] http://www.magrama.gob.es/en/ganaderia/temas/trazabilidad-animal/identificacion-animal/otras-especies/
[155] Guide de l'éleveur pour l'identification des porcins (http://www.alsace.chambagri.fr/fileadmin/documents_alsace/INTERNET/elevage/identification/guide_identification_porcins.pdf)
[156] Council Regulation (EC) No 21/2004 of 17 December 2003 establishing a system for the identification and registration of ovine and caprine animals and amending Regulation (EC) No 1782/2003 and Directives 92/102/EEC and 64/432/EEC (http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32004R0021:EN:NOT)
[157] http://agriculture.gouv.fr/IMG/pdf/2009_identification_bovine.pdf
[158] https://www.gov.uk/cattle-identification-registration-and-movement
[159] http://rpa.defra.gov.uk/rpa/index.nsf/15f3e119d8abcb5480256ef20049b53a/eabe7c431b0e8c5a8025765c0037b3fb/$FILE/bsi-cattle-spec.pdf

[160] http://rpa.defra.gov.uk/rpa/index.nsf/15f3e119d8abcb5480256ef20049b53a/907392505b99169d8025703b00417e1d/$FILE/Cattle%20keepers%20handbook%20V4%20July%202011.pdf
[161] http://www.fao.org/fishery/collection/asfis/en
[162] http://www.w3.org/1999/02/22-rdf-syntax-ns#
[163] Klyne, G. & Carroll, J. J., (2004) Resource Description Framework (RDF): Concepts and Abstract Syntax, World Wide Web Consortium, http://www.w3.org/TR/rdf-concepts/
[164] Miller, E. & Manola, F., (2004) RDF Primer, World Wide Web Consortium, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
[165] Hitzler, P.; Krötzsch, M.; Parsia, B.; Patel-Schneider, P. F. & Rudolph, S., (2009a) OWL 2 Web Ontology Language --- Primer, World Wide Web Consortium, http://www.w3.org/TR/2009/REC-owl2-primer-20091027
[166] Bray, T.; Paoli, J. & Sperberg-McQueen, C.,(1998) Extensible Markup Language (XML) 1.0, W3C, http://www.w3.org/TR/1998/REC-xml-19980210
[167] Brickley, D. & Guha, R., (2004) Resource Description Framework (RDF) Schema Specification, W3C, http://www.w3.org/TR/rdf-schema/
[168] Baader, F. & Nutt, W., (2003) Basic Description Logics, in Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D. & Patel-Schneider, P. F. (ed.) Description Logic Handbook, Cambridge University Press, , 43-95
[169] Motik, B.; Fokoue, A.; Horrocks, I.; Wu, Z.; Lutz, C. & Grau, B. C., (2009) OWL 2 Web Ontology Language Profiles, World Wide Web Consortium, http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/

7 Appendix 1: Description of FIspace Relevant Platform Standards and Industry-specific Standards for further Reading

7.1 Platform Standards

7.1.1 Languages

7.1.1.1 Java

“The Java programming language is designed to meet the challenges of application development in the context of heterogeneous, network-wide distributed environments. Paramount among these challenges is secure delivery of applications that consume the minimum of system resources, can run on any hardware and software platform, and can be extended dynamically.

The Java programming language originated as part of a research project to develop advanced software for a wide variety of network devices and embedded systems. The goal was to develop a small, reliable, portable, distributed, real-time operating platform. When the project started, C++ was the language of choice. But over time the difficulties encountered with C++ grew to the point where the problems could best be addressed by creating an entirely new language platform. Design and architecture decisions drew from a variety of languages such as Eiffel, SmallTalk, Objective C, and Cedar/Mesa. The result is a language platform that has proven ideal for developing secure, distributed, network-based end-user applications in environments ranging from network-embedded devices to the World-Wide Web and the desktop.

The design requirements of the Java programming language are driven by the nature of the computing environments in which software must be deployed. The massive growth of the Internet and the World-Wide Web leads to a completely new way of looking at development and distribution of software. To live in the world of electronic commerce and distribution, Java technology must enable the development of secure, high performance, and highly robust applications on multiple platforms in heterogeneous, distributed networks. Operating on multiple platforms in heterogeneous networks invalidates the traditional schemes of binary distribution, release, upgrade, patch, and so on. To survive in this jungle, the Java programming language must be architecture neutral, portable, and dynamically adaptable.

The Java system that emerged to meet these needs is simple, so it can be easily programmed by most developers; familiar, so that current developers can easily learn the Java programming language; object oriented, to take advantage of modern software development methodologies and to fit into distributed client-server applications; multithreaded, for high performance in applications that need to perform multiple concurrent activities, such as multimedia; and interpreted, for maximum portability and dynamic capabilities.” [104]

7.1.1.2 C++

“The C++ programming language provides a model of memory and computation that closely matches that of most computers. In addition, it provides powerful and flexible mechanisms for abstraction; that is, language constructs that allow the programmer to introduce and use new types of objects that match the concepts of an application. Thus, C++ supports styles of programming that rely on fairly direct manipulation of hardware resources to deliver a high degree of efficiency plus higher-level styles of programming

that rely on user-defined types to provide a model of data and computation that is closer to a human's view of the task being performed by a computer. These higher-level styles of programming are often called data abstraction, object-oriented programming, and generic programming...

C++ was designed to deliver the flexibility and efficiency of C for systems programming together with Simula's facilities for program organization (usually referred to as object-oriented programming)...

The aim of C++ was to improve the quality of programs produced by making better design and programming techniques simpler to use and affordable. Most of these techniques have their root in Simula and are usually discussed under the labels of object-oriented programming and object-oriented design. However, the aim was always to support a range of design and programming styles. This contrasts to a view of language design that tries to channel all system building into a single heavily supported and enforced style (paradigm)." [105]

7.1.1.3 C#

"C# ... is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, procedural, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within its .NET initiative and later approved as a standard by ECMA (ECMA-334) and ISO (ISO/IEC 23270:2006). C# is one of the programming languages designed for the Common Language Infrastructure.

C# is intended to be a simple, modern, general-purpose, object-oriented programming language. By design, C# is the programming language ... that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or Fortran." [106]

7.1.1.4 HTML

"The Hypertext Markup Language (HTML) is a simple markup language used to create hypertext documents that are platform independent. HTML documents are SGML documents (ISO Standard 8879:1986, "Information Processing Text and Office Systems; Standard Generalized Markup Language" with generic semantics that are appropriate for representing information from a wide range of domains. HTML markup can represent hypertext news, mail, documentation, and hypermedia; menus of options; database query results; simple structured documents with in-lined graphics; and hypertext views of existing bodies of information... The specification also defines the semantics of the HTML syntax and how that syntax should be interpreted by user agents." [107]

"A user agent is any software that is used to access web pages on behalf of users... All types of desktop browser software (Internet Explorer, Opera, Firefox, Safari, Chrome etc.) and alternative browsers for other devices (such as the Wii Internet channel, and mobile phone browsers such as Opera Mini and WebKit on the iPhone) are user agents. The automated programs that Google and Yahoo! use to index

the web to use in their search engines are also user agents, but no human being is controlling them directly." [108]

7.1.1.5 HTML5

"HTML5 is a markup language for structuring and presenting content for the Web and a core technology of the Internet. It is the fifth revision of the HTML standard ..." and "its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices (web browsers, parsers, etc.). HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

Following its immediate predecessors HTML 4.01 and XHTML 1.1, HTML5 is a response to the observation that the HTML and XHTML in common use on the Web are a mixture of features introduced by various specifications, along with those introduced by software products such as web browsers, those established by common practice, and the many syntax errors in existing web documents. It is also an attempt to define a single markup language that can be written in either HTML or XHTML syntax. It includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalises the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a potential candidate for cross-platform mobile applications. Many features of HTML5 have been built with the consideration of being able to run on low-powered devices such as smartphones and tablets...

In particular, HTML5 adds many new syntactic features. These include the new <video>, <audio> and <canvas> elements, as well as the integration of scalable vector graphics (SVG) content (that replaces the uses of generic <object> tags) and MathML for mathematical formulas. These features are designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plug-ins and APIs. Other new elements, such as <section>, <article>, <header> and <nav>, are designed to enrich the semantic content of documents. New attributes have been introduced for the same purpose, while some elements and attributes have been removed. Some elements, such as <a>, <cite> and <menu> have been changed, redefined or standardized. The APIs and Document Object Model (DOM) are no longer afterthoughts, but are fundamental parts of the HTML5 specification. HTML5 also defines in some detail the required processing for invalid documents so that syntax errors will be treated uniformly by all conforming browsers and other user agents." [109]

7.1.1.6 CSS

"Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document, including plain XML, SVG and XUL.

CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and

repetition in the structural content (such as by allowing for tableless web design). CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified.

CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998), and they also operate a free CSS validation service...The W3C has now deprecated the use of all presentational HTML markup." [110]

7.1.1.7 JavaScript

"JavaScript is a cross-platform, object-oriented scripting language. JavaScript is a small, lightweight language; it is not useful as a standalone language, but is designed for easy embedding in other products and applications, such as web browsers. Inside a host environment, JavaScript can be connected to the objects of its environment to provide programmatic control over them.

Core JavaScript contains a core set of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects; for example:

- Client-side JavaScript extends the core language by supplying objects to control a browser (Firefox or another web browser) and its Document Object Model (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.
- Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a relational database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

Through JavaScript's LiveConnect functionality, it's possible for Java and JavaScript code to communicate with each other. From JavaScript, one can instantiate Java objects and access their public methods and fields. From Java, one can access JavaScript objects, properties, and methods.

This standardized version of JavaScript, called ECMAScript, behaves the same way in all applications that support the standard. Companies can use the open standard language to develop their implementation of JavaScript. The ECMAScript standard is documented in the ECMA-262 specification.

The ECMA-262 standard is also approved by the ISO (International Organization for Standardization) as ISO-16262.” [111]

7.1.2 Development environments

7.1.2.1 Eclipse

Eclipse “is a multi-language software development environment comprising a workspace and an extensible plug-in system... It can be used to develop applications in Java and, by means of various plug-ins, other programming languages... The Eclipse Platform uses plug-ins to provide all functionality within and on top of the runtime system, in contrast to some other applications, in which functionality is hard coded... With the exception of a small run-time kernel, everything in Eclipse is a plug-in” (wikipedia), so a complete ecosystem adding new functionalities has been created around the core parts of Eclipse. “Eclipse is a community for individuals and organizations who wish to collaborate on commercially-friendly open source software. Its projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. The Eclipse Foundation is a not-for-profit, member supported corporation that hosts the Eclipse projects and helps cultivate both an open source community and an ecosystem of complementary products and services.

The Eclipse Project was originally created by IBM in November 2001 and supported by a consortium of software vendors. The Eclipse Foundation was created in January 2004 as an independent not-for-profit corporation to act as the steward of the Eclipse community. The independent not-for-profit corporation was created to allow a vendor neutral and open, transparent community to be established around Eclipse. Today, the Eclipse community consists of individuals and organizations from a cross section of the software industry.” [112]

7.1.2.2 NetBeans

“NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others. The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform (including the NetBeans IDE itself) can be extended by third party developers... NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, Maven support, refactorings, version control (supporting CVS, Subversion, Mercurial and Clearcase)...

All the functions of the IDE are provided by modules. Each module provides a well defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow NetBeans to be extended. New features, such as support for

other programming languages, can be added by installing additional modules. For instance, Sun Studio, Sun Java Studio Enterprise, and Sun Java Studio Creator from Sun Microsystems are all based on the NetBeans IDE." [113]

7.1.3 Integration mechanisms

7.1.3.1 Widgets

"The specifications of the Worl Wide Web Consortium W3C [143] provide a standard packaging format and metadata for a class of software known commonly as packaged apps or widgets. Unlike traditional user interface widgets (e.g., buttons, input boxes, toolbars, etc.), widgets as specified in this standard are full-fledged client-side applications that are authored using technologies such as HTML and then packaged for distribution. Examples range from simple clocks, stock tickers, news casters, games and weather forecasters, to complex applications that pull data from multiple sources to be "mashed-up" and presented to a user in some interesting and useful way...

The specification relies on PKWare's Zip specification as the archive format, XML as a configuration document format, and a series of steps that runtimes follow when processing and verifying various aspects of a package. The packaging format acts as a container for files used by a widget. The configuration document is an XML vocabulary that declares metadata and configuration parameters for a widget. The steps for processing a widget package describe the expected behaviour and means of error handling for runtimes while processing the packaging format, configuration document, and other relevant files." [114]

7.1.3.2 Java Portlets

This specification (see [115]) deals with Portlets as Java technology based web components. "A portlet is an application that provides a specific piece of content (information or service) to be included as part of a portal page. It is managed by a portlet container that processes requests and generates dynamic content. Portlets are used by portals as pluggable user interface components that provide a presentation layer to information systems. The content generated by a portlet is also called a fragment. A fragment is a piece of markup (e.g. HTML, XHTML, WML) adhering to certain rules and can be aggregated with other fragments to form a complete document. The content of a portlet is normally aggregated with the content of other portlets to form the portal page. The lifecycle of a portlet is managed by the portlet container.

Web clients interact with portlets via a request/response paradigm implemented by the portal. Normally, users interact with content produced by portlets, for example by following links or submitting forms, resulting in portlet actions being received by the portal, which are forwarded by it to the portlets targeted by the user's interactions." [116] "The content generated by a portlet may vary from one user to another depending on the user configuration for the portlet...

A portlet container runs portlets and provides them with the required runtime environment. A portlet container contains portlets and manages their lifecycle. It also provides persistent storage for portlet preferences. A portlet container receives requests from the portal to execute requests on the portlets hosted by it. A portlet container is not responsible for aggregating the content produced by the portlets. It is the responsibility of the portal to handle the aggregation. A portal and a portlet container can be built together

as a single component of an application suite or as two separate components of a portal application." [117]

7.2 Server-side components

7.2.1 Java Servlet

"A servlet is a Java programming language class specification [144] used to extend the capabilities of a server. It is used in combination with Apache Tomcat for instance. Although servlets can respond to any types of requests, they are commonly used to extend the applications hosted by web servers, so they can be thought of as Java Applets that run on servers instead of in web browsers. These kinds of servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET...

Servlets are most often used to:

- Process or store data that was submitted from an HTML form
- Provide dynamic content such as the results of a database query
- Manage state information that does not exist in the stateless HTTP protocol, such as filling the articles into the shopping cart of the appropriate customer ...

A "servlet" is a Java class that conforms to the Java Servlet API, a protocol by which a Java class may respond to requests. Servlets could in principle communicate over any client–server protocol, but they are most often used with the HTTP protocol. Thus "servlet" is often used as shorthand for "HTTP servlet". Thus, a software developer may use a servlet to add dynamic content to a web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or URL rewriting.

To deploy and run a servlet, a web container must be used. A web container (also known as a servlet container) is essentially the component of a web server that interacts with the servlets. The web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

The Servlet API, contained in the Java package hierarchy javax.servlet, defines the expected interactions of the web container and a servlet. A Servlet is an object that receives a request and generates a response based on that request. The basic Servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package javax.servlet.http defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the web server and a client. Servlets may be packaged in a WAR file as a web application.

Servlets can be generated automatically from Java Server Pages (JSP) by the JavaServer Pages compiler. The difference between servlets and JSP is that servlets typically embed HTML inside Java code, while JSPs embed Java code in HTML. The direct usage of servlets to generate HTML has however be-

come quite rare. A somewhat older usage is to use servlets in conjunction with JSPs in a pattern called "Model 2", which is a flavour of the model–view–controller pattern.

The current version of Java Servlet is 3.0." [118]

7.2.2 JavaServer Pages

"JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types... JSP is similar to PHP, but it uses the Java programming language.

To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required... Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime; each JSP's servlet is cached and re-used until the original JSP is modified.

JSP can be used independently or as the view component of a server-side model–view–controller design, normally with JavaBeans as the model and Java servlets (or a framework such as Apache Struts) as the controller. This is a type of Model 2 architecture.

JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, use Java bytecode rather than a native software format. Like any other Java program, they must be executed within a Java virtual machine (JVM) that integrates with the server's host operating system to provide an abstract platform-neutral environment.

JSPs are usually used to deliver HTML and XML documents, but through the use of OutputStream, they can deliver other types of data as well. The Web container creates JSP implicit objects like pageContext, servletContext, session, request & response." [119]

7.2.3 WebSockets

"WebSocket is a web technology providing full-duplex communications channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.

WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request. The WebSocket protocol makes possible more interaction between a browser and a web site, facilitating live content and the creation of real-time games. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server. A similar effect has been achieved in non-standardized ways using stop-gap technologies such as Comet.

In addition, the communications are done over TCP port number 80, which is of benefit for those environments which block non-standard Internet connections using a firewall. WebSocket protocol is currently supported in several browsers including Google Chrome, Internet Explorer, Firefox, Safari and Opera. WebSocket also requires web applications on the server to support it ...

Like TCP, WebSocket provides for full-duplex communication. Websocket differs from TCP in that it enables a stream of messages instead of a stream of bytes. Before WebSocket, port 80 full-duplex communication was attainable using Comet channels; however, comet implementation is nontrivial, and due to the TCP handshake and HTTP header overhead, it is inefficient for small messages. WebSocket protocol aims to solve these problems without compromising security assumptions of the web." [120]

7.3 Event processing

7.3.1 JSON

"JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures." [121]

7.3.2 JMS

"The Java Message Service (JMS) API is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. JMS is a part of the Java Platform, Enterprise Edition, and is defined by a specification developed under the Java Community Process as JSR 914. It is a messaging standard that allows application components based on the Java Enterprise Edition (JEE) to create, send, receive, and read messages. It allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous...

Messaging is a form of loosely coupled distributed communication, where in this context the term 'communication' can be understood as an exchange of messages between software components. Message-oriented technologies attempt to relax tightly coupled communication (such as TCP network sockets, CORBA or RMI) by the introduction of an intermediary component. This approach allows software com-

ponents to communicate 'indirectly' with each other. Benefits of this include message senders not needing to have precise knowledge of their receivers...

JMS provides a way of separating the application from the transport layer of providing data. The same Java classes can be used to communicate with different JMS providers by using the Java Naming and Directory Interface (JNDI) information for the desired provider. The classes first use a connection factory to connect to the queue or topic, and then use populate and send or publish the messages. On the receiving side, the clients then receive or subscribe to the messages." [122]

7.3.3 DDS

"The Data Distribution Service for Real-Time Systems (DDS) is an Object Management Group (OMG) M2M middleware standard that aims to enable scalable, real-time, dependable, high performance and interoperable data exchanges between publishers and subscribers. DDS is designed to address the needs of mission- and business-critical applications like financial trading, air traffic control, smart grid management, and other big data applications. The standard is being increasingly used in a wide range of industries including Intelligent Systems. Among various applications, DDS is currently being smartphone operating systems, transportation systems and vehicles, software defined radio, and by healthcare providers. DDS is positioned to play a large role in the Internet of Things."

DDS simplifies complex network programming. "It implements a publish/subscribe model for sending and receiving data, events, and commands among the nodes. Nodes that are producing information (publishers) create "topics" (e.g., temperature, location, pressure) and publish "samples." DDS takes care of delivering the sample to all subscribers that declare an interest in that topic.

DDS handles all the transfer chores: message addressing, data marshalling and demarshalling (so subscribers can be on different platforms than the publisher), delivery, flow control, retries, etc. Any node can be a publisher, subscriber, or both simultaneously. The DDS publish-subscribe model virtually eliminates complex network programming for distributed applications.

DDS supports mechanisms that go beyond the basic publish-subscribe model. The key benefit is that applications that use DDS for their communications are entirely decoupled. Very little design time has to be spent on how to handle their mutual interactions. In particular, the applications never need information about the other participating applications, including their existence or locations. DDS automatically handles all aspects of message delivery, without requiring any intervention from the user applications, including:

- determining who should receive the messages
- where recipients are located
- what happens if messages cannot be delivered

This is made possible by the fact that DDS allows the user to specify Quality of Service (QoS) parameters as a way to configure automatic-discovery mechanisms and specify the behaviour used when sending and receiving messages. The mechanisms are configured up-front and require no further effort on the

user's part. By exchanging messages in a completely anonymous manner, DDS greatly simplifies distributed application design and encourages modular, well-structured programs.

DDS also automatically handles hot-swapping redundant publishers if the primary fails. Subscribers always get the sample with the highest priority whose data is still valid (that is, whose publisher-specified validity period has not expired). It also automatically switches back to the primary when it recovers." [123]

7.4 Application Store

7.4.1 WSDL

"The Web Services Description Language is an XML-based interface description language that is used for describing the functionality offered by a web service. A WSDL description of a web service (also referred to as a WSDL file) provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. It thus serves a purpose that corresponds roughly to that of a method signature in a programming language...

The WSDL describes services as collections of network endpoints, or ports. The WSDL specifications provide an XML format for documents for this purpose. The abstract definitions of ports and messages are separated from their concrete use or instance, allowing the reuse of these definitions. A port is defined by associating a network address with a reusable binding, and a collection of ports defines a service. Messages are abstract descriptions of the data being exchanged, and port types are abstract collections of supported operations. The concrete protocol and data format specifications for a particular port type constitutes a reusable binding, where the operations and messages are then bound to a concrete network protocol and message format. In this way, WSDL describes the public interface to the Web service.

WSDL is often used in combination with SOAP and an XML Schema to provide Web services over the Internet. A client program connecting to a Web service can read the WSDL file to determine what operations are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the operations listed in the WSDL file using XML or HTTP.

The current version of the specification is 2.0; version 1.1 has not been endorsed by the W3C but version 2.0 is a W3C recommendation.[1] WSDL 1.2 was renamed WSDL 2.0 because of its substantial differences from WSDL 1.1. By accepting binding to all the HTTP request methods (not only GET and POST as in version 1.1), the WSDL 2.0 specification offers better support for RESTful web services, and is much simpler to implement. However support for this specification is still poor in software development kits for Web Services which often offer tools only for WSDL 1.1. Furthermore, the latest version (version 2.0) of the Business Process Execution Language (BPEL) only supports WSDL 1.1." [124]

7.4.2 USDL

The "Unified Service Description Language (USDL) creates a "commercial envelope" around a service. More specifically, USDL allows a unified description of business, operational and technical aspects of services as depicted in the Figure 4. Technical services may be lifted to business services, but USDL also

allows describing more manual or physical services. As many services have a hybrid character with both, a digital and physical or manual footprint, USDL can facilitate the combination and aggregation of such services. Therefore, USDL can be considered one of the foundational technologies to set up an Internet of Services around today's core enterprise systems. [125]

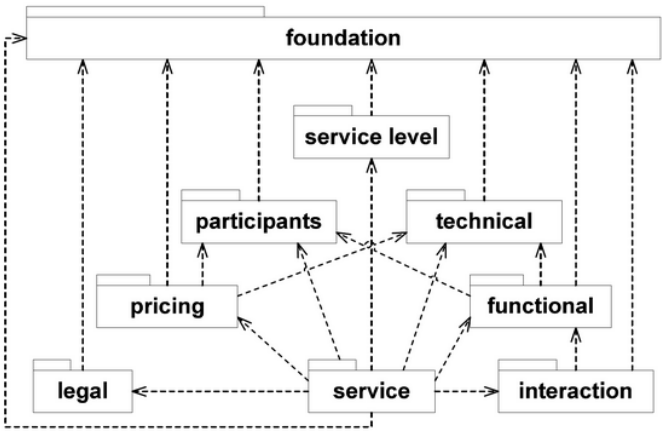


Figure 4: Modules of USDL

USDL went through several iterations since its conception in 2007. The introduction of Linked USDL targets simplification and reuse of existing schemas on the basis of Linked Data principles and technologies (see Chapter 4). Although Linked USDL is sponsored by FI-Ware, the Linked USDL community has been established in order to foster an open development as well as a broad and world-wide adoption.

"The mission of the Unified Service Description Language (USDL) Incubator Group" of the W3C, "part of the Incubator Activity, is to define a language for describing general and generic parts of technical and business services to allow services to become tradable and consumable. Technical services are considered electronic services based on WSDL, REST or other technical specifications. Business services are defined as business activities that are provided by a service provider to a service consumer to create value for the consumer. Thus, the business services are more general and comprise manual and technical services. This enables new business models in the field of service brokerage because services can automatically be offered, delivered, executed, and composed from services of different providers.

The language is usable for any purpose and implementation scenario of future business services on a general level. However, it will be extendable for industry-specific aspects. The specification aims at complementing the technical language stack by adding required business and operational information. The targeted groups for USDL are service providers, hosting providers, gateways, and service consumption channels. Industry-specific and general-purpose attributes of a service will be derived based on these use cases and their target groups." [126]

7.5 System and Data Integration

7.5.1 SOAP

"SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on XML Information Set for its message format, and usually relies on other Application Layer protocols, most nota-

bly Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

SOAP can form the foundation layer of a web services protocol stack, providing a basic messaging framework upon which web services can be built. This XML based protocol consists of three parts: an envelope, which defines what is in the message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing procedure calls and responses. SOAP has three major characteristics: Extensibility (security and WS-routing are among the extensions under development), Neutrality (SOAP can be used over any transport protocol such as HTTP, SMTP, TCP, or JMS) and Independence (SOAP allows for any programming model). As an example of how SOAP procedures can be used, a SOAP message could be sent to a web site that has web services enabled, such as a real-estate price database, with the parameters needed for a search. The site would then return an XML-formatted document with the resulting data, e.g., prices, location, features. With the data being returned in a standardized machine-parsable format, it can then be integrated directly into a third-party web site or application.

The SOAP architecture consists of several layers of specifications for: message format, Message Exchange Patterns (MEP), underlying transport protocol bindings, message processing models, and protocol extensibility. SOAP is the successor of XML-RPC, though it borrows its transport and interaction neutrality and the envelope/header/body from elsewhere (probably from WDDX).” [127]

7.5.2 OSGi

The OSGi “specifications define the OSGi Service Platform, which consists of two pieces: the OSGi framework and a set of standard service definitions. The OSGi framework, which sits on top of a Java virtual machine, is the execution environment for services. The OSGi framework was originally conceived to be used inside restricted environments, such as a set-top box. OSGi can however be used in other domains, as for example, an infrastructure to support underlying release 3.0 of the eclipse IDE...

The framework can be divided in two main elements:

- Services platform.
- Deployment infrastructure.

A services platform is defined as a software platform that supports the service orientation interaction depicted in Figure 5. This interaction involves three main actors: service providers, service requesters and a service registry, although only the service registry belongs to the services platform. In the service orientation interaction, service providers publish service descriptions, and service requesters discover services and bind to the service providers. Publication and discovery are based on a service description.

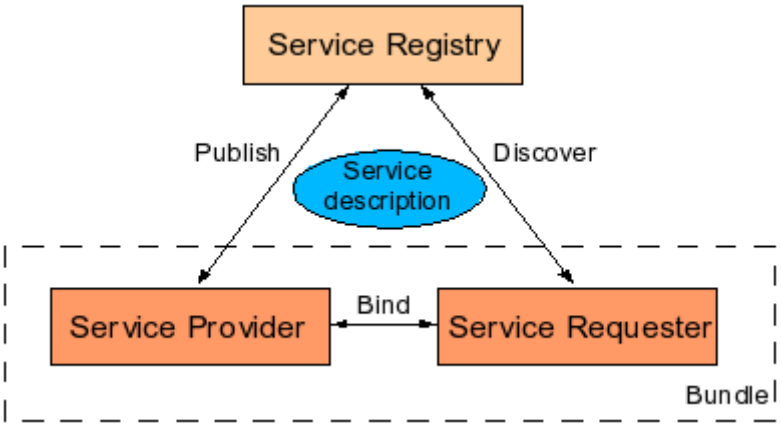


Figure 5: Service orientation interaction

In the context of OSGi, a service is described as a Java class or interface, the service interface, along with a variable number of attributes, the service properties, which are name and value pairs. Service properties allow different service providers that provide services with the same service interface to be differentiated. The service registry allows service providers to be discovered through queries formulated in an LDAP syntax. Additionally, notification mechanisms allow service requesters to receive events signalling changes in the service registry; these changes include the publication or retrieval of a particular service.

In OSGi, service providers and requesters are part of an entity called a bundle that is both a logical as well as physical entity. Service interfaces are implemented by objects created by the bundle. In standard OSGi, the bundle is responsible for run-time service dependency management activities which include publication, discovery and binding as well as adapting to changes resulting from dynamic availability (arrival or departure) of services that are bound to the bundle.

For the deployment infrastructure, a bundle correspond to a delivery and deployment unit that is materialized by a JAR file that contains code and resources (i.e., images, libraries, etc.) along with a file that contains information about the bundle, the manifest file. The OSGi framework provides mechanisms to support continuous deployment activities (for example a local console or an administration web page). These deployment activities include installation, removal, update, starting (activation) and stopping (deactivation) of a physical bundle. Once a bundle is installed in the platform, it can be activated if deployment dependencies that are associated to the bundle are fulfilled.

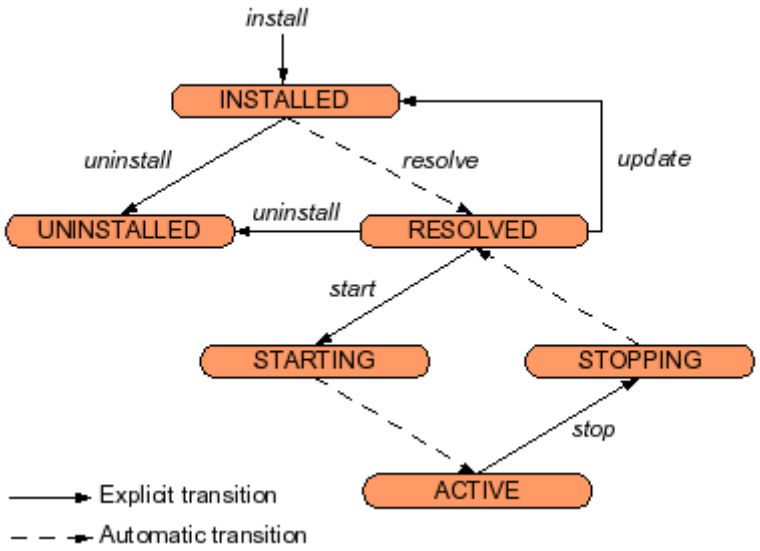


Figure 6: Physical bundle life-cycle

Deployment activities are realized according to a well defined series of states depicted in Figure 6; these states correspond to the physical bundle life-cycle. The activation or de-activation of a physical bundle results in the creation or destruction of a unique logical bundle, materialized by an instance from a class inside the bundle called a bundle activator. When the instance is created, the execution environment calls an activation method that signals the logical bundle that it is active. When the physical bundle is de-activated, the execution environment calls a de-activation method. When the logical bundle is active, it can publish or discover services and bind with other bundles by accessing the framework's services registry. It can also be notified from changes that occur in the framework by subscribing as an event listener." [128]

7.5.3 Dependency Injection

The Dependency Injection JSR 330 specifies a means for obtaining objects in such a way as to maximize reusability, testability and maintainability compared to traditional approaches such as constructors, factories, and service locators (e.g., JNDI). This process, known as dependency injection, is beneficial to most nontrivial applications.

Many types depend on other types. For example, a Stopwatch might depend on a TimeSource. The types on which a type depends are known as its dependencies. The process of finding an instance of a dependency to use at run time is known as resolving the dependency. If no such instance can be found, the dependency is said to be unsatisfied, and the application is broken.

In the absence of dependency injection, an object can resolve its dependencies in a few ways. It can invoke a constructor, hard-wiring an object directly to its dependency's implementation and life cycle:

```
class Stopwatch {
   final TimeSource timeSource;
   Stopwatch () {
      timeSource = new AtomicClock(...);
   }
   void start() { ... }
   long stop() { ... }
}
```

If more flexibility is needed, the object can call out to a factory or service locator:

```
class Stopwatch {
   final TimeSource timeSource;
   Stopwatch () {
      timeSource = DefaultTimeSource.getInstance();
   }
   void start() { ... }
   long stop() { ... }
}
```

In deciding between these traditional approaches to dependency resolution, a programmer must make trade-offs. Constructors are more concise but restrictive. Factories decouple the client and implementation to some extent but require boilerplate code. Service locators decouple even further but reduce compile time type safety. All three approaches inhibit unit testing. For example, if the programmer uses a factory, each test against code that depends on the factory will have to mock out the factory and remember to clean up after itself or else risk side effects:

```
void testStopwatch() {
   TimeSource original = DefaultTimeSource.getInstance();
   DefaultTimeSource.setInstance(new MockTimeSource());
   try {
      // Now, we can actually test Stopwatch.
      Stopwatch sw = new Stopwatch();
      ...
   } finally {
      DefaultTimeSource.setInstance(original);
   }
}
```

In practice, supporting this ability to mock out a factory results in even more boilerplate code. Tests that mock out and clean up after multiple dependencies quickly get out of hand. To make matters worse, a programmer must predict accurately how much flexibility will be needed in the future or else suffer the consequences. If a programmer initially elects to use a constructor but later decides that more flexibility is required, the programmer must replace every call to the constructor. If the programmer errs on the side of caution and write factories up front, it may result in a lot of unnecessary boilerplate code, adding noise, complexity, and error-proneness.

Dependency injection addresses all of these issues. Instead of the programmer calling a constructor or factory, a tool called a dependency injector passes dependencies to objects:

```
class Stopwatch {
   final TimeSource timeSource;
   @Inject Stopwatch(TimeSource TimeSource) {
      this.TimeSource = TimeSource;
   }
   void start() { ... }
   long stop() { ... }
}
```

The injector further passes dependencies to other dependencies until it constructs the entire object graph. For example, suppose the programmer asked an injector to create a StopwatchWidget instance:

```
/** GUI for a Stopwatch */
class StopwatchWidget {
   @Inject StopwatchWidget(Stopwatch sw) { ... }
   ...
}
```

The injector might:

1. Find a TimeSource
2. Construct a Stopwatch with the TimeSource
3. Construct a StopwatchWidget with the Stopwatch

This leaves the programmer's code clean, flexible, and relatively free of dependency-related infrastructure.

In unit tests, the programmer can now construct objects directly (without an injector) and pass in mock dependencies. The programmer no longer needs to set up and tear down factories or service locators in each test. This greatly simplifies our unit test:

```
void testStopwatch() {
   Stopwatch sw = new Stopwatch(new MockTimeSource());
   ...
}
```

The total decrease in unit-test complexity is proportional to the product of the number of unit tests and the number of dependencies.

Programmers annotate constructors, methods, and fields to advertise their injectability (constructor injection is demonstrated in the examples above). A dependency injector identifies a class's dependencies by inspecting these annotations, and injects the dependencies at runtime. Moreover, the injector can verify that all dependencies have been satisfied at build time. A service locator, by contrast, cannot detect unsatisfied dependencies until run time.

A programmer configures a dependency injector so it knows what to inject. Different configuration approaches make sense in different contexts. One approach is to search the classpath for dependency implementations, avoiding the need for the programmer to write explicit code. This approach could be useful in quick-and-dirty prototypes. A programmer working on a large, long-lived application might prefer a more explicit, compartmentalized approach. For example, the programmer could write XML that tells the injector to inject an EJB client proxy named "NetworkTimeSource" when a class needs a TimeSource:

```
<binding type="TimeSource" ejb="NetworkTimeSource"/>
```

It often makes sense to use more than one configuration mechanism in the the same application. For example, a quick and dirty prototype might grow into a real application, and the programmer could incrementally migrate to a more maintainable configuration. As another example, a program might configure some resources explicitly, while others are configured automatically based on an external XML file or database.

This JSR standardizes a low-level kernel API that can be used directly by a user or as an integration point for higher level configuration approaches. This approach enables portable Java applications without quashing innovation in dependency injector configuration. [129]

7.5.4 JAX-RS

The Java JSR 311 specification provides and API that will enable developers to rapidly build Web applications in Java that are characteristic of the best designed parts of the Web. This JSR provides an API for

providing REST(Representational State Transfer) support in the Java Platform. Lightweight, RESTful approaches are emerging as a popular alternative to SOAP-based technologies for deployment of services on the internet. Prior to this API building RESTful Web services using the Java Platform was significantly more complex than building SOAP-based services and requires using low-level APIs like Servlets or the dynamic JAX-WS APIs. Correct implementation requires a high level of HTTP knowledge on the developer's part.

This JSR aims to provide a high level easy-to use API for developers to write RESTful web services independent of the underlying technology and will allow these services to run on top of the Java EE or the Java SE platforms. The goal of this JSR is to provide an easy to use, declarative style of programming using annotations for developers to write REST ful Web Services and also enable low level access in cases where needed by the application. [130]

7.5.5 JAX-WS

XML is a platform-independent means of representing structured information. XML Web Services use XML as the basis for communication between Web-based services and clients of those services and inherit XML's platform independence. SOAP describes one such XML based message format and "defines, using XML technologies, an extensible messaging framework containing a message construct that can be exchanged over a variety of underlying protocols." WSDLis "an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information." WSDL can be considered the defacto service description language for XML Web Services.

JAX-RPC defined APIs and conventions for supporting RPC oriented XML Web Services in the Java platform and interoperability between JAX-RPC implementations and with services implemented using other technologies. JAX-WS 2.0 is a follow-on to JAX-RPC 1.1 providing:

- Support for SOAP 1.2 while maintaining support for SOAP 1.1
- Support for WSDL 2.0 while rmaintaining support for WSDL 1.1.
- Use of JAXB for XML to Java mappings
- Java annotations to simplify the most common development scenarios for both clients and servers
- Support for client side asynchronous operations
- Simplified client and service access to the messages underlying an exchange
- Support for message based session management
- New mechanisms to produce fully portable clients
- Alignment with other updated JSRs related to Web Services Metadata (JSR 181), Java to WSDL mapping (JSR 109), Security (JSR 183) and others.

JAX-WS provides the standard mechanisms for supporting Web Services for Java based systems. [131]

7.6 Security

7.6.1 Authentication

7.6.1.1 SAML

“Security Assertion Markup Language (SAML) is an XML-based open standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider. SAML is a product of the OASIS Security Services Technical Committee. SAML dates from 2001; the most recent update of SAML is from 2005.

The single most important problem that SAML addresses is the web browser single sign-on (SSO) problem. Single sign-on solutions are abundant at the intranet level (using cookies, for example) but extending these solutions beyond the intranet has been problematic and has led to the proliferation of non-interoperable proprietary technologies. (Another more recent approach to addressing the browser SSO problem is the OpenID protocol.)

The SAML specification defines three roles: the principal (typically a user), the identity provider (aka IdP), and the service provider (aka SP). In the use case addressed by SAML, the principal requests a service from the service provider. The service provider requests and obtains an identity assertion from the identity provider. On the basis of this assertion, the service provider can make an access control decision - in other words it can decide whether to perform some service for the connected principal.

Before delivering the identity assertion to the SP, the IdP may request some information from the principal - such as a user name and password - in order to authenticate the principal. SAML specifies the assertions between the three parties: in particular, the messages that assert identity that are passed from the IdP to the SP. In SAML, one identity provider may provide SAML assertions to many service providers. Conversely, one SP may rely on and trust assertions from many independent IdPs.

SAML does not specify the method of authentication at the identity provider; it may use a username/password, multifactor authentication, etc. A directory service, which allows users to login with a user name and password, is a typical source of authentication tokens (i.e., passwords) at an identity provider. Any of the popular common internet social services also provide identity services that in theory could be used to support SAML exchanges.” [132]

7.6.1.2 OpenID

“OpenID is an open standard (see [133]) of the consortium OpenID Foundation that allows users to be authenticated by certain co-operating sites (known as Relying Parties or RP) using a third party service, eliminating the need for webmasters to provide their own ad hoc systems and allowing users to consolidate their digital identities.

Users may create accounts with their preferred OpenID identity providers, and then use those accounts as the basis for signing on to any website which accepts OpenID authentication. The OpenID standard provides a framework for the communication that must take place between the identity provider and the OpenID acceptor (the "relying party"). An extension to the standard (the OpenID Attribute Exchange) facilitates the transfer of user attributes, such as name and gender, from the OpenID identity provider to

the relying party (each relying party may request a different set of attributes, depending on its requirements).

The OpenID protocol does not rely on a central authority to authenticate a user's identity. Moreover, neither services nor the OpenID standard may mandate a specific means by which to authenticate users, allowing for approaches ranging from the common (such as passwords) to the novel (such as smart cards or biometrics). The term OpenID may also refer to an identifier as specified in the OpenID standard; these identifiers take the form of a unique URI, and are managed by some 'OpenID provider' that handles authentication.

OpenID authentication is now used and provided by several large websites. Providers include Google,Yahoo!, PayPal, BBC, AOL, LiveJournal, MySpace, IBM, Steam, Sherdog, Orange and VeriSign." [134]

7.6.1.3 OAuth

"OAuth is an open standard (see [135]) for authorization. OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). It also provides a process for end-users to authorize third-party access to their server resources without sharing their credentials (typically, a username and password pair), using user-agent redirections." [136]

"The OAuth protocol was originally created by a small community of web developers from a variety of websites and other Internet services who wanted to solve the common problem of enabling delegated access to protected resources. The resulting OAuth protocol was stabilized at version 1.0 in October 2007, and revised in June 2009 (Revision A)...

In the traditional client-server authentication model, the client uses its credentials to access its resources hosted by the server. With the increasing use of distributed web services and cloud computing, third-party applications require access to these server- hosted resources.

OAuth introduces a third role to the traditional client-server authentication model: the resource owner. In the OAuth model, the client (which is not the resource owner, but is acting on its behalf) requests access to resources controlled by the resource owner, but hosted by the server. In addition, OAuth allows the server to verify not only the resource owner authorization, but also the identity of the client making the request.

OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end- user). It also provides a process for end-users to authorize third- party access to their server resources without sharing their credentials (typically, a username and password pair), using user- agent redirections.

For example, a web user (resource owner) can grant a printing service (client) access to her private photos stored at a photo sharing service (server), without sharing her username and password with the printing service. Instead, she authenticates directly with the photo sharing service which issues the printing service delegation-specific credentials.

In order for the client to access resources, it first has to obtain permission from the resource owner. This permission is expressed in the form of a token and matching shared-secret. The purpose of the token is to make it unnecessary for the resource owner to share its credentials with the client. Unlike the resource owner credentials, tokens can be issued with a restricted scope and limited lifetime, and revoked independently.

This specification consists of two parts. The first part defines a redirection-based user-agent process for end-users to authorize client access to their resources, by authenticating directly with the server and provisioning tokens to the client for use with the authentication method. The second part defines a method for making authenticated HTTP [RFC2616] requests using two sets of credentials, one identifying the client making the request, and a second identifying the resource owner on whose behalf the request is being made." [137]

7.6.2 Access control

7.6.2.1 XACML

"XACML stands for eXtensible Access Control Markup Language. The standard (see [138]) defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies.

As a published standard specification, one of the goals of XACML is to promote common terminology and interoperability between authorization implementations by multiple vendors. XACML is primarily an Attribute Based Access Control system (ABAC), where attributes (bits of data) associated with a user or action or resource are inputs into the decision of whether a given user may access a given resource in a particular way. Role-based access control (RBAC) can also be implemented in XACML as a specialization of ABAC.

The XACML model supports and encourages the separation of the authorization decision from the point of use. When authorization decisions are baked into client applications (or based on local machine userids and Access Control Lists (ACLs)), it is very difficult to update the decision criteria when the governing policy changes. When the client is decoupled from the authorization decision, authorization policies can be updated on the fly and affect all clients immediately.

Version 2.0 was ratified by OASIS standards organization on February 1, 2005. The first committee specification of XACML 3.0 was released August 10, 2010. The latest version, XACML 3.0, was standardized in January 2013." [139]

7.6.2.2 LDAP

"The Lightweight Directory Access Protocol (LDAP) is an application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network.

Directory services may provide any organized set of records, often with a hierarchical structure, such as a corporate email directory. Similarly, a telephone directory is a list of subscribers with an address and a phone number.

LDAP is specified in a series of Internet Engineering Task Force (IETF) Standard Track Request for Comments (RFCs), using the description language ASN.1. The latest specification is Version 3, published as RFC 4511.

For example, here's an LDAP search translated into plain English: "Search in the company email directory for all people located in Boston whose name contains 'Jesse' that have an email address. Please return their full name, email, title, and description."

A common usage of LDAP is to provide a "single sign-on" where one password for a user is shared between many services, such as applying a company login code to web pages (so that staff log in only once to company computers, and then are automatically logged in to the company intranet).“ [140]

LDAP also provides the mechanisms for maintaining the attributes associated with users and devices, which can be used in determining access privileges to secure resources.

7.6.3 Secure communications

7.6.3.1 S/MIME

“S/MIME (Secure/Multipurpose Internet Mail Extensions) is a standard for public key encryption and signing of MIME data... S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity, non-repudiation of origin (using digital signatures), privacy and data security (using encryption). S/MIME specifies the MIME type application/pkcs7-mime (smime-type "enveloped-data") for data enveloping (encrypting) where the whole (prepared) MIME entity to be enveloped is encrypted and packed into an object which subsequently is inserted into an application/pkcs7-mime MIME entity.

Before S/MIME can be used in any of the above applications, one must obtain and install an individual key/certificate either from one's in-house certificate authority (CA) or from a public CA. The accepted best practice is to use separate private keys (and associated certificates) for signature and for encryption, as this permits escrow of the encryption key without compromise to the non-repudiation property of the signature key. Encryption requires having the destination party's certificate on store (which is typically automatic upon receiving a message from the party with a valid signing certificate). While it is technically possible to send a message encrypted (using the destination party certificate) without having one's own certificate to digitally sign, in practice, the S/MIME clients will require you to install your own certificate before they allow encrypting to others.

A typical basic ("class 1") personal certificate verifies the owner's "identity" only insofar as it declares that the sender is the owner of the "From:" email address in the sense that the sender can receive email sent to that address, and so merely proves that an email received really did come from the "From:" address given. It does not verify the person's name or business name. If a sender wishes to enable email recipients to verify the sender's identity in the sense that a received certificate name carries the sender's name or an organization's name, the sender needs to obtain a certificate ("class 2") from a CA who carries out a more in-depth identity verification process, and this involves making inquiries about the would-be certificate holder...

Depending on the policy of the CA, your certificate and all its contents may be posted publicly for reference and verification. This makes your name and email address available for all to see and possibly search for. Other CAs only post serial numbers and revocation status, which does not include any of the personal information. The latter, at a minimum, is mandatory to uphold the integrity of the public key infrastructure." [141]

7.6.3.2 TLS

"Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide communication security over the Internet. They use asymmetric cryptography for authentication of key exchange, symmetric encryption for confidentiality and message authentication codes for message integrity. Several versions of the protocols are in widespread use in applications such as web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP).

In the TCP/IP model view, TLS and SSL encrypt the data of network connections at a lower sublayer of its application layer. In OSI model equivalences, TLS/SSL is initialized at layer 5 (the session layer) then works at layer 6 (the presentation layer): first the session layer has a handshake using an asymmetric cipher in order to establish cipher settings and a shared key for that session; then the presentation layer encrypts the rest of the communication using a symmetric cipher and that session key. In both models, TLS and SSL work on behalf of the underlying transport layer, whose segments carry encrypted data.

TLS is an IETF standards track protocol, first defined in 1999 and last updated in RFC 5246 (August 2008) and RFC 6176 (March 2011). It is based on the earlier SSL specifications (1994, 1995, 1996)... The TLS protocol allows client-server applications to communicate across a network in a way designed to prevent eavesdropping and tampering.

Since protocols can operate either with or without TLS (or SSL), it is necessary for the client to indicate to the server whether it wants to set up a TLS connection or not. There are two main ways of achieving this; one option is to use a different port number for TLS connections (for example port 443 for HTTPS). The other is to use the regular port number and have the client request that the server switch the connection to TLS using a protocol specific mechanism (for example STARTTLS for mail and news protocols).

Once the client and server have decided to use TLS, they negotiate a stateful connection by using a handshaking procedure. During this handshake, the client and server agree on various parameters used to establish the connection's security... If any one of the steps fails during the negotiation process, the TLS handshake fails and the connection is not created." [142]

7.6.4 System Architecture Considerations

7.6.4.1 Prerequisite for the Allocation of GS1 Identification Keys: GS1 Global Company Prefix (GCP)

The GCP is allocated by GS1. It is part of every GS1 Identification Key and guarantees its uniqueness.

7.6.4.2 Product identification: Global Trade Item Number (GTIN):

The GTIN [9] identifies each product or service by its unique number that is generated based on the GCP of the brand owner, brand co-operative or producer. The GTIN-structure is 14-digit with N1 and N14 being

the indicator and check digit respectively. The remaining digits are used by the GS1 Company Prefix and the item reference (with a variable number of digits assigned to each). Depending on the application surrounding N1 up to N6 can assume the value 0.

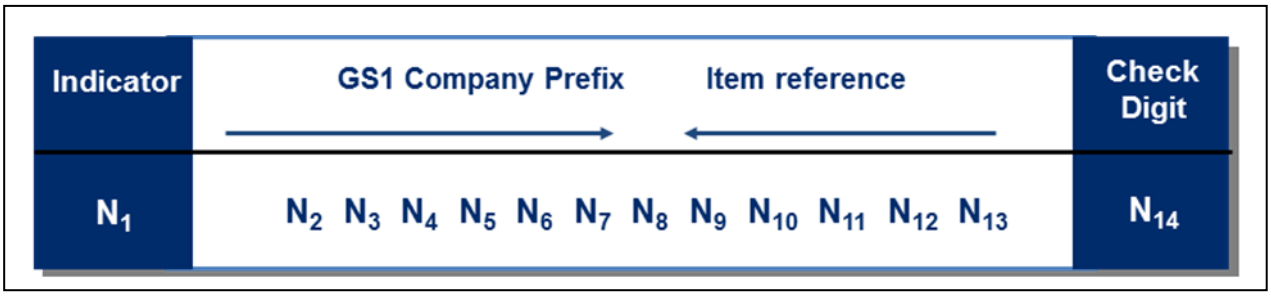


Figure 7: GTIN Structure

7.6.4.3 Asset Identification: GRAI

Global Returnable Asset Identifier [147]: Identifies any returnable containers or packaging that will be returned to their source.

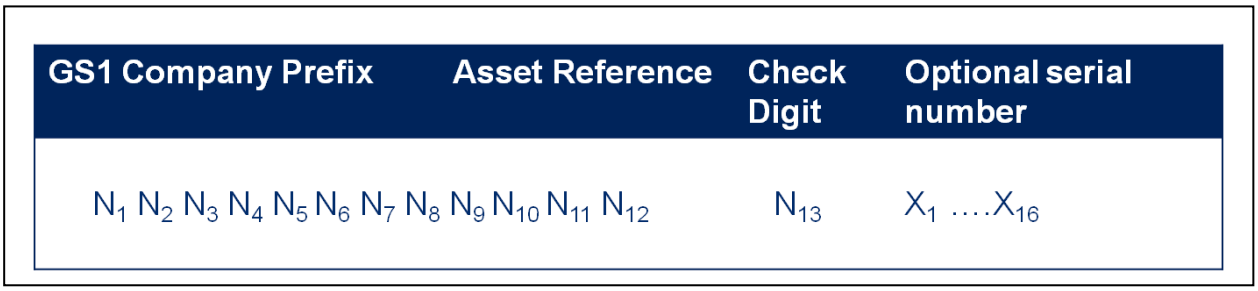


Figure 8: GIAI Structure

Assets, for example crates or boxes belonging to the same type are identified by the same GRAI. GRAIs can be serialized by their optional serial number.

7.6.4.4 Identification of logistic units: SSCC

The Serial Shipping Container Code (SSCC) [11] identifies an item of any composition made up for transport or storage.

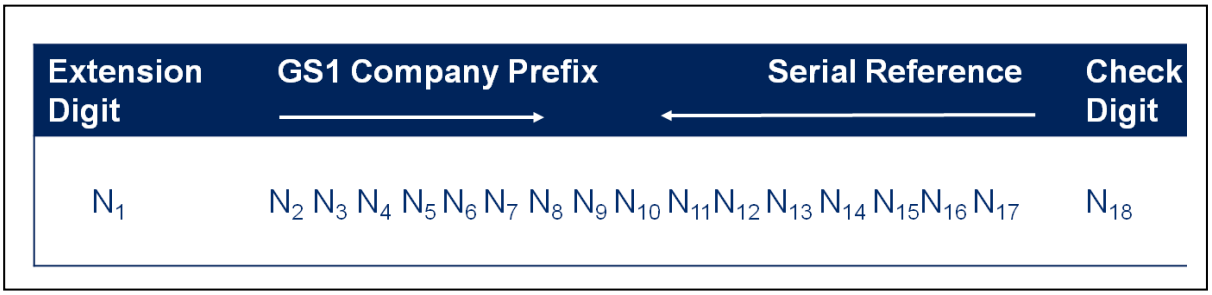


Figure 9: SSCC Structure

7.6.4.5 Global Location Number

Global Location Number (GLN) [10]: The GLN is the worldwide unique identification of each company or physical location within a company.

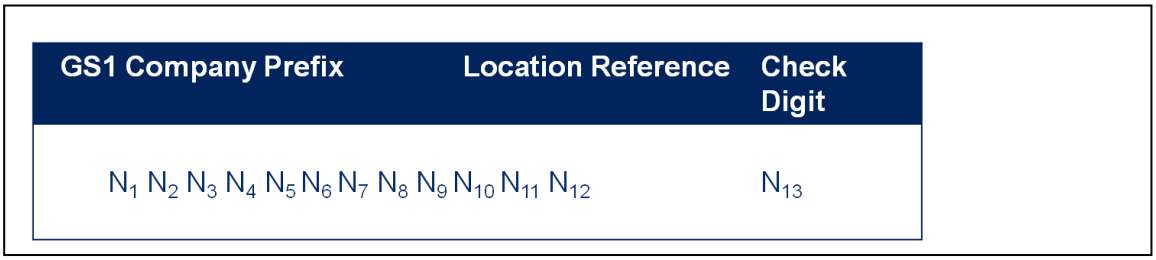


Figure 10: GLN Structure

7.6.4.6 GS1 Application Identifier System

The following table show examples of data elements for both, multi-industry as well as sector specific applications.

Application Identifier	Content
01	Global Trade Item Number
10	Lot/batch number
15	Best before date
20	Serial number
251	Source reference (Number of earmark)
422	Country of origin (birth)
423	Country of initial processing (fattening)
424	Country of Slaughtering
425	Country of dissection
426	Country of all processing stages (if the same)
7002	UN/ECE Meat Carcasses and Cuts Classification
7030 – 7039 (7031 (7032 (7033	Approval Number of Processor with Three-Digit ISO Country Code Approval Number for 1st stage of dissection) Approval Number for 2nd stage of dissection) Approval Number of Processor)
8003	Global Returnable Asset Identifier
8005	Price per Unit of Measure
8006	Identification of the Components of a Trade Item

Table 3: List of GS1 Application Identifiers relevant for the FIspace Project (excerpt)

Exemplary data string for a project identifier (GTIN), the best before date (24 August 2013) and the batch (1249) could be:

(01)04012345123456(15)130824(10)1249

Figure 11: GS1-128 label example

A complete list of all admitted GS1 Application Identifiers for all domains can be found in the GS1 General Specifications [148].

7.6.4.7 EPCIS

Example data set

```
<ObjectEvent>
<eventTime>2013-02-18T06:41:50Z</eventTime>
<recordTime>2013-02-18T06:41:50Z</recordTime>
<eventTimeZoneOffset>+01:00</eventTimeZoneOffset>
<epcList>
<epc>urn:epc:id:sgtin:4000001.001602.112</epc>
<epc>urn:epc:id:sgtin:4000001.001602.130</epc>
</epcList>
<action>ADD</action>
<bizStep>urn:epcglobal:cbv:bizstep:commissioning</bizStep>
<disposition>urn:epcglobal:cbv:disp:active</disposition>
<readPoint> <id>urn:epc:id:sgln:4000001.00005.0</id> </readPoint>
<bizLocation> <id>urn:epc:id:sgln:4000001.00002.0</id> </bizLocation>
<bizTransactionList>
<bizTransaction
type=”urn:epcglobal:cbv:btt:desadv>http://example.com/desadv/9876</bizTransaction>
</bizTransactionList>
</ObjectEvent>
```

Figure 12: Example data set

7.6.4.8 EANCOM and GS1 XML

- EANCOM®

  EANCOM® [149] is a GS1 subset of the UN/EDIFACT standard (United Nations Electronic Data Interchange for Administration, Commerce and Transport), which comprises a set of internationally agreed standards, directories and guidelines for the electronic interchange of data.

  EANCOM® is fully compliant to UN/EDIFACT. It provides the collection of only those message elements which are needed by the business application and required by the syntax (mandatory elements). Omitted are optional elements covering very specific business requirements not relevant for GS1 users.

  EANCOM® incorporates into the electronic messages the GS1 standards of physical identification of trade items, logistics units and the Global Location Numbers identifying the trading partners. It allows integrating the physical flow of goods with related information sent by electronic means.

  The EANCOM® messages are equivalent of traditional paper business documents. Messages available in the EANCOM® standard cover the functions required to complete a trade transaction:

  - messages which enable the trade transaction to take place, e.g. price catalogue, purchase order, invoice, etc;

- messages used to instruct transport services to move the goods;
- messages used in settlement of the trade transactions through the banking system.

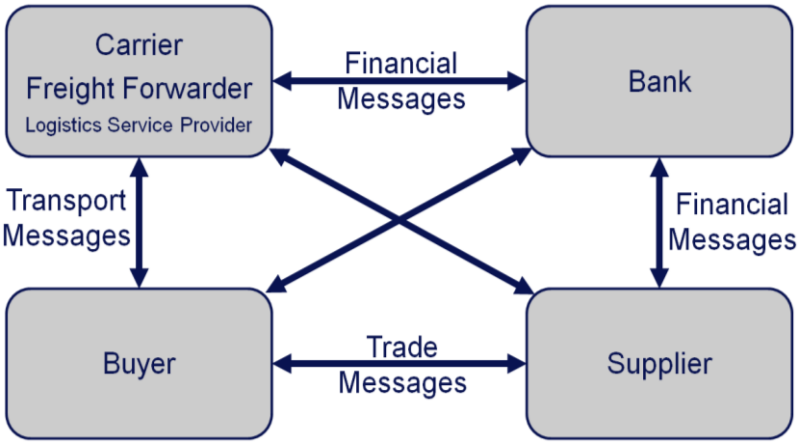


Figure 13: Electronic Messages Overview

- GS1 XML Interface Description

  XML is an acronym for "eXtensible Markup Language". The XML by is designed for information exchange over the internet. The corresponding specification (see [150]) is provided by the World Wide Web Consortium (W3C) as an open standard.

  GS1 has designed an XML grammar for use in Electronic Data Exchange, the GS1 XML Interface Description. GS1 XML [149] is designed in such a way that the messaging is transport agnostic. GS1 supports reliable and secure messaging via the use of the messaging protocols AS1, AS2, AS3, AS4 and ebMS, as well as other transport protocols. It is very simple to exchange GS1 XML documents using any technical solution or profile, such as Web Services.

  The GS1 XML messages are developed using the business process modelling methodology. First, the business process is described, including identification of business data that need to be exchanged between the main parties. This information is then mapped to the electronic messages. Thus, the GS1 XML messages are not always equivalent of traditional paper business documents.

  The messages available in the GS1 XML standard cover the following areas of the supply chain:

  - Data Synchronisation messages that enable sending information about the trade item attributes and support its automated synchronisation between business partners, using the Global Data Synchronisation Network (GDSN)
  - Messages used to order goods and respond to this order;
  - Messages used to announce the despatch of goods and confirm their receipt
  - Messages requesting payment for the goods sold and informing about the payment being sent
  - Messages for planning and execution of transport
  - Messages supporting automated replenishment of goods

GS1 XML standards support both Downstream (between the consumer goods manufacturers and retail) and Upstream (between the consumer goods manufacturers and their suppliers of raw material, packaging, etc.) communication.

GS1 XML standards provide solutions for multiple sectors using the same XML business message. GS1 XML is an international standard, and has been implemented in 33 countries, by more than 22.000 companies. The international network of GS1 Member Organisations (including GS1 China), covering more than 150 countries, provides support, documentation and training in local languages.

Note: data fields for fishing domain, including the legally required catch date and the vessel ID are under development within the GS1 community

7.7 Industry-specific Standards

7.7.1 Industry-specific Identification Specifications

7.7.1.1 ISO 11784 Code Structure of Animal Identification

RFID Code Structure:

- Animal bit: indicating if the transponder is intended for animal identification purposes;
- Country code: a 3 digit number referring to the unique ISO 3166 country number (000-899). The use of country coded transponders is restricted to countries that have a competent authority responsible for the registration and granting of IDcodes. It is the responsibility of the competent authority to maintain the uniqueness of the numbers. Countries without competent authority shall not use transponders with a country code. In these countries so called manufacturer coded transponders (900-998) shall be used. The manufacturer of the transponders is in this case responsible for maintaining unique ID codes;
- Identification code: a 12 digit number that is in combination with the country/ manufacturer code unique worldwide for all animals. The idea of the ISO 11784 standard is that the number itself should not carry any information (e.g. like farm number, breeding organisation or region code), because this leads to inefficient use of numbers. Information in relation to the animal shall be stored in databases;
- Retag counter: in some cases an animal loses the tag or the tag does not function anymore. In this case the owner of the animal has the possibility of retagging the animal with the same ID code. The retagging with the same ID code shall be registered in the database and also in the transponder. When issuing a new ID code the retagging number shall be set to ‘0’. At every retagging the retag counter shall be incremented. The retag counter offers 7 retagging possibilities. In case of anyfurther losses, a new number shall be granted to the animal. The use of retagging is only allowed in combination with country coded transponders. In case of a manufacturer code, the user information field should be set to ‘0’;
- User information field: The use of the user information field is only allowed in combination with the country code. The 2 digits field shall be set to ‘00’ in case of a manufacturer coded transponder. When used in combination with the country code the code of the user information field should be coded based on the specifications of the competent authority;

- Trailer bit: this bit shall be set in case information is written in the trailer of the transponder code, otherwise this bit shall be ’0’;
- RUDI-bit: this bit shall be set if a transponder is of the advanced LF transponder type, in case of a non advanced LF transponder the bit shall be ‘0’;
- Reserved field: This field is reserved for future use; all bits in this field should be set to ‘0’.

7.7.1.2 Identification and Registration of Pigs

The system of identification and registration [151] of pigs is based on the following elements:

- an eartag or a tattoo;
- the keeping of a register on each holding;
- a national computer database in accordance with Decision 2000/678/EC [151].

The holding register contains at least the following information:

- the country code and the identification number consisting of not more than 12 digits (apart from the country code);
- address of the holding;
- name and address of the person responsible for the animals;
- the geographic co-ordinates or equivalent geographic indication of the holding;
- a data field where it is possible for the competent authority to enter sanitary information, for example restrictions on movements, status or other relevant information in the context of Community or national programmes.)

In the following, three examples are given of regulation implementation on national level as there are differences:

In Germany, relevant EU regulations are implemented in [152]. The eartag has to be in black letters on white background (§ 39). The label contains the characters „DE“ (for Germany), the ID of the administrative district (same as for car licence plate) and the last seven digits of the register number of the holding. The register number itself consists of the 8-digit municipal code and the 4-digit ID of the holding.

In Spain [154], the each pig has to be marked with an ear tag or a tatoo. The code consists of the ID of the municipality (three digits maximum), the acronym of the province (e.g. TE-TF: Sta. Cruz de Tenerife, V: Valencia), and a maximum of seven digits identifying the holding.

In France [155], the holding ID consits of the two characters “FR” (for France), the two digit number of the administrative district (“code du départements”, e.g. “33” for Gironde, “75” for Paris) and three characters or digits for the holding unique within the department. In case of a piglet leaving its birth holding, it has to be marked with an ear tag or ear tatoo with the birth holding ID. A pig going for slaughter has to be marked with the feed holding ID on its front shoulder. Breeding animals are marked with their birth holding ID and an additional individual number consisting of one digit for the year of birth and a consecutive number.

7.7.1.3 Identification and registration of ovines and caprines

The eartags and the other means of identification must contain the following characters [156]:

- the first characters identify the Member State of the holding where the animal was first identified. For this purpose two-letter or three-digit country codes are to be used in accordance with ISO 3166 (e.g. “NL” or “528” for the Netherlands),
- the country code is followed by an individual one of no more than 13 digits.

The electronic identifier must conform to the following technical characteristics:

- read-only passive transponders applying HDX- or FDX-B technology, complying with ISO standards 11784 and 11785,
- electronic identifiers must be readable by reading devices, complying with ISO standard 11785, capable of reading HDX and FDX-B transponders,
- the reading distance for portable readers must be a minimum of 12 cm for eartags and a minimum of 20 cm for ruminal boluses, and, for stationary readers, a minimum of 50 cm for both eartags and ruminal boluses.

The holding register must contain at least the following information:

- the identification code of the holding,
- the address of the holding and the geographical coordinates or equivalent indication of the geographical location of the holding,
- the type of production (meat, wool, dairy),
- the result of the latest inventory and the date on which it was carried out,
- the name and address of the keeper,
- additional data in the case of animals moving on or off the holding

and for each animal the following up-to-date information:

- the identification code of the animal,
- the year of birth and date of identification,
- the month and the year of death of the animal on the holding,
- the race and, if known, the genotype.

7.7.1.4 Identification and labelling of bovine animals

In France [157], the ear tags for bovine animals are salmon colored. The label carries the two letters FR (for France) followed by a 10-digit number. The first two digits are the number of the administrative district ("chiffre du département"), the next four digits refer to the ID of the holding, the last four digits (which have to be larger) are an individual working number, which is consecutively allocated to each newborn calf.

In Germany, [151] applies. § 27 specifies that the eartags have to be in black on yellow background. The first eartag has to contain a bar code. The second eartag might carry a read-only passive transponder, complying with ISO standards 11784 and 11785. The ID itself is the country code “DE” for Germany and

a 10-digit number, consisting of 2 digits for the federal state (e.g. 06 for Hessen) and an 8-digit individual number. For each animal, the following data are registered (§28):

- Name and address of the owner,
- the register number of the holding,
- the ID of the ear tag,
- date of birth,
- sex,
- race,
- eartag ID of the mother,
- country of origin (if applicable).

In Great Britain [158], [159], [160], the primary tags are yellow, distance-readable tags, the 'secondary' tag must have the same information as the primary tag, but may be metal, plastic and button tags of any colour. Both eartags shall be marked with the following information: the logo of the competent authority which allocated the eartag, i.e. the crown insignia and the letters "UK", followed by the unique lifetime identification number consisting of a six digit herd number, followed by a six digit animal code, comprising a single check digit and a five digit animal number. A bar code may be included on an official tag for animals that are being exported. Cattle born in GB do not need to be identified with ear tags that have a bar code.

7.7.1.5 Identification of fish

Data according to Article 14 of Council Regulation (EC) No 1224/2009 [27] a fishing logbook has to kept by the master of each fishing vessel:

(a) the external identification number and the name of the fishing vessel;

(b) the FAO alpha-3 code of each species and the relevant geographical area in which the catches were taken;

(c) the date of catches;

(d) the date of departure from and of arrival to port, and the duration of the fishing trip;

(e) the type of gear, mesh size and dimension;

(f) the estimated quantities of each species in kilograms live weight or, where appropriate, the number of individuals;

(g) the number of fishing operations.

Specific rules apply for fisheries subject to a Community regime of fishing effort. Currently, the fishing logbook might be kept in paper form, but in the near future, all vessels with a length of more than 12 m shall be equipped with an electronic logbook.

All lots of fisheries and aquaculture products shall be traceable at all stages of production, processing and distribution, from catching or harvesting to retail stage. The minimum labelling and information requirements for all lots of fisheries and aquaculture products shall include ([27], Article 58):

(a) the identification number of each lot;

(b) the external identification number and name of the fishing vessel or the name of the aquaculture production unit;

(c) the FAO alpha-3 code of each species;

(d) the date of catches or the date of production;

(e) the quantities of each species in kilograms expressed in net weight or, where appropriate, the number of individuals;

(f) the name and address of the suppliers;

(g) the information to consumers provided for in Article 8 of Regulation (EC) No 2065/2001: the commercial designation, the scientific name, the relevant geographical area and the production method;

(h) whether the fisheries products have been previously frozen or not.

The 3-alpha identifier is a unique code made of three letters that is widely used for the exchange of data with national correspondents and among fishery agencies [161].

8 Appendix 2: Outline of Semantic Technologies

As the term is used here, semantic technologies are those standards and related tools created for the Semantic Web [47], [48], [53]. The underlying characteristic of these technologies is the formal representation of domains of interest (agriculture, for instance, or chemistry) using

1. agreed upon conceptual models,
2. expressed in formal languages,
3. having a rigorous (and machine processable) syntax and semantics.

The languages are designed so that the data encoded in them can be readily exchanged over the Web, and also so that data can be combined from many different sources. The core formalisms—the Resource Description Framework (RDF) [163], [164] and the Web Ontology Language (OWL) [165] —are based on symbolic logic and are descendants of the logical formalisms developed during decades of research into artificial intelligence. What is expressed in them amounts to a collection of formal assertions about things in the chosen domain. The point is to structure data so that it can be better organised, exchanged, and otherwise more productively used than was previously possible. This is due to types of standards 1) to the use of agreed upon concepts and vocabularies (standard ontologies), and 2) to the use of formal languages readily processed by machines (standard knowledge representation and data representation languages).

A stereotypical use case is the publishing of information on companies and organizations. Using structured vocabularies, each organization could publish—on its own Web-page—information about its location, operating hours, and services provided. Since the information is in a structured format using agreed upon terms, a Web-crawler can collect it and compile a database, making it possible to search in a very precise fashion for organizations providing particular services. Something similar to this is in fact already done in by websites which follow the schema.org principles (with partial use of Semantic Web standards in the form of RDFa). With it, publishers of Web content can create structured product descriptions and reviews, personal profiles, recipes, etc. This sort of improved search is one use of semantic technologies, but there are others. Common to them all is the use of agreed upon terms and machine processable languages to make assertions about some domain.

8.1 Uniform Resource Identifiers

The dominant representational languages of the Semantic Web are RDF (the Resource Description Language) and OWL (the Web Ontology Language). Both languages are assertional in nature— they allow one to make assertions about things. Both languages make use in an essential way of uniform resource identifiers (URIs). A URI is a character string used to identify a thing, generically called a resource (often the resource is something accessible on the Internet such as a Web page). Syntactically, URIs are identical to the uniform resource locators (URLs) used to identify Web pages. For instance,

http://dbpedia.org/resource/Port-au-Prince

is a valid URI. It is not necessary, however, that a URI be associated with any Web page. URIs are essentially names for things (whether physical or abstract). Like names in natural language, the internal

structure of a URI need not have any significance. However, it is common to specify a namespace URI such as

http://dbpedia.org/resource/

to delineate terms of a common vocabulary (all terms in the vocabulary would use the same prefix). Furthermore, it is common to abbreviate a URI, e.g. writing dbpedia: Port-au-Prince, where dbpedia has been previously specified as a shorthand for http://dbpedia.org/resource/.

URIs are used as the naming mechanism on the Semantic Web because they are part of the XML (Extensible Markup Language) standard [166]. XML is the dominant language for creating structured documents on the Web. It has a simple syntax and is very readily parsed by computers, and it is very widely used. Though not discussed here, Semantic Web languages can be and often are expressed in XML (though it is not essential to do so). This allows easy transmission of, e.g., RDF and OWL documents using existing Web technologies.

Resource Description Framework (RDF)

The Resource Description Framework (RDF) is the basic language of the Semantic Web. With it, one may make simple statements about resources. Each statement consists of three elements (the statements are called triples):

- subject: a URI
- predicate: a URI
- object: a URI or else a literal, e.g., a number or character string (“Franz Farmer”).

The subject and predicate URIs are intended to refer to things (resources) and binary relationships between things (or between things and literals), respectively. Literals are used to stand for constant values, e.g., the text comprising a book title. The following RDF triples assert, intuitively, that Port au Prince is the capital of Haiti and that it has the given latitude and longitude.

dbpedia:Haiti	dbprop:capital	dbpedia:Port-au-Prince
dbpedia:Port-au-Prince	geo:lat	18.53333282470703
dbpedia:Port-au-Prince	geo:long	-72.33333587646484

The above triples are expressed in Turtle (Terse RDF Triple Language) (Beckett and Berners-Lee, 2011), a commonly used syntax for writing RDF. In a complete Turtle document, a prefix must be declared before it can be used in triples, and so the following should also be included with the above triples.

@prefix dbprop: <http://dbpedia.org/property/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

A single RDF triple can be represented as a simple, two–node, directed graph—the subject and object become nodes in the graph, and the predicate becomes an edge leading from the subject to the object. A collection of RDF triples can viewed as a larger graph composed from the graphs for each triple.

As originally envisioned, RDF was to be used for annotation of Web resources with metadata. A set of RDF triples could be used to specify the author, creation date, and contents of a spreadsheet document or Web page, for instance, or the latitude and longitude at which a sensor measurement was recorded. Annotation is not the only use of RDF, however. E.g., in the example triples above, there is really no document or record that is annotated (or if there is, it is odd to assert of it that it has a latitude and longitude). Instead, RDF is being used here as a modelling language, though a very simple one. There is an RDF namespace [162], and the URIs defined for it allow one to specify, e.g., lists of things. However, the namespace consists of only a handful of URIs and what can be expressed using them is limited.

One feature is worth discussing, however. It is possible in RDF to explicitly refer to an RDF statement—this is called reification. Below a named triple ex:triple1 (the identifier is arbitrary) is described.

ex:triple1	rdf:type	rdf:Statement.
ex:triple1	rdf:subject	dbpedia:Haiti .
ex:triple1	rdfs:property	dbprop:capital .
ex:triple1	rdf:object	dbpedia:Port-au-Prince.

Given that the triple has been named, additional statements can be made about it (e.g., that its source is a particular document). This is significant from the standpoint of data integration and provenance. Note that, intuitively, making assertions about a statement is not the same thing as asserting the statement itself. It is the same in RDF—the two are essentially unrelated.

Arguably, the utility of RDF comes from its graph-like nature, as it allows disparate pieces of data to be assembled into a whole. Multiple authors can create RDF triples, much as multiple authors can create Web pages. Provided the authors use at least some of the same URIs, the triples can be gathered together to form a connected graph.

8.2 RDF Schema (RDFS)

RDF Schema (RDFS) [167] significantly extends RDF, allowing one to define RDF vocabularies. Specifically, RDFS allows one to specify that some URIs refer to entities of the following types:

- classes of things;
- individuals (instances of classes); and
- properties (binary predicates/relations between things).

Furthermore, it allows one to specify class and property hierarchies—i.e., that one class (e.g., Bulldozer), is a subclass of another (HeavyVehicle)—and declare that an individual is an instance of a particular class. One may also specify a domain and range for properties. These extensions to RDF are accomplished through the use of a small set URIs (e.g., rdfs:subClassOf) having a fixed meaning.

ex:Bulldozer	rdf:type	rdfs:Class .
ex:TrackedVehicle	rdf:type	rdfs:Class .
ex:Bulldozer	rdfs:subClassOf	ex:TrackedVehicle .
ex:bd49501138	rdf:type	ex:Bulldozer .

The URI rdf:type is used to declare a class and assert that an individual is a member of a class. Above, ex:Bulldozer is an instance of rdfs:Class, and ex:bd49501138 is an instance of class ex:Bulldozer. Also, ex:Bulldozer is a subclass of ex:TrackedVehicle. The URI rdfs:subClassOf has a fixed meaning in RDFS. It is transitive; in the above example, any in- stance of ex:Bulldozer will be inferred to be an instance of ex:TrackedVehicle (by software properly designed to work with RDFS).

The ability to define class hierarchies, and to make assertions that individuals are instances of particular classes, gives RDFS significant advantages over plain RDF. However, RDFS itself is quite limited. For instance, although one can declare in RDFS that the classes Plant and Animal are both subclasses of LivingThing, one cannot express that they are disjoint classes (that is, that they have no members in common). In general, RDFS lacks many of the capabilities of even simple Boolean logic (including nega- tion, and so it is impossible to say “no plant is an animal”). RDFS is a “lightweight” modelling language.

8.3 The Web Ontology Language (OWL)

The Web Ontology Language [165] is significantly more sophisticated. Like RDFS, OWL also allows one to distinguish between individuals, classes, and properties. However it allows more complex assertions to be made about them. OWL is based on a class of logic called a description logic (DL) [168].

A basic syntactic unit of a description logic is a class description. Atomic class descriptions can be speci- fied, e.g., Man and Woman. Using logical connectives—e.g., for And (Intersection), Or (Union), Not (Complementation)—simpler descriptions are composed into more complex descrip- tions. In each case, a class description describes a set of things. Examples of the basic sorts of descriptions available in OWL (expressed in the OWL functional syntax) are shown in Table 4.

Class descriptions are used to make logical assertions in OWL. Terminological axioms assert relation- ships between classes, stating for instance that one class is a subclass of another, or that two classes are equivalent, or that they are disjoint. Note that such axioms are universal (“All men are mortal”; “No man is an island”). As in RDFS, such axioms can be used to create a class hierarchy. Axioms stating that one class is a subclass of another are called class inclusion axioms. OWL also allows one to define property inclusion axioms; for technical reasons, these are more restricted.

ObjectIntersectionOf(ex1:Male ex1:Married)	married males
ObjectUnionOf(ex1:Male ex1:Female)	things male or female.
ObjectComplementOf(ex1:Cat)	non-cats
ObjectSomeValuesFrom(ex1:hasChild ex1:Male)	things with at least one male child.
ObjectAllValuesFrom(ex1:hasChild ex1:Female)	things with only female children.
ObjectMaxCardinality(2 ex1:hasChild ex1:Male)	things with at most 2 male children.
ObjectMinCardinality(2 ex1:hasChild ex1:Male)	things with at least 2 male children.

Table 4: Example class descriptions in OWL. Each description denotes a set of individuals. Ab- breviated URIs are used. “Object” indicates that data literals are not involved.

Terminological axioms are used to model the general features of a domain of interest. The axioms chosen constitute a so-called ontology of the domain—a model of the domain expressed in a formal language.

Assertional axioms are another variety of axiom. These make assertions about particular individuals, e.g., that Port-au-Prince is a city, (this is a class assertion) or that Port-au-Prince is the capitol of Haiti (this is a property assertion). A collection of assertional and terminological axioms is shown inTable 5.

ClassAssertion(ex1:Male ex1:john)
SubClassOf(ex1:Person ObjectUnionOf (ex1:Female ex1:Male))
EquivalentClasses(ex1:Person ex1:Human)
DisjointClasses(ex1:Cat ex1:Dog ex1:Bird)
ObjectPropertyAssertion(ex1:hasHusband ex1:mary ex1:john)
SameIndividual(ex1:john ex2:bill)
DifferentIndividuals(ex1:john ex1:mary ex1:frank)

Table 5: Examples of assertional and terminological axioms in OWL.

An ontology together with a set of assertional axioms is called a knowledge base (KB). In the following, “knowledge base” will be used to refer to any set of assertions in a given mod- elling language. Also, unless otherwise stated, “assertion”, “axiom”, and “statement” will be used interchangeably.

8.4 Formal Semantics

RDF/RDFS and OWL are logic-based languages, and as such they have a formal semantics (technically, a model-theoretic semantics similar to that defined for mathematical logic). The semantics specifies rigorous rules for constructing interpretations of the assertions in a knowledge base. Though the precise nature of interpretations and the rules for constructing them are unimportant for the present discussion, what is important to understand is that each assertion is either true or false according to a given interpretation. Each interpretation assigns a truth value to each assertion, and different interpretations might assign different values to a given assertion. An interpretation in which all of the assertions are true is called a mod-

el of the knowledge base. A knowledge base that has a model is said to be consistent (it is free of internal contradictions).

Importantly, the formal semantics also defines a logical consequence relation which governs when an assertion “follows from” a knowledge base. Formally, If K is a knowledge base and P is an assertion, then P is a consequence of K, written K |= P if and only if every model of K is a model of P.

The importance of the semantics and associated consequence relation cannot be overstressed. The semantics specifies what can be correctly inferred from the knowledge base. The semantics determines correct answers to queries posed to a knowledge base. Importantly, it also governs the development of algorithms allowing machines to reason over the knowledge base.

Some of the basic computational tasks performed on a given knowledge base are listed below.

1. consistency: Determining whether the knowledge base is consistent.
2. satisfiability: Determining whether a given class in the knowledge base can have instances.
3. class subsumption: Determining whether one class is a subclass of another.
4. equivalence (disjointness): Determining whether two classes are equivalent (disjoint).
5. instance checking: Determining whether a given individual is a member of a given class.

These are low level computational tasks. They form the basis for tasks that are of interest to users, however. For instance, if, in a knowledge base storing employee lists of hospitals, statements also appear asserting that hospitals are a type of health care facility, it becomes possible to query the knowledge base for employees of healthcare facilities and receive the correct results. This is so despite the fact that it has not been explicitly asserted, for each hospital, that it is a healthcare facility. Such reasoning, simple though it is, can be useful.

8.5 Computational Complexity; OWL Profiles

It was stated previously that OWL is a more sophisticated modelling language than RDF or RDFS. What this means is that OWL can more faithfully model a larger class of states of affairs than RDF or RDFS. This greater expressiveness, however, comes with a computational cost. OWL has greater complexity, meaning that more computational resources are needed to work with OWL ontologies. This is a general characteristic of modelling languages: Greater expressiveness implies greater computational complexity.

The current version of OWL is called OWL 2. It has a formal semantics based on description logics. An alternative graph–based semantics (more in line with that of RDFS) also exists which is not equivalent to the model-theoretic semantics. And so one may speak of two forms of OWL 2: OWL 2 DL, which specifies consequences according to the model theoretic semantics; and OWL 2 Full, which extends the semantics of RDFS.

OWL 2 Full is computationally undecidable. What this means is that no algorithms exist (even in theory) that are guaranteed to determine—using a finite amount of computational resources— whether or not, for instance, a given knowledge base is consistent. OWL 2 DL, in contrast, is decidable, achieving this by placing syntactic restrictions on knowledge bases. It nevertheless is computationally too complex for many real-world applications. The time required to solve certain problems is an exponential (actually, worse) function of the size of the knowledge base.

Because of this, three syntactic fragments, or profiles, of OWL 2 have also been specified [169]. The profiles—OWL 2 EL, OWL 2 QL, and OWL 2 RL—have useful applications and are of a lower computational complexity than OWL 2 DL. Low complexity is achieved by eliminating features—such as the ability to make negative assertions (“It is not the case that ...”) —from the logics. OWL 2 EL is based on the description logic EL++, which has been used to define ontologies in the life-sciences. Very large ontologies in this logic can be very quickly analyzed. OWL 2 QL is “designed so that data (assertions) that is stored in a standard relational database system can be queried through an ontology via a simple rewriting mechanism, i.e., by rewriting the query into an SQL query...” (Motik et al., 2009). OWL 2 RL is designed so that its reasoning problems can be decided by rule-based systems, i.e. systems designed to process rules of the form “If A1 and A2 and ... An, then B”. Many such systems exist (a programming paradigm, logic programming, is devoted to them).

It is important to be aware of issues of computational complexity. They impact the systems that are developed and used in the field.


FIspace
Business Collaboration