DI1.4: Assessment of existing standards

Written by:

Ajith Parlikad and James Brusey, Cambridge University

Altuğ Metin, InMediasP

Daniel Barisic, Infineon

Kary Framling, HUT

Juergen Anke, SAP

DELIVERABLE NO	DI1.4: Assessment of existing standards
DISSEMINATION LEVEL	**PUBLIC**
DATE	15. August 2006
WORK PACKAGE NO	WP I1: Standardisation
VERSION NO.	0.3
ELECTRONIC FILE CODE	promise di1 4 assessment of standards_final
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT	This report focuses on several key existing standards that have been identified as being important to PROMISE. These standards are assessed according to how well they fit the desired requirements. This report provides a basis for making a decision about which existing standards will be used.

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE** (dd.mm.yyyy)
SUBMITTED (author(s))	James Brusey	31.08.2006
VU (WP Leader)	Duncan McFarlane	dd.mm.yyyy
APPROVED (QIM)	Dimitris Kiritsis	dd.mm.yyyy

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
06.07.2006	0.1	J. Brusey	Initial version
14.07.2006	0.2	J. Brusey	Updates suggested during teleconference
26.07.2006	0.3	A. Metin	Added sections: STEP, PLCS, STEP NC, MANDATE, PLM XML, ANSI/ISA-95 and References
01.08.2006	0.4	A. Metin	Conclusions and corrections added
29.08.2006	0.5	A. Parlikad	Final version

Author(s)' contact information

Name	Organisation	E-mail	Tel	Fax
James Brusey	Cambridge University	jpb54@cam.ac.uk	441223765605	441223338078
Altuğ Metin	InMediasP	metin@inmediasp.de	+493302559409	+493302559124
Ajith Parlikad	Cambridge University	aknp2@cam.ac.uk	441223765606	441223765597
Daniel Barisic	Infineon			
Kary Framling	HUT			
Juergen Anke	SAP			

Table of Contents

List of figures

List of Tables

Abbreviations

UPnP: Universal Plug and Play

STEP: Standard for the Exchange of Product model data

PLCS: Product Life Cycle Support

WWAI: World Wide Article Information

RFID: Radio Frequency IDentification

WSDL: Web Services Definition Language

ISO: International Standards Organisation

EAN-UCC: European Article Numbering-United Code Council

TDS: Tag Data Standards

TDT: Tag Data Translation

ONS: Object Name Service

EPC: Electronic Product Code

EPCIS: EPC Information Service

ALE: Application-Level Events

1 Introduction

The main aim of this report is to examine the various existing standards relevant to PROMISE, and assess their appropriateness for the PROMISE architecture.

DI1.1 provided a summary of international standards that are considered relevant to the PROMISE project. DI1.2 dealt with the definition of standardisation domains, and for each of these domains candidate standards were nominated following on from DI1.1. In DI1.3, we presented the strategy for standardisation efforts in the PROMISE project, whereby it was concluded that standardisation should be a central activity within the project. In addition, we also concluded that PROMISE should avoid creating too many new standards or creating standards for elements that do not need to be standardised.

In keeping with this conclusion, the project would focus more on existing standards and supporting other standards bodies in extending the standards if required. This report establishes the basis for recommendation of the appropriate standard for each of the components and interfaces in the PROMISE architecture. The final recommendations will be presented in the next deliverable DI1.5.

The report is structured as follows: In section 2, we describe the methodology for assessing the standards for PROMISE – based on the PROMISE architecture, specially examining the different components and interfaces, and discuss the proposed usage of standards pertaining to each of the components and interfaces. In section 4, we assess some of the existing standards separately for appropriateness for the PROMISE architecture. Finally, section 5 concludes this report.

2 Assessment methodology

The matrix in Figure 1 forms the basis for the analysis of the applicability of various standards to the PROMISE architecture (see Figure 2). This matrix is derived from the realisation that the use of many low-level standards, such as Unicode or XML, is inevitable and requires little or no comment. This matrix is designed to focus attention on higher layer standards, especially where these imply the lower layer ones. For example, the Web Services standard builds on standards such as XML, TCP/IP, and HTTP. Where PROMISE makes use of Web Services, the use of XML is implied and need not be stated explicitly.

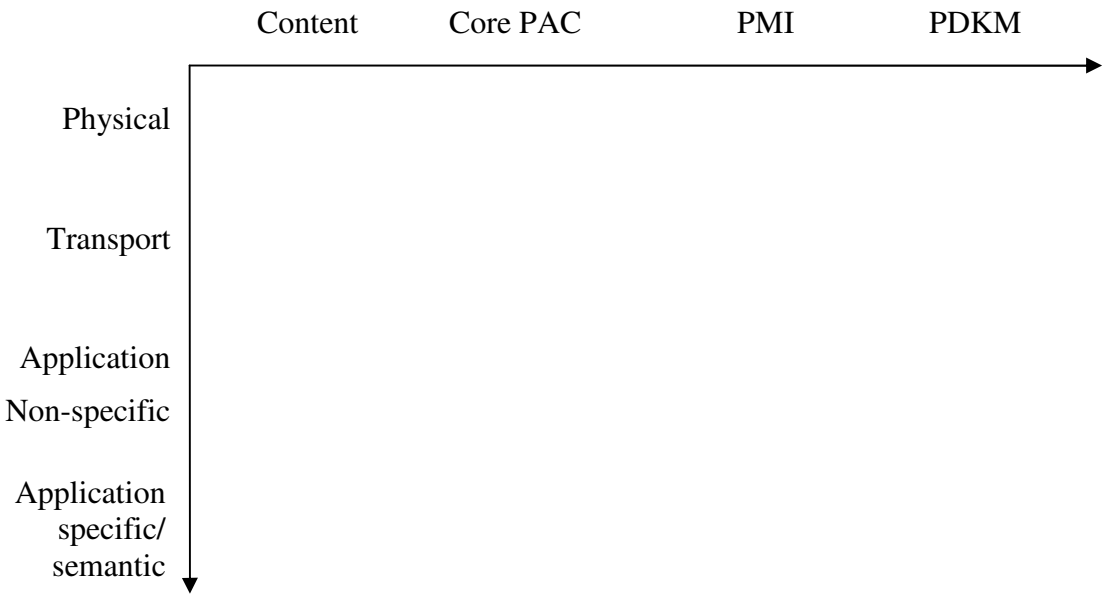

Figure 1: Matrix for identifying role of standard in PROMISE

Conversely, the matrix would also suggest areas where no standard exists and thus may provide a basis for recommending the development of future standards. For example, the use of UPnP within PROMISE consists of some "application non-specific" parts where the technology is used generically, but there are also some "application specific" aspects, and these aspects need to be clearly specified to ensure that different PROMISE implementations are interoperable.

The horizontal axis of the matrix corresponds to different components of the PROMISE architecture. The arrangement of components is deliberate. Adjacent components interface with each other. For example, the PMI layer interfaces with the Core PAC and PDKM layers. In general, standardisation will occur at the interface. This allows different implementations of a layer to be used interchangeably. For this reason, some standards will have matrix coordinates that lie between two PROMISE architecture layers.

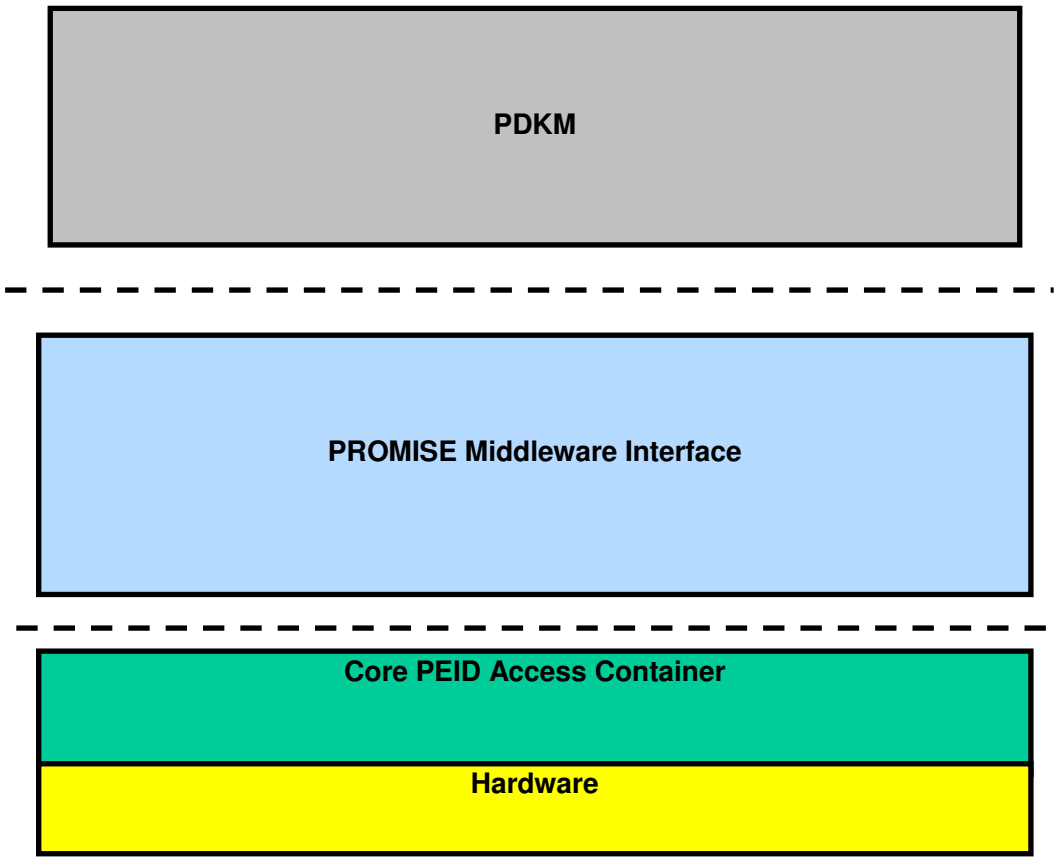

Figure 2: PROMISE Architecture

The vertical axis of the matrix corresponds to different protocol layers. Each layer is described in detail below:

Physical: The physical layer roughly corresponds to the physical layer of the OSI reference model. This layer corresponds to such things as the radio communication mechanism used to communicate with the PEID.

Transport: The transport layer roughly corresponds to the transport layer of the OSI reference model. Standards that ensure correct and robust communication over a physical medium fall into this layer.

Application Non-specific: The "application non-specific" layer refers to high-level protocols that are not specific to a particular application. For example, Web Services is a generic protocol that can be used in a wide variety of applications, and thus falls into this category.

Application Specific: The "application specific" or "semantic" layer refers to specific uses of high-level protocols that assign particular semantics or meaning to the communication. For example, EPC/IS is an example of an application specific protocol as it builds on the Web Services protocol and defines a specific set of method calls. Note that rather than any protocol being clearly non-specific or specific, there is a continuum from being highly general to being intended for a single purpose.

3 Assessment of existing standards

In this section, we examine the different existing standards and assess their suitability for the PROMISE architecture.

3.1 UPnP

Universal Plug and Play (UPnP) is a service-oriented middleware that provides connectivity between networked devices. It is based on the definition of standards for inter-device communication which are completely independent of the physical network layer, the operating system and the programming language. To this end, UPnP uses existing standards for communication and data exchange that are situated above the TCP/UDP layer like HTTP and SOAP.

Within PROMISE, UPnP is introduced as the device interoperability layer. This means that communication between the middleware and the devices/products is done using this standard. As UPnP poses no restrictions on the concrete implementation, neither on the hardware nor on the software side, the implementation of the respective interface can be done in a manor which fits the application case best. For example, in some cases like for Bombardier locomotives or Caterpillar vehicles it might be adequate to implement the UPnP interface on the on-board computers. In scenarios where passive RFID tags are used, implementation of the interface cannot be done on the tags themselves and might therefore be done on the RFID reader. Different concepts of how the interface can be applied to different products can be found in deliverable DR5.4. It is also important to mention that despite the chosen implementation, the UPnP interface stays the same and therefore transparency towards the upper layers of the architecture is achieved.

Let us now take a look at how UPnP can be used in PROMISE. Considering the features described above, there are mainly three possible ways to adopt UPnP in the scope of PROMISE:

- Integration of products: As already mentioned before, UPnP is the designated standard to integrate products into PROMISE. For items, like the Caterpillar crack-first sensor, one simply needs to implement the Core PAC interface (described in DR4.2) using UPnP. The PROMISE middleware is designed to look up and interact with Core PAC implementations. Therefore the sensor information can be accessed by the middleware and if there is also memory allocated on sensor, other information can be written and read by the middleware. Via the middleware the PDKM and DSS can conduct their work. Thus, integration of products is the main area of application of UPnP in PROMISE.
- Management of compound products: What we have discussed in the last paragraph was using UPnP as a way of connecting products to the upper layers of PROMISE. In addition to that, we can use UPnP as the means of communication inside a product. In cases where we have complex products consisting of multiple components these components can identify each other and exchange information via UPnP. If we stick to the Caterpillar crack-first example, we have an UPnP enabled crack first sensor. This sensor is part of a complete vehicle whose central processing unit is the board computer (BC). It might be desirable that the BC is aware of which sensors are attached to the vehicle and what the current value of these sensors is. Using the feature of dynamic lookup and self description of UPnP devices, the board computer can identify and communicate with the existing sensor. Also, sensors can dynamically be added and removed (for servicing) without the need of changing the BC program. As the UPnP mechanisms are independent of the implementation of the sensor, one can also change the sensor type e.g. from wired to wireless without having to adjust the BC. The advantages of UPnP in this area of course increase with the complexity of the product. The more components exist and the more flexibility is needed, the greater benefit is obtained using UPnP.
- Integration of systems: The third approach addresses the need in PROMISE to integrate existing systems. For example, there may already exists applications which include lifecycle monitoring devices, communication infrastructures, databases, etc. In these cases

it makes sense not to try to integrate every single product but the complete existing system into PROMISE. In the previous deliverables it has already been outlined that such an integration can be done by either implementing a so called specific device controller for this system (this allows reuse of parts of the middleware) or equip it with the interface of the middleware so that it can directly be used by the PDKM. Another way of integration is to equip the complete system with the UPnP based Core PAC interface. As the Core PAC is designed to provide access to arbitrary data via key value pairs, one cannot only provide information directly stored on the devices but also added value information that has been created by processing lifecycle information. Thus, a system which is reflected in a Core PAC can be looked up and interacted with just like a single product.

In general, the approaches of "Integration of products" and "Management of compound products" are the most promising areas of application for UPnP and will most certainly be used in PROMISE. The approach of "Integration of systems" is only an option that we should be aware of. Implementation of this option cannot be foreseen at the moment.

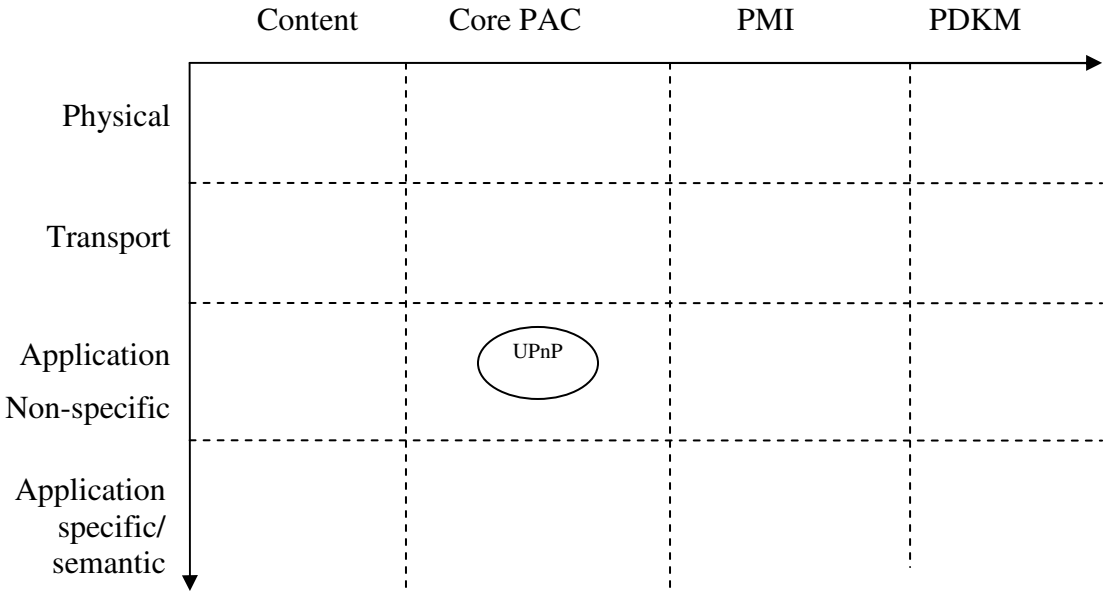

Figure 3: Role of UPnP standard in PROMISE

3.2 STEP – ISO 10303

ISO 10303 – STEP (STandard for the Exchange of Product model data) is an international standard for the computer-interpretable representation and exchange of product definition data. It was developed with the aim to provide a mechanism capable of describing product data as defined in ISO 10303-1 ("*representation of facts, concepts or instructions about one or more products in a formal manner suitable for communication, interpretation, or processing by human beings or by automatic means*"), independently from any particular system. Its natural implementation is that of a computer system and CAD, CAM, CAE software for product design. The way it was designed for describing product data makes it suitable for neutral file exchange among different software solutions, also in a distributed engineering or manufacturing environment. It can also operate as a basis for implementing and sharing product databases and archiving.

The STEP standard is organized as a series of parts, each one published separately. The structure is shown in Figure 4. The Information Models, in particular the Application Protocols describe the

data structures and constraints of a complete product model. Each application protocol combines one or more information models and places additional constraints on those models. For example, the application protocol for 2D drafting combines parts #42 and #46 and restricts the former so that it only describes two-dimensional data.

The Implementation Methods are protocols that are driven by the EXPRESS language. They are used to move real EXPRESS-defined application data between tools, and to make that data available to application developers.

STEP does not attempt to produce a single standard model that applies across disciplines. Rather, the STEP approach aims at producing standard product models for use within specific areas of application, called *Application Protocols* (APs), and strives to harmonize and coordinate these models across application areas to the greatest extent possible.

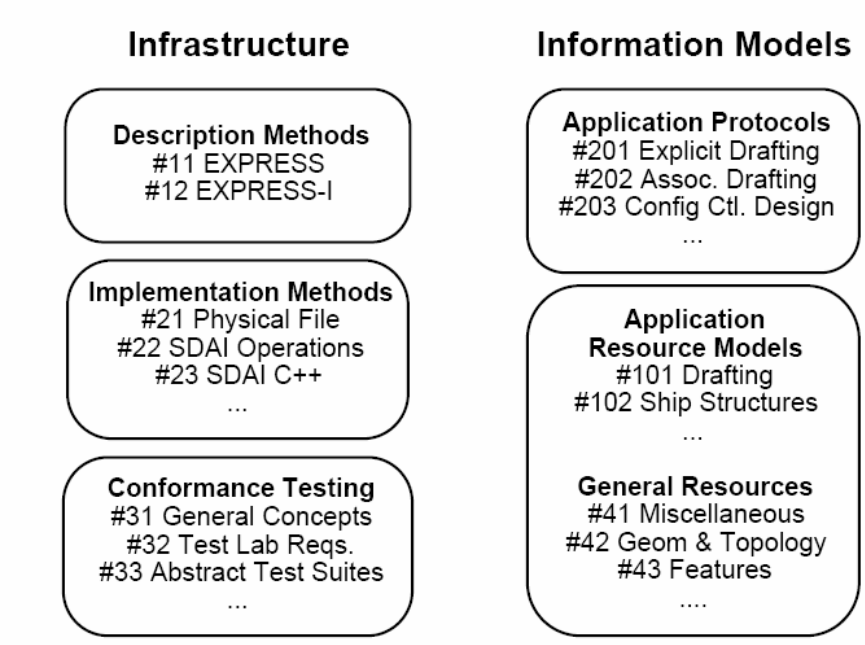

Figure 4: Parts defining STEP implementation

A fully developed AP presented in EXPRESS language is intended to be implemented to support information exchange. There are two primary implementation approaches directly supported by STEP. The first is file transfer in which the data corresponding to some specific project is exported from one computer application, structured according to the AP, and formatted as defined by the STEP *Physical File Format* (STEP part 21). This file can then be imported and interpreted by another computer application. The second implementation method is direct access to the data as stored in a database or similar system. In this case, STEP Part 22 defines the *Standard Data Access Interface (SDAI)* which provides a programming interface to the on-line data (bindings of the SDAI interface to languages such as C, C++, and IDL are being defined).

STEP is a comprehensive standard and implementing a STEP interface for a PLM system is a challenging task. The PDKM system is based on the commercial PLM system SAP ECC which covers a STEP interface. Using this interface data from the SAP system can be exported in STEP data format and data available in STEP format can be imported back into the SAP system. Nevertheless there is no known application scenario where an exchange of product model data between different systems is required. In the current scope of PDKM there is no functionality included which makes use of the existing STEP interface.

3.2.1 PLCS - ISO 10303-239:2005

Product lifecycle support (PLCS) is an application protocol (AP 239) of STEP (ISO 10303). PLCS was born as an initiative supported by both industry and national governments with the aim to accelerate development of new standards for product support information. PLCS, in fact, should be able to describe products needing support and the work required to sustain and maintain such products in operational conditions. It specifies the use of the integrated resources necessary for the scope and information requirements for product life cycle support.

The benefits from using PLCS to organise product support information rise as the product, or the support arrangements, are subject to more frequent change.

PLCS was built around STEP, thus it is easy to integrate PLCS data and applications within complex and heterogeneous software systems, reaching a high degree of interoperability. PLCS, indeed, shares the same common interface of other STEP-based software for product design and development, for maintenance management, for manufacturing scheduling etc.

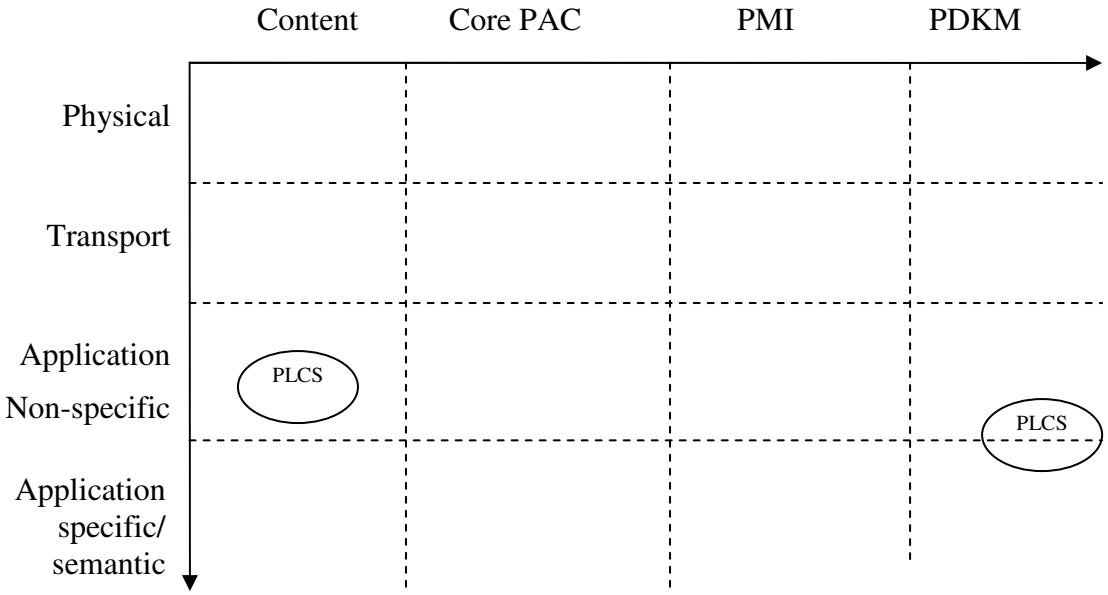

Figure 5: Role of STEP standards in PROMISE

It has already been mentioned that PDKM is based on SAP ECC which already offers a STEP interface. PLCS is a comparative new standard and the extensions to the existing STEP interface are not implemented yet. If PLCS will become an established standard, it is highly assumable that support for this standard will be provided with SAP ECC. Since PLCS is aiming at assistance on various topics that are addressed by PROMISE, PLCS support in SAP ECC – and PDKM in general – could be utilized to communicate with third party systems.

3.3 EPCglobal standards

EPCglobal is a subsidiary of GS1 (previously EAN (European Article Numbering) in Europe and UCC (Universal Code Council) in America). EPCglobal are responsible for the ongoing standards development process and commercialization of the EPC Network.

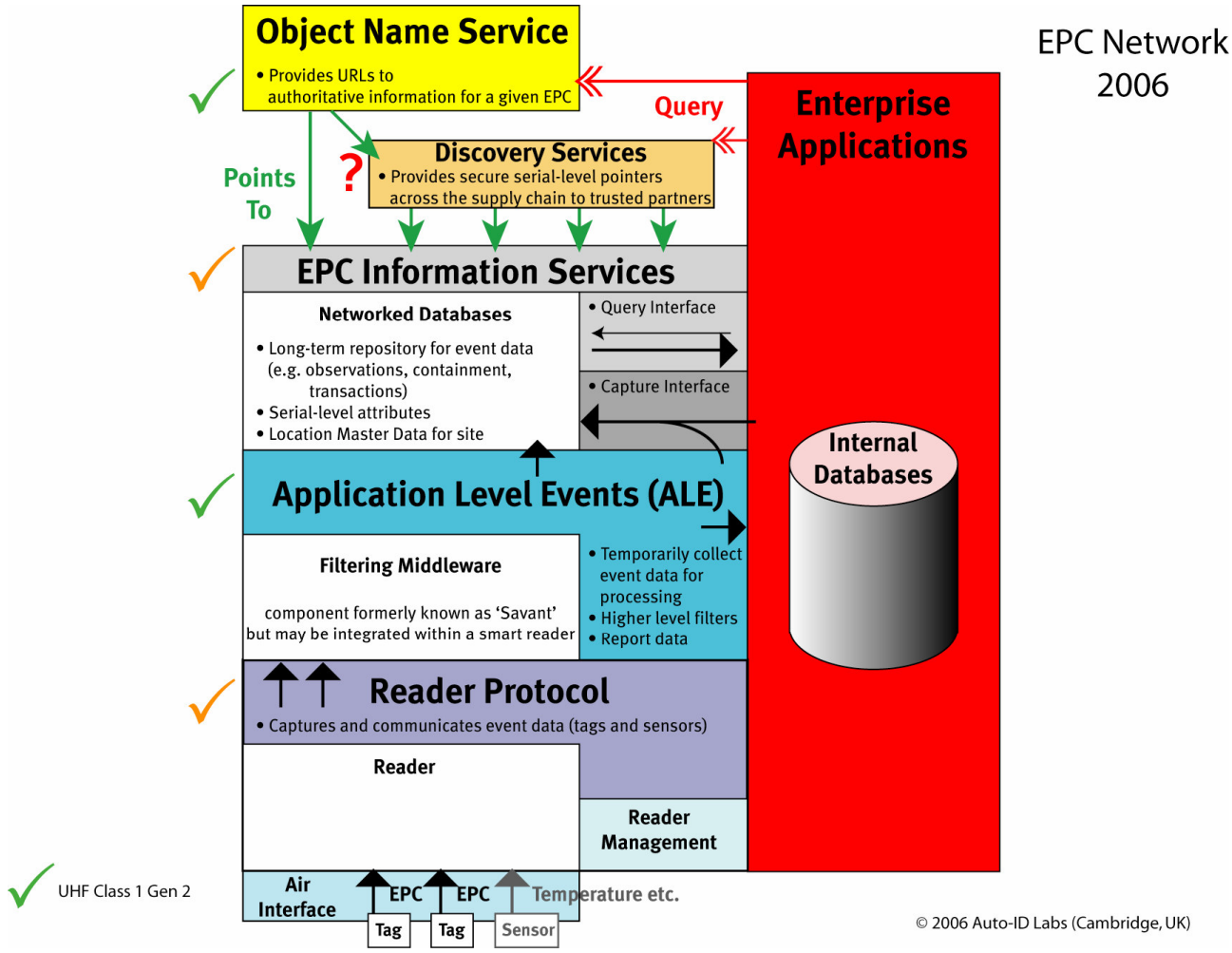

Figure 6 EPC Network architecture as of 2006 (EPCglobal Inc., 2006)[1]

Based on the EPC Network Architecture (see Figure 6), the EPCglobal body has now ratified six standards in addition to the standards that were developed and passed on by the erstwhile Auto-ID Center. Table 1 summarises all the standards within the EPCglobal Architecture Framework, and also presents their current status in terms of ratification. We will now provide a brief outline of the standards, and summarise the section by discussing their relevance to the PROMISE project. More details on each of these standards can be found on the EPCglobal website (http://www.epcglobalinc.org/standards/).

1. **EPC Tag Data Specification (TDS) Version 1.1** – This EPCglobal Board ratified specification identifies the specific encoding schemes for a serialized version of the EAN.UCC Global Trade Item Number (GTIN), the EAN.UCC Serial Shipping Container Code (SSCC), the EAN.UCC Global Location Number (GLN), the EAN.UCC Global Returnable Asset Identifier (GRAI), the EAN.UCC Global Individual Asset Identifier (GIAI), and a General Identifier (GID). EPC numbers can be assigned to physical objects, loads, locations, assets, and other entities that need to be tracked through the supply chain.

[1] Green ticks indicate ratified standards. Orange ticks indicate elements undergoing standardisation, for which ratified standards are expected in 2006. Red question marks indicate elements for which formal standards development work has not yet begun.

These numbers, encoded into RFID tags (PEIDs) and linked to relevant sensor data, could support the kind of information management that PROMISE requires. Also see Framling *et al.* (2006) for more discussions on the applicability of EPC in PROMISE.

2. **UHF Class 1 Generation 2 (Gen 2) Air Interface** – Defines the physical and logical requirements for a passive-backscatter, Interrogator-talks-first (ITF), radio-frequency identification (RFID) system operating in the 860 MHz – 960 MHz frequency range for use with passive tags. This specification is intended to replace the UHF Class 0 and Class 1 specifications inherited from the Auto-ID Center, which were mutually incompatible. It also provides several improvements over UHF Class 0 and 1 in terms of performance and security.

   In addition to being ratified by EPCglobal board, it has also been ratified as an ISO standard as Type C of ISO 18000-6. This standard therefore could be used as the communication standard in PROMISE where such RFID tags are used as PEIDs. However, since air interface is device-dependent, it is possibly not necessary to standardise at this level. One could adhere to a number of other standards depending on the application.

3. **ALE (Application Level Events) Specification** – Given the need to minimize bandwidth requirements, encapsulate complexity, respond to events in a timely fashion, and facilitate the distribution of processing, the Application Level Events (ALE) specification is for a software application programming interface (API), associated data specifications, and reporting mechanisms, through which clients may obtain filtered, aggregated tag read data from a multiplicity of tag read sources.

   Within PROMISE, ALE's scope lies in the middleware (PMI) and the interface between middleware and the PDKM.

4. **Reader Protocol Specification** – This standard specifies how data and commands are exchanged between hosts and RF readers. It will support reading, writing to, and deactivating RFID tags.

   Naturally, this specification could be used within the PROMISE project for interacting with the PEIDs, where RFID tags are used. However, as with the EPC Tag Data Specification, it would be unnecessary to develop standards at this level for PROMISE.

5. **Tag Data Translation (TDT) Specification** – This standard is a specification to express the current EPC Tag Data Standards encoding and decoding rules in an unambiguous machine-readable format, which will allow any component in the EPC Network technology stack to automatically convert between the binary and tag-encoding and pure-identity URN formats of the EPC as appropriate.

   Since, this specification can be seen as one that "interprets" the information that the tag provides, the scope is relatively wide within the PROMISE architecture, spanning from the content (PEID) layer through to the middleware interface.

6. **ONS (Object Name Service)** – Specification describes a framework for obtaining authoritative information services for a given EPC identifier. This standard specifies how the Domain Name System (DNS) is used to locate authoritative metadata and services associated with a particular product at the product class-level (as opposed to item-level data). The Object Name Service (ONS) is used to convert an EPC into a number of internet addresses where further information about a given object may be found. In order to avoid overloading and degrading the performance of DNS, the static ONS will generally not perform serial-level

lookup but rather object-class lookup – e.g. product-line (SKU) granularity for trade items. ONS therefore usually returns a number of manufacturer-oriented web addresses for various services, such as HTML web pages, web services, XML product data, etc.

Recognising that other parties on the supply chain may also hold relevant data about an object and that issues such as traceability require a robust solution, it is likely that ONS will be augmented with a dynamic counterpart called 'Discovery Services', which is able to provide the serial-level lookup for instances of a given product, pointing to the various other parties across the supply chain, which also hold information.

Table 1: EPCglobal Standards

Activity	Standard	Status
Object Exchange	UHF Class 0 Gen 1 RF Protocol	Auto-ID Center standard
	UHF Class 1 Gen 1 RF Protocol	Auto-ID Center standard
	HF Class 1 Gen 1 RF Protocol	Auto-ID Center standard
	UHF Class 1 Gen 2 RF Protocol	Ratified
	EPC Tag Data Specification	Ratified
Infrastructure	Reader Protocol	Ratified
	Reader Management	Under development
	Tag Data Translation	Ratified
	Application Level Events (ALE)	Ratified
	EPCIS Capture Interface	Under development
	EPCIS Data Specification	Under development
Data Exchange	EPCIS Query Interface	Under development
	ONS	Ratified
	EPCIS Discovery	TBD
	Subscriber Authentication	TBD

In addition, EPCglobal also have two standards that are under development:

1. **EPCIS (EPC Information Services) Specification** – This is a set of specifications for a software application programming interface (API), associated data specifications, and security mechanisms, through which various clients may capture, secure, and access EPC-related data and the business transactions with which that data is associated. The EPC Information Service allows trading partners to access and exchange well-defined subsets of their live real-time data, through a standard interface, with full web service security access controls and authentication, while interfacing the back-end to diverse databases and information systems from multiple vendors, without their partner needing to know the details or have direct access to the underlying systems.

2. **Reader Management Specification –** This specification is intended to define a set of standard functions that enable configuration, provisioning, monitoring, and alarm notification of individual RFID readers. It will leverage the standard communication protocol defined by the Reader Protocol Specification where applicable.

Both ALE and EPCIS support polling and publish/subscribe interfaces, as PROMISE requires in DR4.2. EPCIS v1.0 supports reading of key/value pairs for an object via its Query Interface - but does not define a set Method to write these values. ALE 1.0 lacks support for sensor data - so it only handles the ID observation events. Thus the extension of ALE to support sensor data could be an activity for the PROMISE standardisation work package.

The PROMISE Middleware Interface plays a similar role as that of the EPCIS - an abstracted way of accessing data, without needing to know how the underlying data is stored or retrieved. However, the data binding to the underlying database is not supported by the EPCIS. Querying for parent/child relationships is supported in the current EPCIS interface - so it could probably be re-used on a device-centric data device, by specifying the parent ID and querying for the "children". Moreover, the IOCI component within the PROMISE middleware (refer to DR9.4) plays a role similar to that of Discovery Services, except that in PROMISE, it is combined effectively with the EPCIS query interface in such a way that an application can get to a remote organization's middleware by sending commands to the IOCI.

DR9.2 describes a specific data model to support end of life. A concrete XML schema would probably be the next logical step after the UML diagrams in the document - and it may be possible to align this with an EPCIS vocabulary for the key/value pairs, since EPCIS is also agnostic regarding the industry sector.

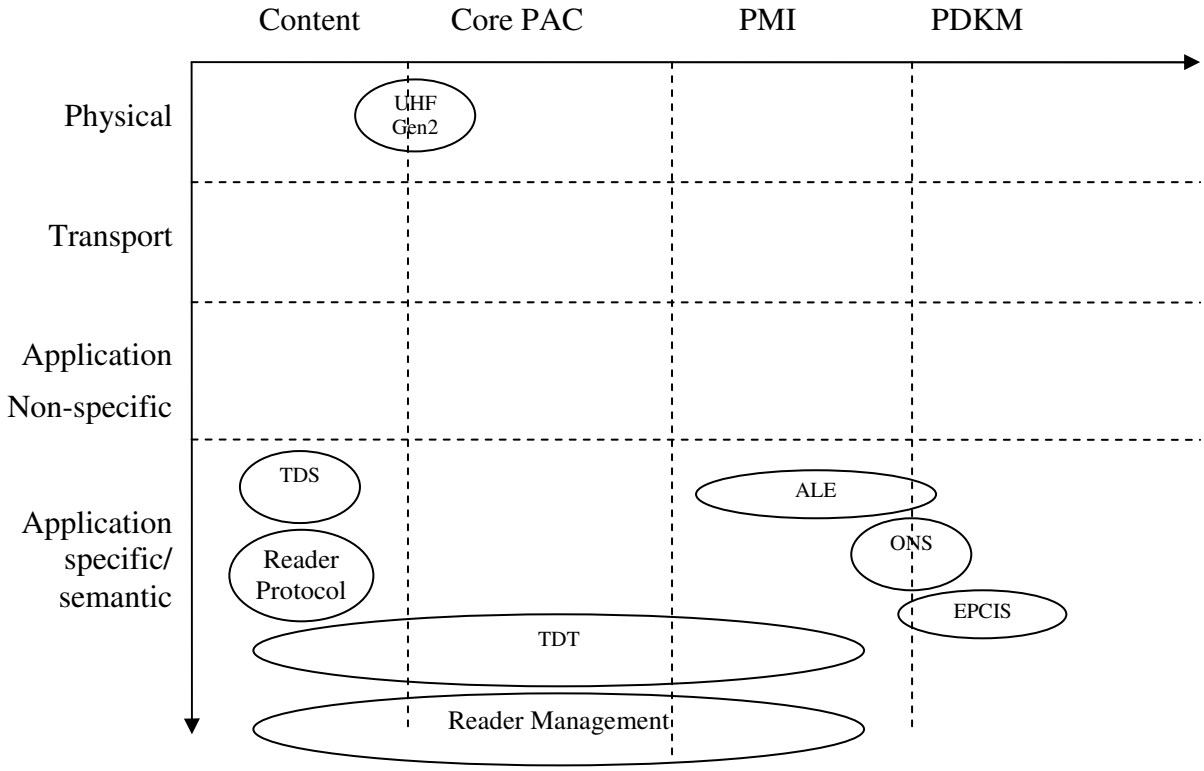

Figure 7: Role of EPCglobal standards in PROMISE

There are some other ways of interpreting the PROMISE approach in terms of the EPC Network - except that these involve lower-level interfaces, such as EPC Reader Protocol to read/write to tag memory and the next generation of EPC Tag Data Standards to handle representation of additional data in user memory - although this is quite specific to RFID tags .

The EPCglobal standards that are most relevant to PROMISE are ALE and EPCIS, since they already have the pub/sub interfaces, event messaging and key/value information about objects

(EPCIS). PROMISE could also identify and propose requirements for extensions to these standards.

3.4 World Wide Article Information (WWAI) protocol

The World Wide Article Information (WWAI) protocol is an open protocol specification for managing distributed object centric information in a networked environment. WWAI specifies a generic language for systems to communicate about objects in the WWAI network.

The WWAI (www.wwai.org) approach uses existing product-item identifiers and links to product information in backend systems through a peer-to-peer (P2P) based lookup mechanism. P2P systems are mainly known for file sharing of music and movies. However, P2P also has many desirable features for identifying nodes in the network as well as individual items. New nodes and items can be dynamically added at any time and are immediately integrated into the network. The network protocol usually takes care of assigning unique identifiers both for nodes and items automatically. Therefore there is no need for an external authority to manage codes as in the EPC approach. Other advantages of P2P solutions are that all nodes can maintain complete control of what data is distributed to whom (even though most file sharing applications do not check or restrict who gets access), good fault-tolerance (breakdown of one node affects the whole network very little) and possibilities to do load-balancing by using nodes that are "close" (in the network communication sense).

Existing company codes as issued by EAN/UCC or other standardisation bodies identify nodes of the network. When a node has joined the network, it can autonomously issue identifiers for individual items (e.g. product items). New nodes are dynamically discovered when appropriate. The WWAI protocol defines messages that enable nodes to exchange any kind of information and link any kinds of objects to each other by named relations. From a P2P point of view, the main criticism against WWAI is that it requires certificates issued by a certification authority in order to become an information provider in open networks. This certification is the basis for ensuring uniqueness of WWAI codes and represents a compromise between existing coding standards and ensuring the uniqueness of the codes, as well as ensuring data integrity. On the positive side, certificates automatically guarantee the authenticity of the information provider.

A WWAI node lookup (Figure 8) is usually only needed when a product identifier for a given company is seen for the first time. After that, network addresses of known nodes are cached so that new node lookups do not need to be performed unless the cached address fails or changes for some reason. In order to get access to other nodes in the network that have information about a specific product item (or some other item, e.g. a document, or a multimedia clip), it is sufficient to find one node with information about the product. This is because every time that a new node adds some information about a product item, this is automatically communicated to the other nodes that already have some information about that product item. Therefore there is no need for a separate "discovery" mechanism for accessing all information from all sources/organisations that have information about the product item.

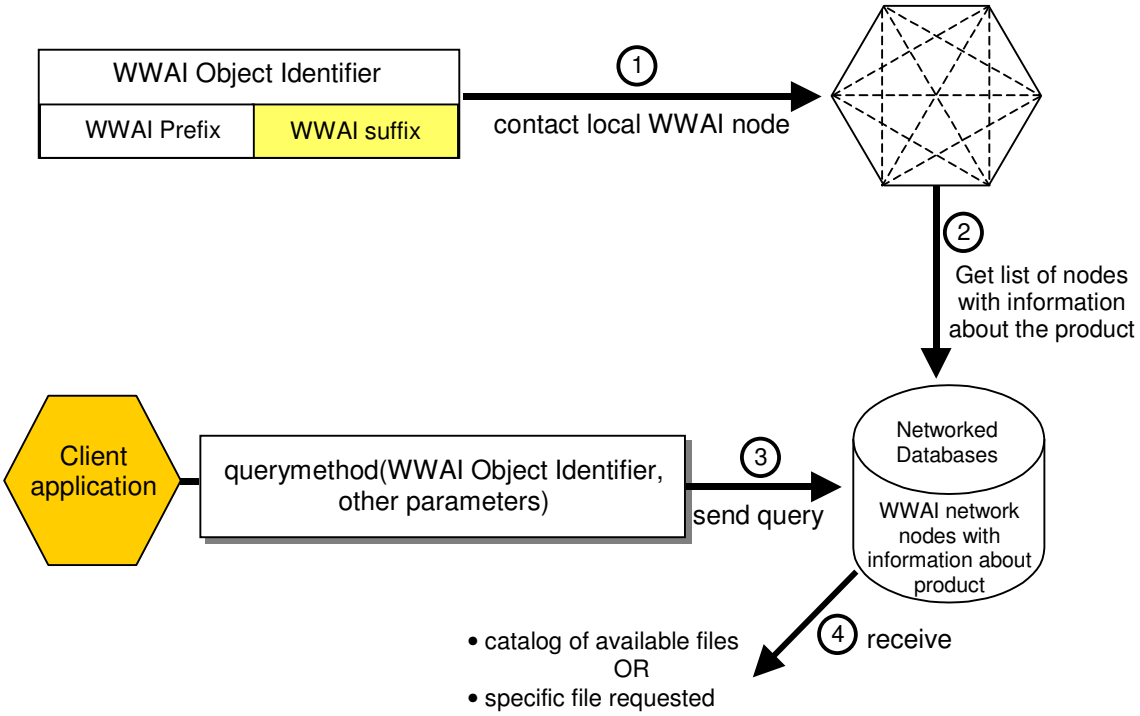

Figure 8. Product information lookup with WWAI approach.

In PROMISE, WWAI provides a mature and tested middleware platform for applications involving RFID/barcode/other identifiers, and especially so when multiple organisations are involved. WWAI has an integrated node identification, authentication and encryption mechanism which is essential in inter-organisational communication. The distributed nature of both information lookup and retrieval mechanisms is also a clear advantage in inter-organisational contexts where problems with one or more nodes in the network do not prevent accessing the rest of the information e.g. about a specific product item. In the PROMISE architecture, WWAI fits in as a messaging protocol that is accessible through PROMISE Web Service interfaces defined in work package R6. Those Web Service interfaces are intended to make it possible to use WWAI, JMS used by SAP or the DIALOG system of HUT in a transparent way, where it can be assumed that the best choice depends on the application.

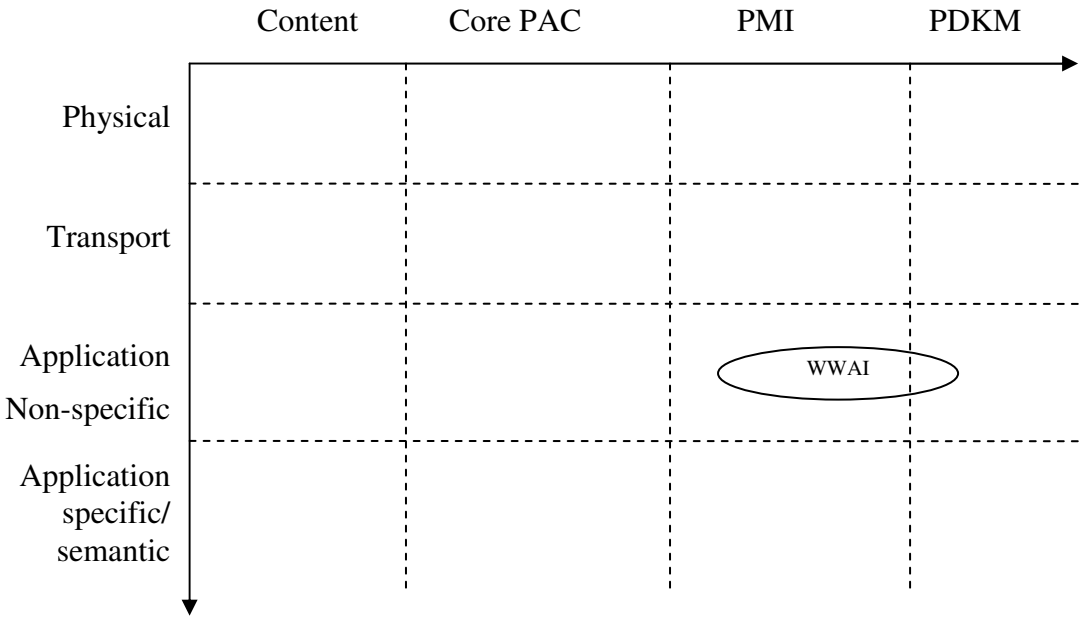

Figure 9: Role of WWAI in PROMISE

3.5 Web services

Communication between software located on different computing devices has existed almost as long as computers themselves, beginning with terminal programs allowing users to interact with mainframes. When rapid computer networks like Ethernet became widespread, it made it possible to distribute computing and data storage onto different computers. Databases were among the first implementations of "distributed computing", where client software components were used for interacting with the database. This communication initially used proprietary and database-specific protocols. Interoperability between different databases was greatly improved when the relational database concept became predominant and the SQL (Structured Query Language) could be used for querying different databases in similar ways. With the Java programming language and the JDBC (Java Database Connectivity) standard, it is now possible at least for Java-based programs to communicate with nearly any database.

Creating distributed applications where the application consists of software components located on different computing devices is more challenging. Nowadays such programs use the concepts *port* and *socket*, where one or several port numbers are allocated for communication and a socket software component handles the communication over those ports. In order to facilitate programming, the RPC (Remote Procedure Call) concept was developed, which made it possible for a program running on one computer to call functions/procedures of a program running on another computer instead of just exchanging data between sockets. Because most modern programs are based on the OOP (Object Oriented Programming) concept, RPC has since then been extended to support distributed OOP, where it is possible to obtain a reference to a remote object and call any of its public methods in the same way as if the remote object would be a local one. The most universal and well-known of these OOP protocols is Corba (Common Object Request Broker Architecture). RMI (Remote Method Invocation) is another widely used OOP protocol but it is Java-language specific. Both Corba and RMI use a registry mechanism which is used for retrieving a reference to the first remote object, which can then be used as if it would be a local object.

When the XML language became wide-spread for describing structured data as text, there were also initiatives towards using XML for defining remote interfaces. The most well-known protocols that use XML are XML-RPC and SOAP (Simple Object Access Protocol). Because these protocols use the standard GET and POST operations defined in HTTP, they are usually easier to use in Internet applications than Corba and RMI. Using HTTP makes it possible to use standard Internet ports (notably port 80), which reduces problems with firewalls. All exchanged data is encoded into XML, which makes XML-RPC and SOAP practically independent of programming languages. For the Java programming language, for instance, several different tools exist that are able to automatically convert Java objects into corresponding XML code.

Table 2 gives a summary of the advantages and disadvantages of binary versus XML-based protocols. For PROMISE, the three first criteria are the most important ones because many PLM scenarios require inter-organisational communication, where it is not possible to control firewalls, programming languages and software versions in a centralized manner. For these criteria, XML-based protocols offer clear advantages. This is the reason why XML-based protocols are selected for implementing the PROMISE middleware. Messaging protocols based on XML and other representations that use structured text are analyzed more in detail in the next section.

Table 2. Comparison between binary and XML-based protocols for distributed applications.

Selection criteria	Binary protocols (Corba, RMI, etc.)	XML-based protocols
Firewall configuration	Usually difficult in inter-organisational applications due to security policies (rather than for technical reasons).	Uses HTTP port 80 so this is usually not a problem. Still, experience has shown that e.g. SOAP does not always go through firewalls.
Programming-language independence	Good for Corba, not for RMI.	Good, similar as Corba.
Version management	Usually all components have to use the same version of the remote interface. For RMI, successful communication also puts constraints on the RMI and Java versions used.	Good. If methods or method parameters are added or removed, this means that only parts of the data are lost but communication still remains possible between components using different versions.
Size of exchanged data	Good. Binary representations are more compact than textual ones.	Large size because e.g. numerical data has to be represented as text. Especially SOAP also adds a lot of overhead due to headers and similar information.
Speed of data exchange	Usually good.	Slower than binary protocols due to larger messages and especially due to conversion of native data into XML structures.

3.5.1 Messaging protocols based on text structures or XML

HTML (HyperText Markup Language) is the standard for encoding contents of web pages that, together with the HTTP protocol, made the World Wide Web possible. HTML is based on a more extensive standard for representing formatted and structured documents called SGML (Standard Generalized Markup Language). HTML is mainly a standard for defining how documents should be formatted for being read by humans but it is less suitable for representing data in form that can be interpreted by machines. This is why XML was developed – representing data in a structured and machine-readable format. As mentioned in the previous section, this also makes XML suitable e.g. for exchanging data in a programming-language independent way.

How the XML data is sent from one computer to another may vary. Even though all XML-based protocols use HTTP as the communication protocol, the data can be sent at least using the following standards:

1. **HTML forms**: this is the oldest technique, used in common-place web forms whose contents are sent as parameter values to web applications for processing. It is also possible to send XML-formatted data as parameter values. Even though HTML forms are usually sent by browser software (e.g., Mozilla, Netscape, etc.), they can easily be generated by any software. The HTTP response can be used as a method return value. HTML forms are probably the most light-weight and straightforward way of creating text- or XML-based distributed applications.
2. **XML-RPC**: probably the first standard that uses XML serialisation (i.e. converting native data into XML) and HTTP for creating distributed applications. XML-RPC has a reputation of being light-weight and easy to use.
3. **SOAP** (Simple Object Access Protocol): standard proposed to W3C by Microsoft in 2001 that appears to have its origins in XML-RPC. Has the same functionality as XML-RPC and additional features but is also considered to be more "verbose" than XML-RPC, which increases the amount of network traffic.

In practice, SOAP has become the de-facto standard. Most so called Web Service (WS) standards assume the use of SOAP even though it is not a requirement as such. Many different WS standards are currently under development e.g. for ensuring message delivery, security, binary XML types etc. In the next section, we will shortly describe one of them – the Web Service Definition Language (WSDL).

3.5.2 WSDL - Web Service Definition Language

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME.

To express this in another way, WSDL is a way of describing WS interfaces so that they can be searched for and used by other software components who may not know about the WS in advance. In practice, what is declared in a WSDL document are the "methods" that can be called e.g. using SOAP. The parameters and return values of the methods are also described, as well as their data types. All "basic" data types (e.g., integer, double, string) as well as arrays of these can be used. However, all structured types of different programming languages are not supported

(e.g., Java HashTable). To summarize, creating a typical WS requires writing a WSDL that describes it and implement support for SOAP messaging. Still, as can be seen from the citation above, other messaging protocols than SOAP could also be used.

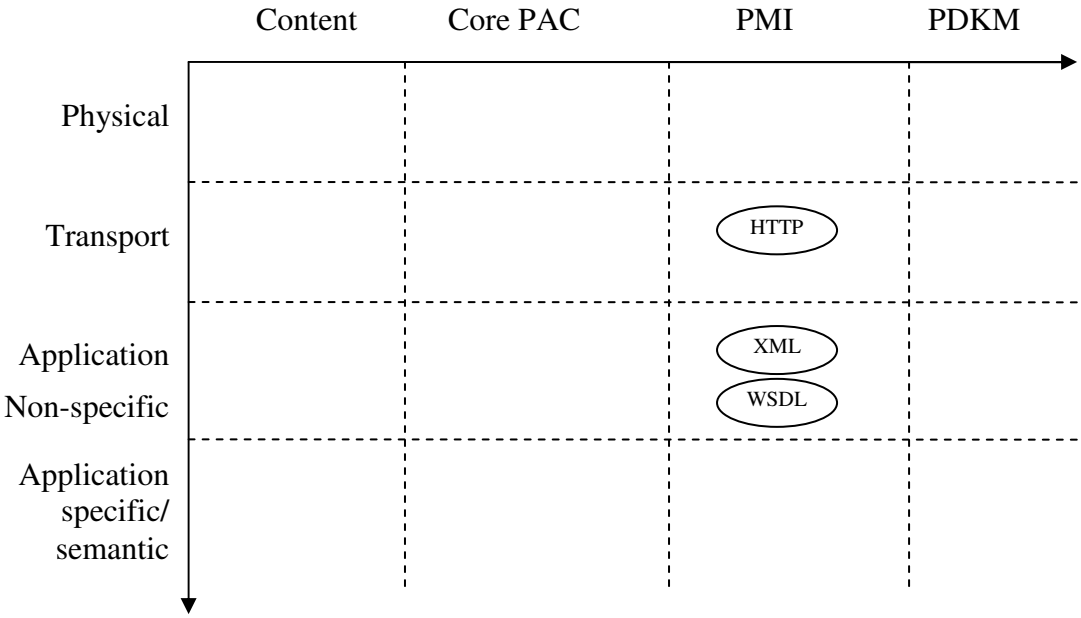

Figure 10: Role of messaging protocol standards in PROMISE

4 Conclusions

In this report, we examined several standards with a view to assess their suitability for PROMISE. In particular, we evaluated the family of standards developed by EPCglobal, which supports the most of the layers in the PROMISE architecture, especially from an application-specific viewpoint (refer to Figure 11 for a summary of all the standards assessed in this report). Within the EPCglobal family of standards, we also identified opportunities and need for extension to suit the application scenarios within PROMISE.

In addition, we also found that several application non-specific standards exist that can support all the layers of the PROMISE architecture. In particular, from Figure 11, there is a gap in the application-specific domain within PMI, which could be filled by developing an application-specific specification that illustrates how UPnP can be used specifically within the PROMISE application scenarios.

This report thus provides a good basis for recommendation of the best standard (or extension to standards) for the different standardisation domains.

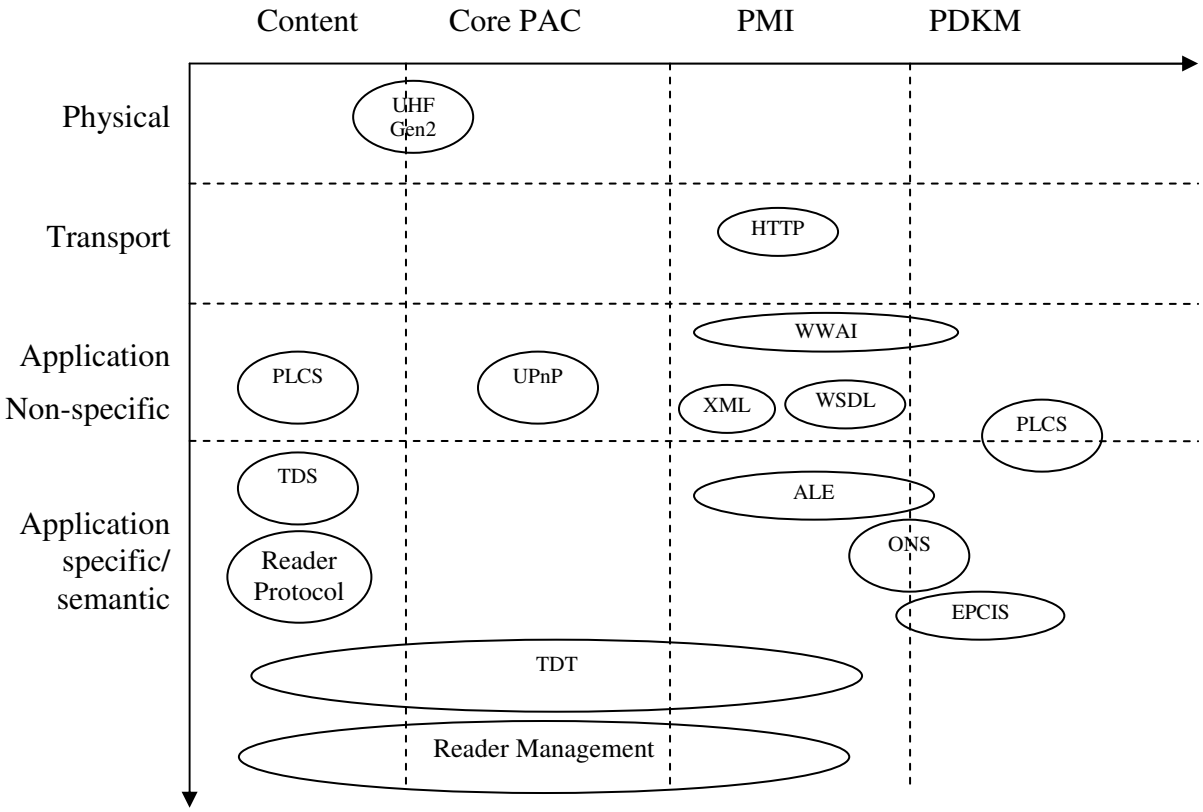

Figure 11: Role of standards in PROMISE

5 References

1. www.iso.org
2. http://www.tc184-sc4.org
3. http://www.step-nc.org
4. International standard ISO 10303-239:2005(E)
5. http://www.plcs.org
6. http://www.plcsinc.org/whitepapers/03pdt-eu.pdf
7. http://stepmod.sourceforge.net
8. http://www.eds.com
9. http://www.tc184-sc4.org
10. http://www.isa.org
11. http://www.wbf.org
12. http://www.epcglobalinc.org
13. Framling, K., Harrison, M., Brusey, J., "Globally Unique Identifiers – Requirements and Solutions to Product Lifecycle Management," INCOM 2006.