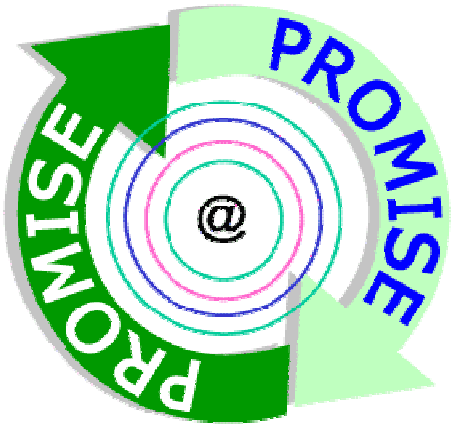

DI1.2: PROMISE Standardization Domains

Written by:
Guido Stromberg, Infineon Technologies AG
Daniel Barisic, Infineon Technologies AG
Gregor Hackenbroich, SAP Research, Germany
Mario Neugebauer, SAP Research, Germany
Jürgen Anke, SAP Research, Germany
James Brusey, Camebridge University
David Potter, Indyon GmbH
Andreas Edler, InMediasP GmbH
Altug Metin, InMediasP GmbH

DELIVERABLE NO	DI1.2: PROMISE Standardization Domains
DATE	31. October 2005
WORK PACKAGE NO	WP I1: Standardization
VERSION NO.	1.0
ELECTRONIC FILE CODE	DI1.2 PROMISE Standardization Domains.doc
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT:	This deliverable deals with the definition of standardization domains. These domains are the Content, Device Interoperation, Middleware and Backend domain. For each of these domains promising standard candidates are nominated taking the work of DI1.1 into account.

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE** (dd.mm.yyyy)
SUBMITTED (author(s))	Daniel Barisic	31.10.2005
VU (WP Leader)	Guido Stromberg	31.10.2005
APPROVED (QIM)	Dimitris Kiritsis	15.11.2005

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
31.10.2005	1.0	Daniel Barisic	Version submitted

Author(s)' contact information

Name	Organisation	E-mail	Tel	Fax
Guido Stromberg	Infineon	guido.stromberg@infineon.com	+49 89 234 40430	+49 89 234 52227
Daniel Barisic	Infineon	daniel.barisic@infineon.com	+49 89 234 20691	+49 89 234 52227
Gregor Hackenbroich	SAP Research	gregor.hackenbroich@sap.com	+49 351 4457 2303	+49 6227 78-43474
Mario Neugebauer	SAP Research	mario.neugebauer@sap.com	+49 351 4457 2312	+49 6227 78-44321
Jürgen Anke	SAP Research	juergen.anke@sap.com	+49 351 4457 2304	+49 6227 78-44661
James Brusey	Cambridge	jpb54@cam.ac.uk	+44 1223 765605	+44 1223 338076
David Potter	Indyon	david.potter@indyon.de	+44 23 9234 5152	+44 23 9259 2327
Andreas Edler	InMediasP	edler@inmediasp.de	+49 3302 559420	+44 3302 559124
Altug Metin	InMediasP	metin@inmediasp.de	+49 3302 559409	+44 3302 559124

Table of Contents

1 Purpose of this Deliverable

This deliverable aims at exploring the potential areas where standardization may foster the exploitation and commercialization of the PROMISE software and hardware components, and where it may help to agree on common interfaces between the PROMISE components. We therefore see existing standards as a helping guideline for implementing the PROMISE architecture and will strongly favour using existing standards wherever possible.

The areas of standardization are therefore defined by the interfaces that are given by the PROMISE system architecture, as it has been defined by the corresponding deliverables. Thus, there is a natural choice of relevant standardization domains, so that we in this document already make an attempt to foresee on each layer of the PROMISE architecture the extent to which standardization will be useful or even possible, and propose existing standards if applicable. In this delivery we in particular strive to identify the standardization domains that allow utilizing a tight standard without restricting its area of applications. However, as the fields of applications in PROMISE are wide, enforcing a particular standard for a particular standardization area may turn out as inappropriate in the end.

We believe that the standards mentioned in this document must be considered as suggestions. On each architectural layer, i.e. in each standardization domain, more than one standard may be proposed. However, for each application domain, one of these standards should be proposed per standardization domain. The proposed standards will be the ones that are used for the demonstrators until month 18 in the PROMISE project. In the next deliverable due at month 18, these choices will be reviewed and summarized as suggestions for the following months of development.

In the overall PROMISE project, the dealing with standards should follow the following guideline. The first approach is to use an existing standard if it is applicable for the respective application area. If no such standard is available, the second approach is to extend an existing standard. The proposed solution that is used in PROMISE should still be standard-compliant and could thus be used as a reference implementation for future standard-shaping activities. The third approach is to find an agreement in the overall PROMISE consortium on a best-practice definition. Clearly, the first and the second approach are favoured over the third. We expect that the third approach to define the interface between the PROMISE components is concurrently the one that inherits the highest risk to make component interoperation fail. However, it should be noted that the interface definition is in the responsibility of the respective research cluster work packages.

This document is organized as follows. In the following section, we will review the PROMISE system architecture and identify the individual interface layers on which standards could be applied. These are the so-called standardization domains, and will be detailed on in Sec. 3. In particular, we will refer to some potential standard candidates for each individual standardization domain. As explained above, these candidates serve as examples rather than as the final choices.

2 Introduction

In the following, we will identify and assess the utilization of standards in the PROMISE project. As the project acts in a wide area of application as well as it has a large vertical scope from hardware layers to backend enterprise systems, it is necessary to define suitable standardization domains.

We have opted to organize these domains vertically. That is, the interfaces between the individual components of the PROMISE architecture are potential candidates for either adopting an existing standard or for standardization. Within these standardization domains, the appropriate interfaces may depend on the specific application, so that on each layer more than one potential candidate for standardization could be identified. However, for each application domain, one particular interface should be suggested on each architectural layer.

Fig. 1 shows the accepted view on the overall PROMISE architecture. In the sequel, we will briefly characterize the standardization domains.

The *content domain* deals with the data that is collected by or stored on the PEID. Its format and semantics could vary widely for the different application cases, therefore a PROMISE objective should be to standardise on the protocols and data structure used for accessing data collected by or stored on the PEID while giving maximum flexibility to the actual data elements required by any single application..
Standards and formats in the content domain must take account of the following elements:

1. Interfacing with subordinate devices e.g. RFID tags, analogue and digital sensor signals, bar-code data, diary data (let us call this *raw* data).
2. Data received from the upper layers of the PROMISE architecture, which should normally be in the format preferred by PROMISE (XML format and semantics).
3. The need for transformation of *raw* data into the preferred PROMISE format, and vice versa.

The *device interoperation domain* deals with the communication of the PEIDs content to the PROMISE middleware layer. The functional interface for this purpose is the Core Product Embedded Information Device Access Container (Core PAC) introduced in DR4.2. The Core PAC is not necessarily a physical thing, but a view on a container of PEIDs with universal, predefined methods how to find PEIDs, how to read information and how to write information to them. An application independent definition of the Core PAC is one candidate for standardization in this domain.

In order to allow access to the Core PAC by the PROMISE middleware a suitable device interoperability layer has to be agreed upon. The desired device interoperability layer provides mechanisms so that devices can be automatically discovered by the middleware, and it supports the description of the device's functionalities. Further, the device interoperability layer provides communication mechanisms which allow it to convey and receive the content to and from the middleware layer. The definition of the device interoperability layer is the second goal in the device interoperation domain.

The *PROMISE middleware domain* deals with the connection of the Core PAC and the backend applications. The core of the PROMISE middleware is specified in DR6.2. In the description, the Device Controller is divided in two parts: the Device Handling Layer and the Request Handling Layer. The interoperability of the devices and the middleware should be ensured with the interfaces that are defined in the device interoperation domain. The standardization efforts in this work package should also deal with the interfaces between the Device Controller, the Enterprise Communication Infrastructure and the Backend Applications. As described in DR6.2 the PROMISE middleware should be designed following the service oriented paradigm. This implies that certain standardization or use within the middleware is required as well and has to be considered.

The *backend domain* deals with the management of product data over the entire product life cycle. Therefore on the basis of a Product Data Management System the PROMISE Product Data and Knowledge Management System (PDKM) will be designed and implemented (see also DR9.1).

As a central component of the PROMISE approach the PDKM system integrates and manages product-related data from all lifecycle phases to support comprehensive data analysis and to enhance operational businesses with obtained insights. The PDKM system provides the structures for representation and distribution of field data from BOL (production phases), MOL and EOL and serves for management of engineering knowledge, derived from field data to support BOL processes (design phases). A major challenge in realizing the PROMISE vision is the consistent integration of heterogeneous product-related data from various operational sources of the different lifecycle phases to support comprehensive data analysis. This includes – besides the integration of PEID data which will carried out by the PROMISE middleware and the PDKM application interface – the integration of data from different applications like PDM systems, field data bases or ERP systems. These tasks require certain standards that have to be considered.

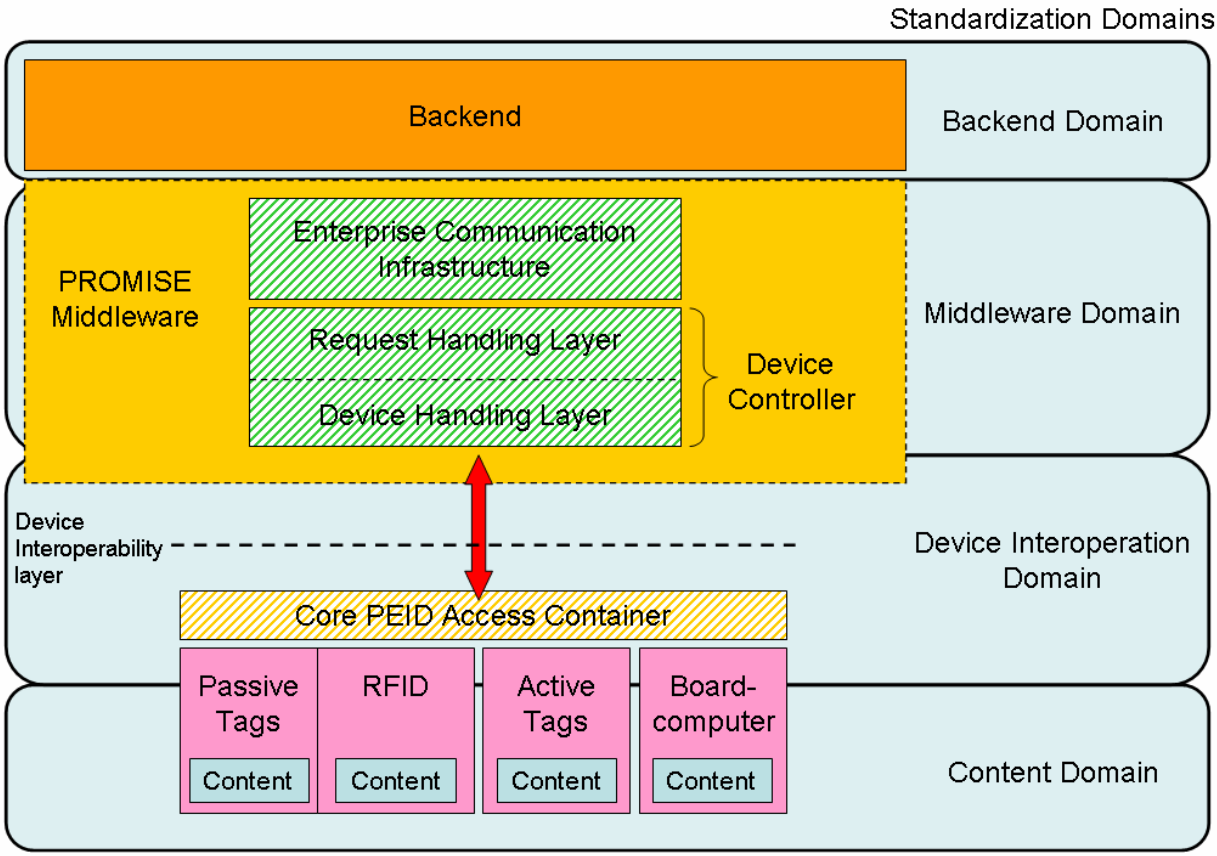

Fig. 1: PROMISE component architecture

3 Standardization Domains

Let us now take a more detailed look at the standardization domains and point out promising candidates for standardization.

3.1 Content Domain

The Content Domain must support a variety of formats standards and protocols owing to the different data sources that it must include, and different levels of PEID functionality.

A PEID could be any one of the following:
1. simply an RFID device
2. a device using Sindrion technology
3. functionality embedded in the firmware of a microprocessor controller (e.g. domestic appliance)
4. functionality embedded in the software/firmware of a highly sophisticated on-board management system (e.g. vehicle)

In the first case, the data formats and protocols for data exchange are mostly covered by the various standards included in the EPCglobal “EPC Network” architecture, and PROMISE should focus on those standards. The EPCglobal standards were described in PROMISE document DI1.1. The key element of this set of standards, relevant to the content domain, is the EPC numbering scheme. This scheme provides each tagged object with a globally unique identifier. This scheme has the advantage that it is compact (requiring only 96 bits) and compatible with other usage within the EPC Network.

However we must also take care not to exclude any other established formats outside of the EPCglobal standards. Other possible formats and protocols include the DIALOG system proposed by Helsinki University of Technology and the WWAI mechanism proposed by Stockway. These standards are examined in more depth in section 3.1.1.

In the other three cases (i.e. other than a simple RFID device), the ideal situation would be to impose a single PROMISE standard, XML-based schema for the storage and access of PEID data. An approach for this would be to identify any existing, robust data models which already contain the majority of data elements required for PLM (maybe, for example, SAP data models), then to perform a gap analysis on them to identify additional data elements that must be added. The final stage will be to incorporate those additions into the target standard(s).

However we must recognize that not all users may be willing to re-engineer existing “PEID” functionality, in which case a methodology for mapping between the proprietary formats and the desired PROMISE standard must be defined. Even where the PEID is being designed from scratch, it may not be feasible to provide enough memory or processing capability to allow data to be stored in an XML form. Therefore, we should recognize that the content domain standards should not dictate format.

Since the Content Domain must cater for bit-oriented encoding schemes on one hand, and XML-based schemas on the other, transformation of data is unavoidable at some points and with some kinds of data. Although not strictly a standards issue, the question of responsibility for transformation of data from one format to another for storage on the PEID needs to addressed.

In the case of a sophisticated PEID (possibly embedded in an on-board computer or microprocessor) it may be possible, even desirable, to make the transformation within the PEID. However some PEID implementations will require "external" software to make the transformations. Perhaps this could always be the Middleware, and in some instances the Middleware would be physically external to the PEID and in others alongside it in a sophisticated on-board system.

Certain kinds of data, e.g. diary data, trend data, alarm data, imply other requirements such as standardised date and time stamping, sampling rates etc. We should consider any data format issues to be within the scope of this document. However, sampling rates and event triggers should be considered only to the extent that they may require additional data elements to define those rates and triggers.

In the following section, we examine the naming standards that have the potential to be used by PROMISE for assigning each PEID a unique name (or number).

3.1.1 Naming Standards

There are a number of naming standards that appear to be suitable for PROMISE applications. In this section, we compare these standards and examine their fit to PROMISE.

3.1.1.1 Electronic Product Code

The electronic product code (or EPC) has been standardised by EPCglobal, a subsidiary of GS1 (previously known as EAN.UCC). An EPC is typically stored on a passive, low capacity RFID tag. This technology was originally chosen as a way to minimise cost, although EPCglobal have left open the possibility of using an EPC with a more sophisticated tag that has memory and / or sensors. A diagram showing the use of an EPC is given in figure 2.

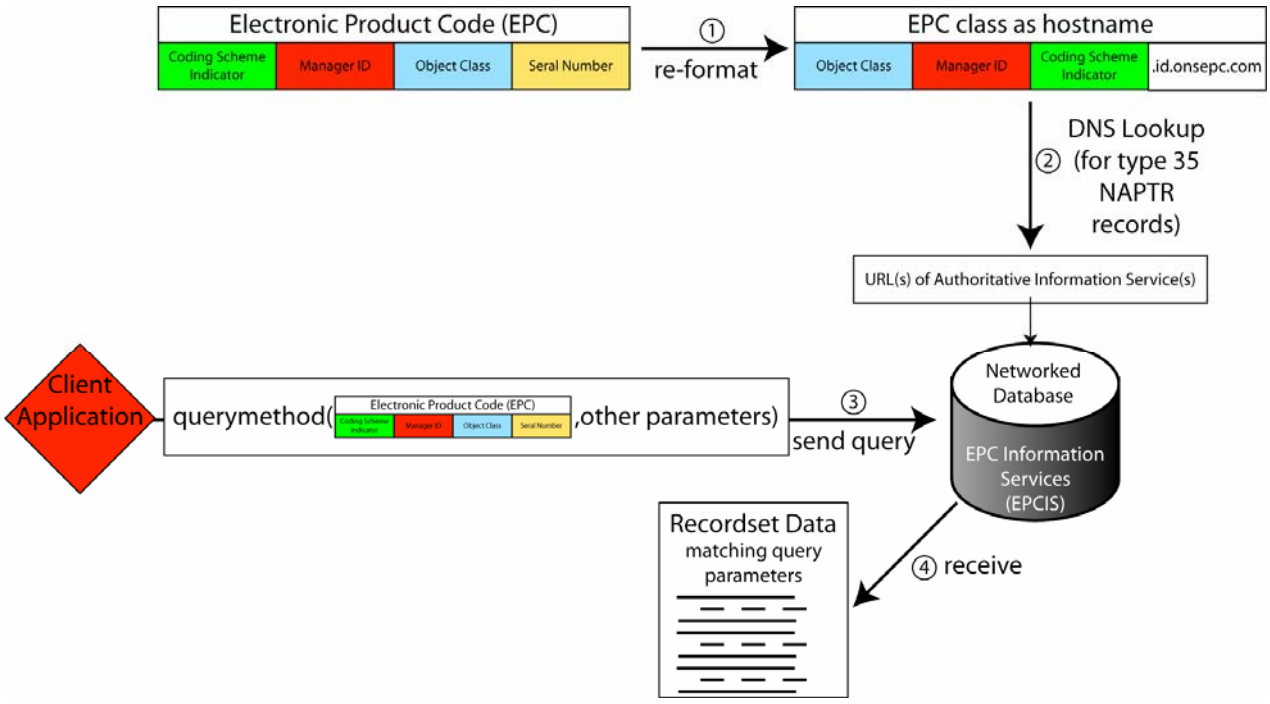

Fig. 2 Overview of EPC Approach

As can be seen from this diagram, the EPC itself consists of 4 components: a coding scheme indicator, a manager identifier, the object class, and a serial number. An EPC can be roughly considered to be a Universal Product Code (such as that used for barcodes on consumer products) plus a serial number. The current format uses just 96 bits and thus is quite economical in its use of memory on a RFID tag.

In a typical EPC Network application, the first stage (1) is to re-format the EPC into an Internet hostname (or URI). Existing DNS infrastructure (2) using a special lookup type translates this into a URL. This two level indirection is a feature of this approach and ensures that it is possible to

change the location of web services associated with an EPC without having to update the RFID tag. Note also that although the serial number is dropped in the process of reformatting, it can be reinserted during step 2. A typical EPC Network application (3) might be querying for specific information (such as identifying the manufacturing batch number), or general information, such as the location of online product manuals. Results are returned to the client (4) in the form of a recordset as a response to a query.

One factor that may influence the selection of approach is that the EPC numbering scheme usually requires that users pay EPCglobal for registration of each different product class (the exception being US DoD suppliers). This is currently more expensive than simply registering a DNS entry however it has the advantage that a single body manages the numbering scheme and that the possibility of duplicates is thus reduced.

3.1.1.2 DIALOG System

A diagram summarising the DIALOG system is shown in figure 3.

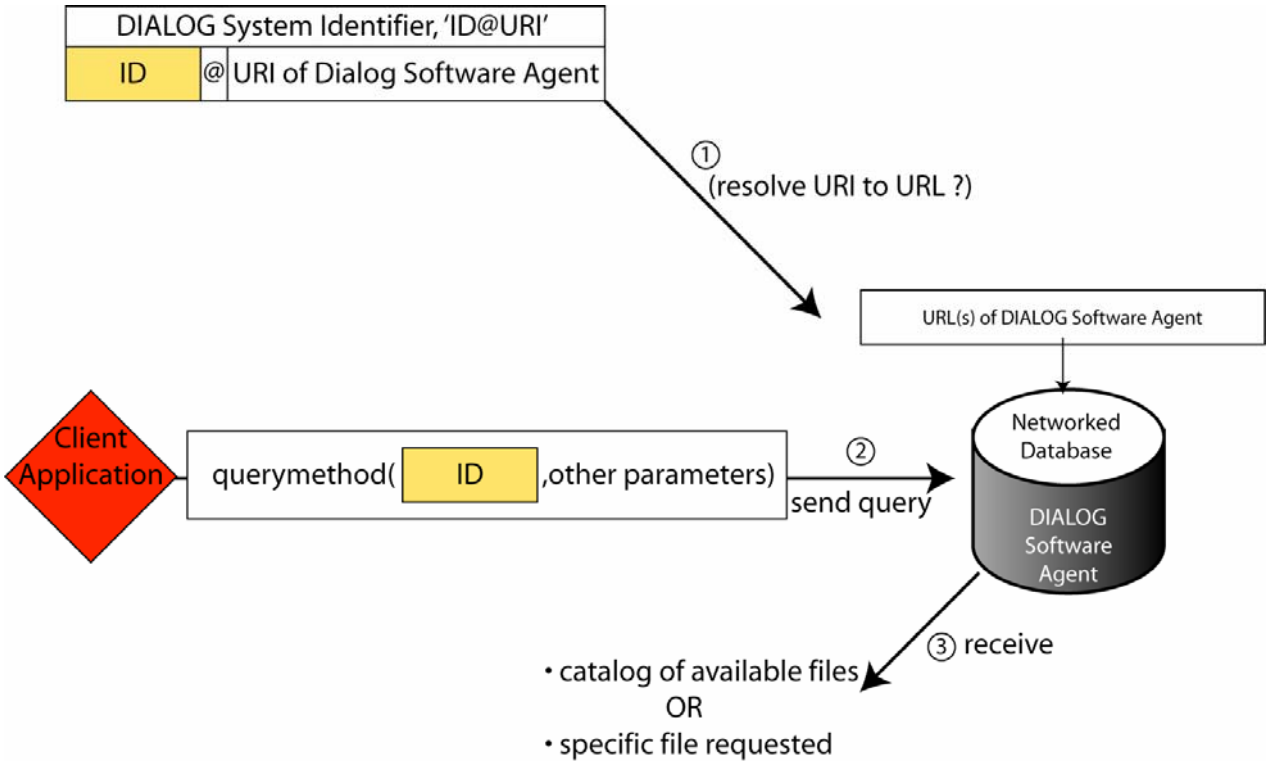

Fig 3. Overview of the Dialog System approach

The DIALOG system identifier is based on the IETF Universal Resource Identifier (URI) standard. The DIALOG approach is to write a serial identity number followed by the "@" character and then a URI. The first stage (1) is to resolve the URI into a URL (i.e. a normal web address). Although not strictly defined, in most cases this would be a one-to-one mapping. An advantage of this approach is that it reduces the reliance on DNS. A disadvantage is that it may not be suitable if the web address of the networked database changes during the life of the product. However this disadvantage might be overcome by using some more flexible mapping between URI and URL.

3.1.1.3 WWAI

Stockway's WWAI approach is summarised by in figure 4.

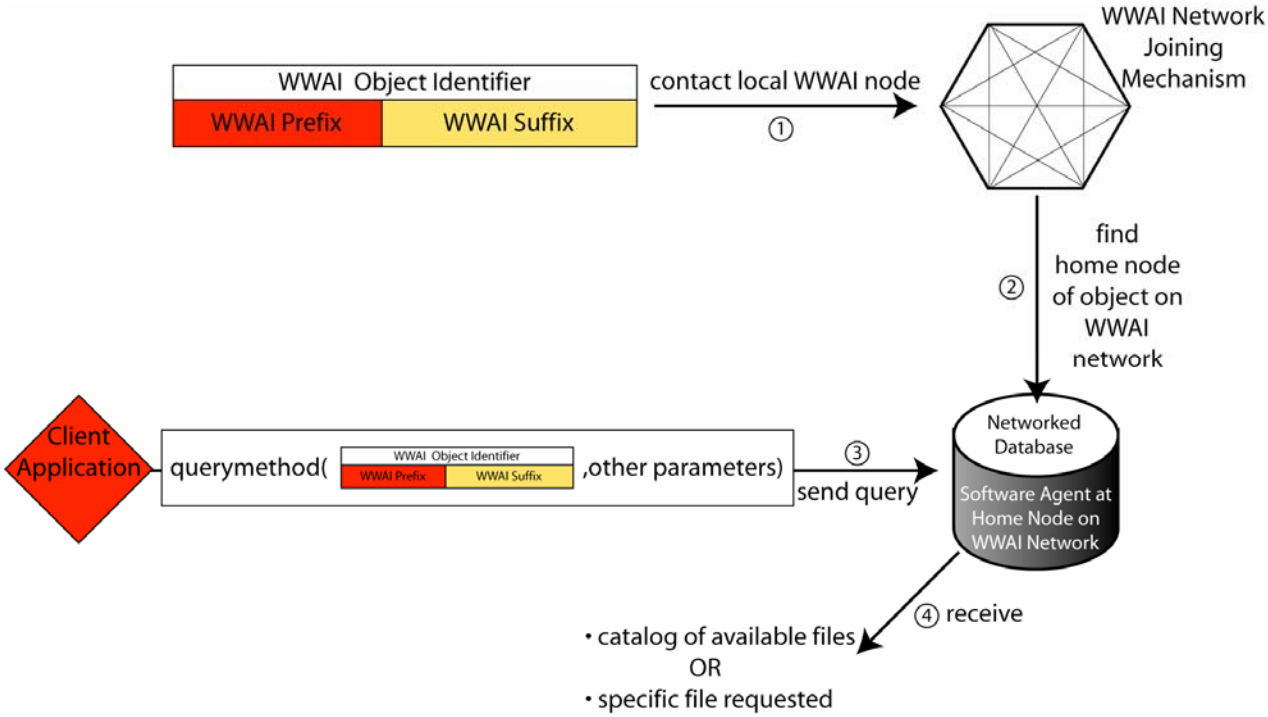

Fig 4. Overview of Stockway's WWAI approach

The WWAI approach is partially based on peer to peer networking principles. For a node to join the WWAI network and provide an information service, it must first obtain certification from a certificate authority. The basis for ensuring uniqueness of WWAI codes is based on this certification and it can roughly be considered equivalent to registration under the EPC Network approach.

In summary, there are a number of competing approaches. Currently the EPC Network approach seems furthest along in terms of market acceptance and maturity of the standards. Although it is generally considered more centralised due to its reliance on DNS, some use of DNS is inevitable in all approaches.

3.1.2 Other Content Domain Standards

Apart from the identity of the PEID, as defined by some form of naming standard discussed in the previous section, there may also be a need to define new standards or require certain existing standards for the format and structure of data as it is stored on the PEID and communicated to the upper layers.

In principle, the storage could be somewhat independent of how the data is communicated, and internal storage formats could be left to the PEID designer, but this demands that there is available function to translate between internal storage and external communications formats. However, as a minimum, it is critical that the formats used for communication are clearly defined, as this is key in ensuring interoperability

As long as data stored on a PEID is of interest or value to only a single organisation, then the format/structure/encoding etc. is of little importance as long as that organisation can retrieve and interpret it. However once there is a desire to make that data accessible across organization boundaries so that any stakeholder can access the data stored on the PEID, then data standards are essential.

The beginnings of such a data model are defined already in PROMISE in the form of the Content Service described in DR4.2 “PEID Core Prototype”. It is not critical to finalise this issue for this document; there is scope for a wider discussion and opportunity to canvass opinion and input from other PROMISE partners by leaving this item open at this stage.

3.2 Device Interoperation Domain

As already mentioned, a standardization of the device interoperability layer which allows communication between a Core PAC and the PROMISE middleware is desirable. We first need to assess which requirements are stated for such a layer. A summary of these requirements given in DR4.2 are:

1. Client-Poll model for invocation
   First of all the PROMISE middleware needs to access the functionalities of a Core PAC by the remote invocation of methods that are provided by the Core PAC. These methods are called *actions*. The access method on actions must be open, common to all actions, should be platform and programming language independent, and preferably standardized

2. Server-Push model for information
   Each remote device such as a Core PAC has specific information associated with it. The sum of all this information is called the state of the remote device, and the variables describing the state of the remote device are called *state variables*. Changes of the state should automatically be communicated by the Core PAC to the parts of the network that are interested in its state.

3. Server Advertisement
   Besides the exchange of information the interoperability layer must allow the PROMISE middleware to connect to the Core PACs in a seamless and ad-hoc fashion. Therefore a new Core PAC that enters the network should actively inform the PROMISE middleware about its presence. To this end, it will spread certain messages in the network which reveal that a new device has joined and is ready to share its services with clients. This step is called *advertisement*. Similarly, methods must be provided to un-register Core PACs from the network.

4. Client Search
   Vice versa, the PROMISE middleware may also need to search for the PEIDs in reach. This may be the case when the machinery containing the PEIDs is immobile and a mobile terminal is used e.g. for maintenance purposes. The maintenance terminal may either run the PROMISE middleware itself fully or partially. Thus, the Device Interoperability Layer must also define appropriate search mechanisms.

The desired features of the device interoperability layer can be implemented via or are already provided by numerous existing semantic middlewares. It is more beneficial for the PROMISE project to use and extend these standards rather than defining new ones. Some of the existing middleware platforms that are applicable in this area are:

- Corba (*Common Object Request Broker Architecture*) [CORBA] is a generic architecture which enables a client application to execute functions and methods on a remote server. The core of the Corba architecture is the Object Request Broker (ORB), which is respon-

sible for client-server communication. The ORB is also able to communicate between different systems and hardware architectures.

- Java RMI (*Remote Method Invocation*) [RMI] is similar to Corba but RMI is based on Java and, therefore, only supports Java-based communication between applications. It allows data and code to be transferred and executed on different platforms.
- Jini (a pseudo-acronym for *Jini Is Not Initials*) [JINI] is an extension of RMI that provides functionalities for seeking and supporting services. Jini also supports code download and, as a consequence, the integration of drivers at runtime.
- JXTA (is short for Juxtapose, meaning side by side) [JXTA] was developed on an initiative of Sun Microsystems and attempts to simplify the structure and operation of Peer-to-Peer (P2P) networks. It specifies mechanisms to find its participants and allows communication via XML-RPC.
- UPnP (*Universal Plug and Play*) [UPNP] was developed by the UPnP forum (www.upnp.org) and allows almost self-configuring, service based, peer-to-peer connectivity between networked devices using the Extensible Markup Language (XML) to describe services and to process communication. For client-server communication, the Simple Object Access Protocol (SOAP) [SOAP] is used. The eventing of state changes is done via the General Event Notification Architecture (GENA) [GENA].

For the demonstrators developed due to month 12, UPnP has been defined as the dedicated device interoperability layer. It has been chosen as it is an open, platform and programming language independent middleware architecture which provides all mechanisms needed in the PROMISE context. As UPnP poses no restrictions on the concrete implementation, neither on the hardware nor on the software side, the different partners are able to realize their UPnP interface fitting for their field of operations. UPnP is a promising candidate for the standard device interoperability layer and will therefore be assessed in the following months by reviewing its applicability in the PROMISE demonstrators.

In addition to the specification of a device interoperability layer the Core PAC has defined in DR4.2. Based on the variety of application cases and the different requirements that arise in their respective we believe that a key achievement of PROMISE is not the restriction of the cases but the provision of uniform access methods on product-related information regardless of the specific application in order to abstract the process of information retrieval and storage from the data processing tasks. To this end the Core PAC defines the access methods and information that is common to all PEIDs which can be used by the PROMISE middleware and backend layers. The concrete implementation of a Core PAC can be done specifically for each application case and therefore take its features and restrictions into account. A first definition of the Core PAC's properties can be found in Sec. 4.2 of DR4.2.

In the following months a UPnP based implementation of the Core PAC will be adopted in the PROMISE demonstrators and will be extended or amended to the arising needs. The resulting Core PAC specification could become the standardized interface for the PROMISE middleware to access general PEID information.

3.3 Promise Middleware

The standardization in the PROMISE middleware domain has several different aspects. One aspect is that the access to the devices should comply to a standard that allows unified communication with general semantics that exceed pure standards for formatting (as XML does). This will assure a standardized semantically correct communication between the Device Handling Layer of the Device Controller and the PROMISE Core PAC. Also, the access to the Request Handling Layer from the Enterprise Communication Infrastructure that is placed above should be standard

compliant in order to make Enterprise Communication Infrastructure and the PEIDs exchangeable. The standardization activities according to the PROMISE middleware should also address the handling of the data and the transformation into PLM applications, at least as far as it is possible due to the distributed character of the transformations and the possibly coupled data for correlation.

In the following part of the section the general requirements of the PROMISE middleware will be explained shortly with the related standards to use or modify.

Request and Subscription/Cancellation Handling: From the upper layers subscriptions to InfoItems can be made. This means that certain data/events will be forwarded (and possibly preprocessed) to the subscriber. The mechanisms are explained in detail in Deliverable DR6.2 (Section 5.3).

Standard Candidates: Java Message Service (JMS) can be used for the handling of the messages to/from the Device Controller. It provides mechanisms for reliable and flexible message exchange based on standardized interfaces. JMS can be a candidate for standard use. In work package R6 it will be examined if it is appropriate to wrap the PROMISE middleware as a web service in order to have standardized access to the functionalities. For the description of the interface the Web Service Description Language (WSDL) is a candidate for standard use. Therewith, a unified description of the interface can be provided and the service can be consumed independently from the platform. So far, the development in work package R6 intends to provide an interface for subscription and cancellation of InfoItems. An InfoItem contains data that could be pre-processed to a certain degree by middleware-internal services. The Business Process Execution Language (BPEL) could be one candidate for the specification of the cooperation of services from outside of the middleware. Therewith, the standard would be used for the external orchestration that partly might be available from outside. Certain parts of the interface are domain specific for product lifecycle management applications. Based on the standards for message formats and data transport these domain specific issues could be standardized as well. Further examinations are required to determine the differences of the PLM Services proposal and the interface description of the PROMISE middleware. Subsequently, domain specific issues for standardization are to identify and have to be incorporated in the standardization process. This includes message formats and schemas that apply to the PLM domain.

Service Download and Execution: In work package R6 download of code containing additional services is planned. Service download is required since the pre-processing of the data would be possible in a flexible way. The steps of pre-processing (in the middleware to filter out redundant data) should not only be fixed at design time of the middleware. Rather, the results gathered in the backend by complete analysis of operating data allow the conclusion to make the pre-processing more effective. This is reflected in additional services that should be available for download to the middleware at runtime (see also DR6.2 and DR6.1). The code and the execution environment should be standard compliant such that the code development and the subsequent execution is easily possible. In the first instance it should be possible to download code from the middleware to the PEID in order to push the data processing to the edge of the network. Subsequently, the service download from the backend or external independent software providers is examined. This would significantly enrich the functionality of the middleware and possibly disburden the superior backend systems.

Standard Candidates: Standards that should provide a platform for execution or service discovery are Universal Plug & Play (UPnP), Open Services Gateway Initiative (OSGi) and Jini. The aim of

OSGi is to provide a standardized platform for execution of different services (written in Java) that can coexist within one execution environment. So far, it is widely used already and seen as a quasi standard for execution of distributed services. Hence, OSGi is one strong candidate for standard use in order to provide a unified execution environment for services. Jini and UPnP cover the service discovery process. These protocols enable to automatically find and register services on a standards base. Since the development in work package R4 and R6 is based on UPnP for the communication between the devices and the middleware it is a candidate for standard use in the project. Nevertheless, Jini provides similar mechanisms for service discovery and binding. The development in work package R6 will further explore whether Jini is important within the project context.

Service Repository and Execution: The Service Repository hosts services that can collaborate in order to provide InfoItems to the application. Certain InfoItems might require a specific interaction of services orchestrated by the service manager and running on a unified execution platform.

Standard Candidates: Similarly to the requirements for service download and execution a standardized platform and mechanisms for service discovery and execution are required. Again, OSGi, UPnP and Jini are standard candidates that are to examine in detail. Also, Universal Description, Discovery and Integration (UDDI) can serve for storage and management of the services in the repository. Since UDDI requires a unified service description WSDL is another candidate for the standards use. Additionally, the developments in the PLM Services standard driven by the OMG should be monitored and shaping by the PROMISE consortium should be considered. It is to be studied how a distributed infrastructure for data gathering could impact the standard proposal (distributed PLM services). Furthermore, implications for further standardization should be identified. Relevant extensions of the PLM Services standard, due to the results in PROMISE, should incorporate with the standardization efforts.

3.4 Backend

At present there is no indication of demand to develop new international standards within the Backend domain. However, there are some standards that have to be considered or even agreed within PROMISE. Furthermore standards for an exchange format for field data may be aimed in order to ease some of the integration aspects.

For the integration of PDKM and other systems that manage product data the new OMG (Object Management Group) standard PLM Services [OMG] is of high relevance for PROMISE. The standard PLM Services is available as an international standard since 15.04.2005. It is the first international standard for the exchange of product lifecycle data via Web-Services. The standard defines a STEP AP214 compliant data model and all the necessary functionality to realize use cases for Collaborative Engineering. The Information Model of the proposed PLM Service is based on the STEP PDM Schema and extended by relevant subsets of STEP ISO 10303-214:2000, especially the Configuration Management modelling parts according to CC8. It defines a platform- independent model (technology independent) and a platform-specific model that is bounded to WSDL, XML and SOAP. The standard enables the standardized access to product data and basic mechanisms for manipulation of product data.
Though, aspects according to distributed services that are prevalent in PROMISE are not reflected in the standardization effort so far. For example, issues that concern the pre-processing of measurements or distributed recording of operating data should be regarded. One possibility for exten-

sion is to expand the client-server-infrastructure in the PLM Services standard by the introduction of middleware that can perform intermediate (pre-) processing.

The consideration of PLM Services in the PDKM concepts and architecture ensures the integration of systems that manage product data and thus the access to product data from different sources via PDKM in order to perform required analyses.

A file-based exchange of field data (product data from MOL and EOL phases) may require a standard for an adequate file format – at least a PROMISE-internal standard. This will enable the PDKM import functions to assign field data objects to their corresponding objects in the product lifecycle models. Moreover it needs to be analysed whether the extension of the PLM Services standard with field data aspects should be aimed for.

Product Lifecycle Support (PLCS) can be regarded as another relevant standard for the backend domain. It is a standard that also bases on ISO 10303 (STEP); furthermore it is an Application Protocol (AP 239) of STEP. 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. As it was built around STEP, 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.
PLCS seeks to provide a mechanism to maintain the information needed to support complex products and systems in line with the changing product over its complete life cycle from concept through design and manufacture to operation and disposal. This includes among other things:

- Identification and composition of a product design from a support viewpoint (as an extension of the STEP PDM Modules);
- Definition of documents and their applicability to products and support activities (as an extension of the STEP PDM Modules);
- Identification and composition of realized products;
- Configuration management activities, over the complete life cycle;
- Properties, states and behavior of products
- Activities required to sustain product function;
- Resources needed to perform such activities;
- Planning and scheduling of such activities;
- Capture of feedback on the performance of such activities, including the resources used;
- Capture of feedback on the usage and condition of a realized product;
- Definition of the support environment in terms of support equipment, people, organizations, skills, experience and facilities
- Definition of classes of product, activities, states, people, organizations and resources.

For specifying or recording required support activities through product lifecycle, a set of Assured Product and Support Information (ASPI) is defined. Lifecycle data for a specific product are composed by both ASPI and their related information, such as feedback on product history, activities and resources used.
In PLCS products are described by means of a specific Application Module: Product Structure (AM 1134). It references other AMs to define product subcomponents, their relationships, their assembly structure and many type of breakdown by which a product can be affected. Activities

are defined within AM 1047 and examples of activities are works done by people or organizations, usage of products, planned maintenance, etc. It's notable how PLCS distinguishes between future planned activities and activities that have already taken place or recently started (defined in AM 1259, Activity as Realized). Resources are required to perform a task, can be quantified, specified and are distinguished between required resources (AM 1267) and resource item (AM 1266). These resources are used by activities involving products and can represent, for example, people of support, instrumentation, software, tools for repairing products and so on.
Both products, activities and resources are characterized in terms of location, properties and state attributes (known as conditions).
PLCS addresses different objectives of the PROMISE project and could play an important role for standardization activities.

4 Conclusions

Summarizing it can be stated that in all standardization domains appropriate standards have been nominated that can be introduced into the PROMISE architecture and will contribute to its success. Nevertheless, the standards presented in this deliverable are proposals reflecting the current project state and need to be reviewed in the following month in order to assess their applicability and their acceptance among the PROMISE partners. In particular the question whether to define standards for the data stored on the PEID has to be discussed to come to a mutual agreement. Furthermore the integration of PLM services and a possible shaping influence of this uprising standard are notable issues that need to be addressed in the future.

5 References

[UPNP] Contributing Members of the UPnP Forum, "UPnP Device Architecture 1.0", June 2003
[GENA] J. Cohen, S. Aggarwal, Y. Y. Goland, "General Event Notification Architecture Base: Client to Arbiter", http://www.upnp.org/download/draft-cohen-gena-client-01.txt
[SOAP] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte und Dave Winer, *SimpleObject Access Protocol (SOAP) 1.1*. World Wide Web Consortium (W3C), http://www.w3.org/TR/2000/NOTE-SOAP-20000508
[CORBA]O. M. Group. Minimum corba specification. http://www.omg.org/, August 2002.
[RMI]I. Sun Microsystems. Java Remote Method Invocation (Java RMI), http://java.sun.com/products/jdk/rmi/, April 2004.
[JINI] Sun Microsystems, Inc. Jini. http://www.jini.org, March 2002.
[JXTA]P. JXTA. JXTA. http://www.jxta.org, March 2002.
[OMG] Object Management Group, http://www.omg.org
[OASIS] Oasis Consortium, Organization for the Advancement of Structured Information Standards, www.oasis-open.org
[POLIMI] J. Cassina, G. Chiari, S. Terzi, Holonic Product Traceability: Open Issues And A Preliminary Survey, Politecnico di Milano, Department of Economics, Industrial and Management Engineering, Italy; University of Nancy I, CRAN Laboratories, France; 2004
[PLCS] J. Dunford, Product Life Cycle Support (PLCS), Eurostep Open Day, Stockholm, 16 Sept 2002

