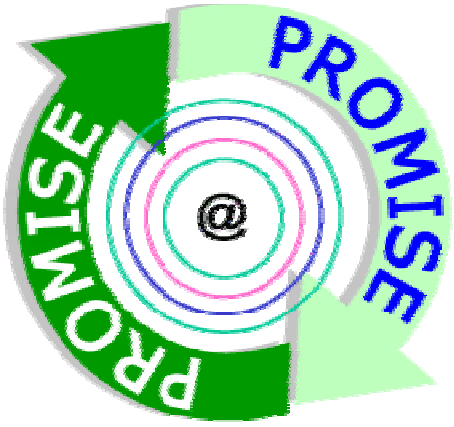


DR12.1: Test Report

Written by:
List author(s) name, organisation; List author(s) name, organisation;
List author(s) name, organisation

DELIVERABLE NO	DR12.1: Test Report
DISSEMINATION LEVEL	**CONFIDENTIAL**
DATE	06. December 2007
WORK PACKAGE NO	WP R12: Architecture Integration
VERSION NO.	1.0
ELECTRONIC FILE CODE	dokument1
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT	This document reports on the integration tests and corresponding test results concerning the integrated PROMSE architecture. According to tasks DR12.1 to DR12.3, these tests are integration tests which consider generic test cases. An overall summary is provided in Sec. 5.

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE** (dd.mm.yyyy)
SUBMITTED (author(s))	Guido Stromberg	06.12.2007
VU (WP Leader)	Guido Stromberg	06.12.2007
APPROVED (QIM)	Dimitris Kiritsis	07.12.2007

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
24.09.2007	0.1	Guido Stromberg	Document Structure and Sec. 1
28.09.2007	0.2	Daniel Barisic	Updated Sec. 2
02.11.2007	0.3	Falk Brauer	Added Section 3
02.11.2007	0.4	Hong-Hai Do	Reviewed and updated Section 3
30.11.2007	0.4	Altug Metin	InmediasP input to Sec. 1 and Sec. 4
06.12.2007	1.0	Guido Stromberg	Integration of InmediasP input and Sec. 5

Author(s)' contact information

Name	Organisation	E-mail	Tel	Fax
Guido Stromberg	Infineon Technologies AG	Guido.stromberg@infineon.com	+49 89 234 40430	
Daniel Barisic	Infineon Technologies AG			
Falk Brauer	SAP AG			
Hong-Hai Do	SAP AG			
Altug Metin	InmediasP			

Table of Contents

1 Purpose of the document

This document reports on the integration tests and corresponding test results concerning the *integrated* PROMSE architecture. According to tasks DR12.1 to DR12.3, these tests are integration tests which consider generic test cases. Application-specific tests are conducted under the umbrella of the demonstration cluster in work packages Ax.

The tests target the compatibility of PROMISE components with respect to the following interfaces:

- Integration PEID – Middleware
  The interface between PEID and PROMISE middleware is the CorePAC interface, which has been specified at month 18 in deliverable DR 4.3: "Specification of the Embedded Core PEID". An updated version of the CorePAC interface is expected at month 36 (DR4.5: Assessment and refinement of ECP specification), but has not been considered for the tests documented in this deliverable. This is not critical, since the CorePAC specification will only undergo minor changes. The amended specification will specifically remain backwards-compatible, so that no integration problems need to be expected.

- Integration Middleware – PDKM
  The interface between Middleware and PDKM is the Promise Middleware Interface (PMI). The version 1.0 of PMI has been specified in DR 6.5: "Interface definition and design of enterprise communication infrastructure" at M24. All updates of this specification will be documented in the architectural series. The PMI version 2.0 is now available (M33) and substantially extends v1.0 with improvements and enhancements obtained from the first implementation and integration experiences with the PROMISE demonstrators. All PROMISE demonstrators are currently being realized with PMI v2.0. In PMI v3.0 additionally elements for subscribing to events like PLM-events, device management events or system management events will be added. PMI v3.0 will thus incorporate all enhancements made during the project and be available at M42.

- Integration DPKM – DSS
  The DSS system has been developed using a temporary database. In order to enable the DSS system to access PDKM data, the algorithms had to be re-directed to the PDKM back-end. This integration task has been tested based on selected data elements.

2 Integration Test PEID – Middleware

For the integration tests between PEID and middleware, the interoperability between PEID-Middleware has been tested. The tests verify:

a. The compliance of implemented PEIDs to Core PAC
b. The compliance of middleware to Core PAC

We will now specify test scenarios for this in detail and then outline the results of our tests.

2.1 Description of Test Scenarios

The Core PAC interface has been specified in DR4.2. The Core PAC is a container that allows information retrieval for multiple PEIDs using three UPnP services (*Info*, *Content*, *PTinfo* service). The main functionalities that are delivered via the Core PAC are:

- Discovery: A middleware node is able to look up all PEIDs in the network and is informed when PEIDs join/leave the network.
- Description: A middleware node is able to retrieve a list of application specific content that is available on the PEID (e.g. sensor measurements).
- Information Retrieval: A middleware node is able to retrieve generic information (ID, manufacturer, etc.) as well as application specific information form the PEID.

In order to asses the Core PAC compliance of both PEIDs and middleware, the test scenarios and expected results have been defined as:

- Discovery I:
  - Setup: The middleware is running and a new Core PAC for at least one PEID is started
  - Result: The middleware gets aware of the existence and creates an entry in its internal registry for each PEID in the Core PAC
- Discovery II:
  - Setup: A Core PAC for at least one PEID is running and the middleware is started
  - Result: The middleware gets aware of the existence and creates an entry in its internal registry for each PEID in the Core PAC
- Discovery III:
  - Setup: A PEID joins a previously discovered Core PAC
  - Result: The middleware gets aware of the existence and creates an entry in its internal registry for the PEID
- Discovery IV:
  - Setup: A Core PAC that has been previously discovered by the middleware becomes unavailable
  - Result: The middleware gets aware of the change and removes the entries of each PEID inside the Core PAC in its internal registry
- Discovery V:
  - Setup: A PEID that has been previously discovered by the middleware becomes unavailable
  - Result: The middleware gets aware of the change and removes the entry in its internal registry
- Description I:

- Setup: The middleware is aware of a PEID and tries to retrieve the list of available, application specific content
- Result: Middleware receives a valid list and creates corresponding entries in its internal registry

- Information Retrieval I:
  - Setup: The middleware is aware of a PEID and tries to retrieve generic information (using the info service)
  - Result: Middleware receives a valid response
- Information Retrieval II:
  - Setup: The middleware is aware of a PEID and tries to retrieve application specific information (using the content service)
  - Result: Middleware receives a valid response

Using these test cases, we have tested the compliance for three different component setups:

- **ECP based solution - SAP Middleware**: In this setup the ECP hardware with an attached switch and temperature sensor is used. A PC is executing a proxy application (implemented using Infineon's Java UPnP Stack and Infineon's CorePAC programming framework) that supports the Core PAC. ECP and proxy communicate over a proprietary protocol on top of the TCP/IP protocol suite.
- **RFID Reader – SAP Middleware**: In this setup a Handheld RFID reader is used. Also a proxy application is executed on a PC to support the Core PAC interface. The Cyberlink UPnP Stack (Java) was used to implement this solution
- **Pure Software PEID – SAP Middleware**: In this setup a Core PAC application (C#) has been implemented using the Intel® Device Builder for UPnP™ Technologies. It implements 6 PEIDs with all in all 40 information items and is purely software based.

2.2 Coverage of the Test Scenarios

For all mentioned PEID implementations Discovery I-V, Description I, Information Retrieval I-II has been successfully tested. In all tests the PEIDs have been discovered accordingly and communication between Middleware and PEIDs could be conducted without error. Thus the compatibility of the three implemented PEIDs with the Middleware has been validated, and the general applicability and platform independence of the CorePAC interface has been shown.

As the general compatibility of PEIDs and middleware with the CorePAC interface is achieved, it is possible to submit a request to the middleware which then fetches information from a PEID and returns it to the requester. To validate that this is also possible, we have built a distributed PEID-Middleware setup. This can be seen as a simulation of an application scenario, where the products are distributed over multiple sites (e.g. cars in service garages).

To this end, a distributed PEID-Middleware setup has been created. It uses the three PEID implementations mentioned above and the SAP middleware. Following SAP's approach of the middleware, the so called Device Handling Layer (DHL) is installed in the local networks which then directly connect to the PEIDs. In our test setup, the PEIDs and one DHL each are located at Infineon's site (Munich), SAP's site (Dresden) and BIBA's site (Bremen). The so-called Request Handling Layer is also part of the middleware and uses the DHLs to fulfil requests. The RHL is executed in Karlsruhe. We then used a generic Web Services Tool accessible via the internet to invoke a query on the RHL. As expected, we were able to access and store on-line information of the distributed PEIDs, thus validating the overall PEID-Middleware setup.

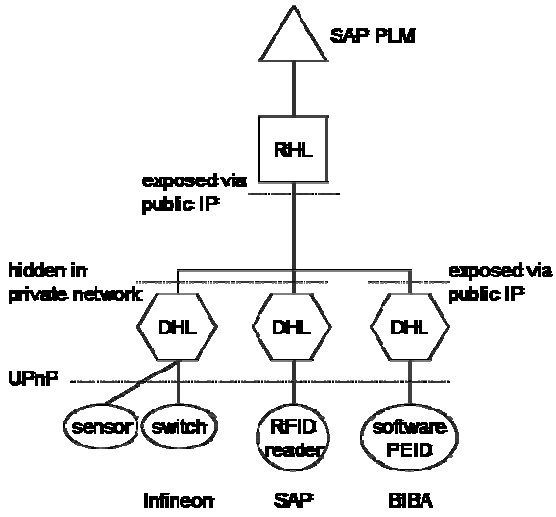


Figure 1: Distributed PEID-Middleware Setup

2.3 Test Results

In summary, the tests have proven the interoperability and suitability of the solutions that have been developed in the area of PEID and middleware within PROMISE. The interoperability tests have further shown that PEIDs with different technologies (hardware and software) are fully compatible with the PROMISE system, and that the CorePAC interface is capable to support all different flavours of PEID implementations (software, hardware, or proxy-based distributed). This validates the choice of interface and its inherent abstractions. Further, the test of a distributed PEID-Middleware setup has demonstrated that the necessary components to build PROMISE applications have been implemented correctly, and according to the specifications; further, we have thereby validated that also more complex scenarios can be executed efficiently under real-live conditions including network delays typical for large-scale distributed deployments.

3 Integration Middleware – PDKM

For the integration tests between Middleware and PDKM, the interoperability between SAP's Middleware (consists of Request Handling Layer and Device Handling Layer) and PDKM based on PMI has been tested. The tests verify the compatibility between the PMI implementations. We will now specify test scenarios for this in detail and then outline the results of our tests.

3.1 Description of Test Scenarios

The PMI v1.0 has been specified in DR 6.5. The PMI specifies an XML-format to describe requests and responses between PMI-enabled Device Controllers, different Middleware implementations, and the PDKM. It consists of elements describing communication schemes like time-to-live (for asynchronous/synchronous requests) or a Web Service call back point if a push communication is preferred. Additionally requested data and response data can be described. Main functions that are needed for communication between PDKM and Middleware are:

- **PDKM sends request**: PDKM creates a PMI Request for placing a subscription for an info item provided by a PEID.
- **Middleware retrieves request**: The Middleware retrieves a request, interprets it, places a subscription, and cares of data delivery to the PDKM.
- **Middleware sends response**: The Middleware decides that data delivery to the PDKM is needed, creates a response for the specified Web Service callback address and sends it to the PDKM.
- **PDKM retrieves response**: The PDKM receives the response of the Middleware, interprets it, extracts the data value from the response, and stores the measurement point in the PDKM.

```
<pmiEnvelope type="readData" version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="request.xsd">
        <readDataRequest  interval="1" ttl="3" wsCallBack=
        "https://sapportal.inmediasp.de/irj/servlet/prt/soap/com.sap.portal.prt.soap.XMLMiddleware"
        requestTargetType="device">
                <targetDevices>
                        <targetDevice>
                                <ids>
                                        <id>ECU_9</id>
                                </ids>
                                <infoItems>
                                        <infoItem>
                                                <id>FUEL</id>
                                        </infoItem>
                                </infoItems>
                        </targetDevice>
                </targetDevices>
        </readDataRequest>
</pmiEnvelope>
```

Listing 1: PMI Subscription Request

In order to assess the compatibility of SAP's PMI implementation and PDKM's implementation the following use cases have been defined and tested:

- **PDKM sends request and Middleware retrieves request**:

- *Setup*: The DHL is running within an intranet. The RHL is reachable from the internet, the PDKM is able to communicate to the internet (e.g. over a proxy server) ;
- *Communicated data*: The request sent to the middleware contains a target PEID, an info item, and a Web Service callback point. Further information can be seen in Listing 1.
- *Result*: The middleware is able to interpret the request and to create a subscription to the PEID, so that the request can be answered as soon as the PEID is online and can provide the required data.

- **Middleware sends response and PDKM retrieves response**:
  - *Setup*: The DHL is running within a intranet. The RHL is able to connect to the internet (maybe over a proxy server). The PDKM is reachable over internet. A PEID with subscribed data has connected to the DHL
  - *Communicated data*: The request sent from the middleware contains the subscribed data, the request id, the targeted PEID and the info item. Further information can be seen in Listing 2.
  - *Result*: The middleware is able to detect the need of sending data and is able to send the response to the PDKM. The PDKM is able to interpret the response and to place the data in the PLM system.

```
<pmiEnvelope type="dataResponse" version="1.0">
        <dataResult type="read">
        <requestId>56</requestId>
                <result>
                        <targetDevices>
                                <targetDevice>
                                        <id>ECU_9</id>
                                        <infoItems>
                                                <infoItem>
                                                <id>FUEL</id>
                                                        <value timestamp="2007-08-29 13:16:35">
                                                                100
                                                        </value>
                                                </infoItem>
                                        </infoItems>
                                </targetDevice>
                        </targetDevices>
                </result>
        </dataResult>
</pmiEnvelope>
```

Listing 2: PMI Response Structure

The test use cases described above cover all needed functionality for implementing demonstrators and real world applications. The network infrastructure set up of the test use cases is not trivial because the communication has to bridge several firewalls and proxy servers. Additionally the communication over the internet needs to be secured.

Both use cases have been successfully tested. They assess the compatibility of SAP's and InmediasP's implementation of the PMI and show the proof PMI's concepts. As SAP and HUT also evaluated the compatibility of their Middleware by using the other ones request to call their Middleware, intersystem communication compatibility between these systems is ensured. With this, we have shown that the PMI specification and subscription concept developed in the Research Cluster are applicable to real world scenarios.

3.3 Test Results

In summary, the integration tests of Middleware and PDKM have shown the interoperability and suitability of the communication solutions that have been jointly developed in the Research workpackages R4, R6, and R9 in PROMISE. The feasibility and viability of the PMIs concepts have been proven by the tests. This validates the choice of the interface and its inherent abstractions. In the next steps, different technical configurations of the different components (PEIDs, Middleware, and PDKM) will be specified for further comprehensive tests for the integration of the components. Further experiences will be collected from the process of implementing the PROMISE demonstrators. All enhancements and improvements of PMI will be incorporated into the version 3 of the PMI and documented in the Architecture Series of WPR12.

4 Integration DPKM – DSS

4.1 Description of Test Scenarios

The PDKM framework and the DSS system have been developed within different work packages (R9 and R8), although DSS is considered as a part of the PDKM framework. Due to this architectural point of view, some integration work is required in order to make these systems work together.
According to the Description of Work, the PDKM system has been developed based on an existing PDM system. The chosen system was mySAP PLM which is part of SAP ECC and the work in R9 was based on this choice. The result of the work in R8 will be integrated into the PDKM framework.
The development of the DSS system is carried out by using a temporary database but in the final DSS solution, DSS algorithms access data that is stored in PDKM. The design of the temporary database was crucial because the intention was to allow the seamless change-over to the PDKM database when the integration is carried out. During the development period the temporary database was acting like the PDKM database. In order to reach this goal the tables of the temporary database had the same layout as the back-end tables of the PDKM system.
As the development of both systems matures and the DSS algorithms and PDKM back-end structures are finalized, the method for data retrieval in the DSS system should be changed. In this concrete case, the DSS algorithms which are processing the retrieved data are re-directed so that they access the PDKM database directly. The re-direction of the DSS algorithms and the correct retrieval of PDKM data have been tested for selected data fields in the scope of the PDKM-DSS integration tests.

4.2 Coverage of the Test Scenarios

The goal of the DSS-PDKM integration is to enable the DSS algorithms to access PDKM back-end data. In order to accomplish this task, the semantic mapping between the PDKM object model and mySAP PLM objects has been used. The utilisation of the semantic mapping is important since the developers of DSS algorithms are not experienced PDKM users and it is not indicated which data object in the DSS system corresponds to which back-end data object in the PDKM system.

Tabelle 1: Mapping of object model terminology

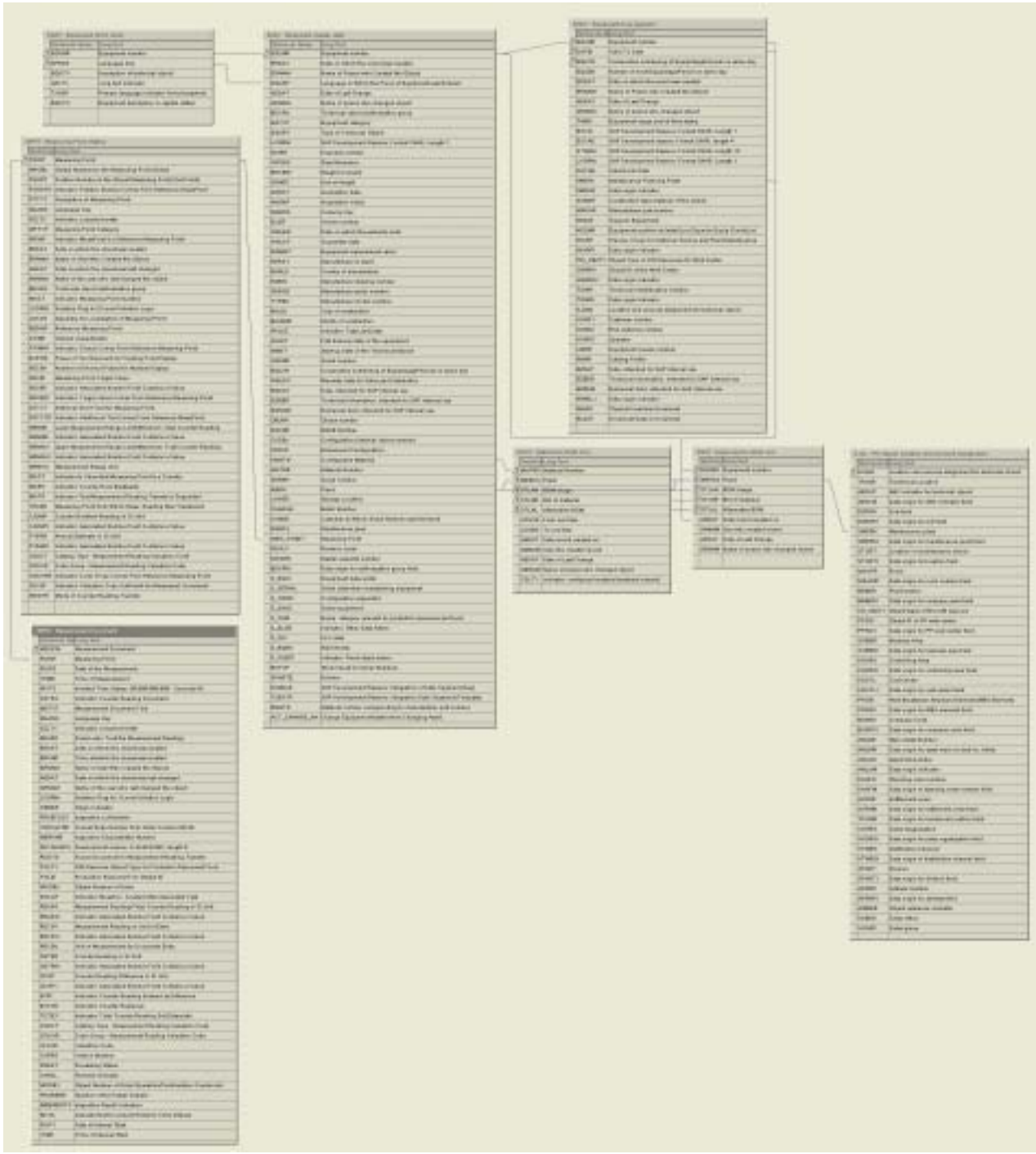
Semantically (examples)	SAP	Object model (DR9.2)	Comment
specific product type	Material	AS_DESIGNED_PRODUCT	
as-designed product structure	Bill of Material, BoM	self-association of AS_DESIGNED_PRODUCT	
individual product	Equipment with Serial Number, shortly Equipment	PHYSICAL_PRODUCT	
as-build/as-used (shortly “as-used”) product structure	Installed Base	self-association of PHYSICAL_PRODUCT	
product templates	Equipment	AS_DESIGNED_PRODUCT	From SAP-

Semantically (examples)	SAP	Object model (DR9.2)	Comment
	Template (PDKM-term)		system point of view an Equipment Template is an instance of an Equipment semantically recognizable as Template.
product template structure	Installed Base Template (PDKM-term)	self-association of AS_DESIGNED_PRODUCT	analogue to Equipment Template
customisable subset of metadata of products	Classification	PROPERTY	
document metadata	Document Info Record	DOCUMENT or (depending on the context) DOCUMENT_RESOURCE with associated DOCUMENT	
document (in contrary to document metadata), (physical) file	Original	FILE	
field data in the form of a single value	Measuring Document	FIELD_DATA	
field data in the form of a document	Document Info Record semantically recognisable as field data with associated Original	FIELD_DATA with associated DOCUMENT with associated FILE	
type of field data	Characteristic	VALID_FD_TYPE	
a sensor/an info item of a PEID	Measurement Point	FD_SOURCE	
incident, event	Notification	EVENT	
type of incident, event	Notification Type	(no correspondence)	
additional (field) data describing an incident, event	Item	FIELD_DATA	
types of possible additional (field) data describing an incident, event	Code	VALID_FD_TYPE	
classes of types of possible additional (field) data	Code Groups	(no correspondence)	

Semantically (examples)	SAP	Object model (DR9.2)	Comment
describing an incident, event			
knowledge	depending on the context Notification, Measuring Document, or Document Info Record with associated Original(s) semantically recognisable as Knowledge	depending on the context FIELD_DATA or EVENT with eventual associations	Certainly all product-related data stored in the system is accessible for the user. Context-dependent the methodical access to it might result in knowledge.

The dependencies of some of the essential PDKM back-end tables are illustrated in the following table.

Tabelle 2: PDKM data model (extract)



4.3 Test Results

The configuration of DSS algorithms according to the mapping has been performed and for selected data elements the database access has been re-directed to the PDKM back-end. As a result of this, data from the PDKM system could be retrieved and processed in the DSS system.

5 Overall test summary and conclusion

Integration tests aim at potential weaknesses on two levels: firstly, on the level of interface specifications (e.g. applicability, generality, feasibility, preciseness); secondly, on the level of component implementations (correctness).

This report shows that the development of PROMISE components has matured to a grade which ensures that

- various implementations of each PROMISE components, developed by different partners, can interact with the adjacent PROMISE layer components, which **validates the preciseness of interface specifications**;
- the implementations of PROMISE components fulfil the PROMISE interface specifications, which **validates the feasibility of interface specifications**;
- implementations of PROMISE components can be exchanged, and various PROMISE components can co-exist without interfering, which **validates the generality of the interface specifications**;
- implementations of PROMISE components can operate under synthetic real-life conditions, which **validates the applicability of interfaces**;
- **implementations of PROMISE components are correct** regarding the described synthetic test scenarios.

As all test have been defined thoroughly (prior to their execution) to provide a coverage that is similar to the requirements of typical applications, further generic integration test are not assumed to bring about significant insights. Therefore, further tests will be conducted in an application specific way within the demonstrator developments.