# DA9.6: Implementation of the PLM Process model for the Demonstrator

**Written by:**
**Dimitra Pli, Intracom Telecom**
**Christos Pronios, Intracom Telecom**

| | |
|---|---|
| **DELIVERABLE NO** | DA9.6: Implementation of the PLM Process model for the Demonstrator |
| **DISSEMINATION LEVEL** | **CONFIDENTIAL** |
| **DATE** | 05.06.2008 |
| **WORK PACKAGE NO** | WP A9: PROMISE MOL information management for Telecom equipment |
| **VERSION NO.** | V1.0 |
| **ELECTRONIC FILE CODE** | DA9.6_v.1.0_FINAL.doc |
| **CONTRACT NO** | 507100 PROMISE<br>A Project of the 6th Framework Programme Information Society Technologies (IST) |
| **ABSTRACT** | This deliverable (DA9.6) summarises the implementation of the PLM process model for the demonstrator, in terms of scenes, PROMISE components and technology implemented, as described in DA9.3 and DA9.4. The motivation for eventual discrepancies is given, together with the detailed results of the activities performed for the implementation. |

| **STATUS OF DELIVERABLE** | | |
|---|---|---|
| **ACTION** | **BY** | **DATE** (dd.mm.yyyy) |
| **SUBMITTED** (author(s)) | Dimitra Pli, Christos Pronios | 14.05.2008 |
| **VU** (WP Leader) | Dimitra Pli | 04.06.2008 |
| **APPROVED** (QIM) | Dimitris Kiritsis | 06.06.2008 |

## Revision History

| Date (dd.mm.yyyy) | Version | Author | Comments |
|---|---|---|---|
| 20.04.2008 | V0.1 | Christos Pronios | Initial Draft |
| 05.05.2008 | V0.32 | Christos Pronios | Additional input Chapters 2&3. |
| 15.05.2008 | V0.5 | Christos Pronios | Revised chapter 2.1 |
| 30.05.2008 | V0.6 | Dimitra Pli, Christos Pronios | Revised chapter 2.2 & 2.3 |
| 03.06.2008 | V0.7 | Christos Pronios | Revised Chapter 3 |
| 04.06.2008 | V0.9 | Christos Pronios, Dimitra Pli | Cross referenced Ch.2 with DA9.4 & DA9.3 |
| 05.06.2008 | V 1.0 | Christos Pronios, Dimitra Pli | FINAL |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## Author(s)' contact information

| Name | Organisation | E-mail | Tel | Fax |
|---|---|---|---|---|
| Dimitra Pli | INTRACOM TELECOM | dimpl@intracom.gr | +302106674370 | +302106677101 |
| Christos Pronios | INTRACOM TELECOM | hpro@intracom.gr | +302106671456 | +302106677101 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

## Table of Contents

## List of figures

## List of Tables

## Abbreviations

Abbreviations used in this document:

| | |
|---|---|
| DSS | Decision Support System |
| MOL | Middle of Life |
| PDKM | Product Data Knowledge Management |
| PEID | Product Embedded Information Device |
| PLM | Product Lifecycle Management |
| ADARES: | Advanced Action Request System |
| APU: | ATM Processing Unit (ATM: Asynchronous Transfer Mode) |
| BE: | Backend System |
| DSS: | Decision Support System |
| ETL: | Extract-Transform-Load |
| FT: | Field Technician |
| IBAS: | INTRACOM Broadband Access System |
| ID: | Identifier |
| PCIM : | Promise Component Identification Media *(EQUIV: RFID Tag)* |
| PCIS: | Promise Component Identification System *(EQUIV: RFID Reader)* |
| PDKM: | Product Data Knowledge Management |
| PEID: | Product Embedded Information Device |
| PLM: | Product Lifecycle Management |
| PPIM: | Promise Product Identification Media *(EQUIV: PEID )* |
| PPIS: | Promise Product Identification System *(EQUIV: PEID Reader)* |
| PT : | Production Technician |
| RFID: | Radio Frequency Identification |
| RLT: | Repair Lab Technician |
| SIS: | System Inter-registration System |
| PMI | Promise Middleware Interface |

# 1 Introduction

## 1.1 Purpose of this deliverable

## 1.2 Objective of demonstrator

The central objective of Demonstrator A9 was to evaluate a Promise-based system that could bridge all the phases during a product's MOL. In this respect it intended to integrate information from different information sources and use all of the main Promise components, namely PDKM/DSS, Promise Middleware, PEID and RFIDs.

# 2 Description of the demonstrators

This section presents detailed information on the final version of the Demonstrator, as compared to planned and designed features presented in DA9.3 and DA9.4. For the reader's easier comparison with previous deliverables (and to assist the writer in compiling all differences), all sections headers presented in DA9.4 are also shown here in the same order with the appropriate comments.

## 2.1 Scenes implemented

The following are the scenes described in DA9.3 with information on final implementation differences from previous deliverables.

### 2.1.1 "Production" Scene

This scene comprises all in-company areas where iBAS product Component-handling, of any nature, takes place. Product Components go through several phases during production and more importantly (for the Promise case) later during HW repairs, SW amendments, etc. Transitions between those phases, as well as actual information retrieval, updating and storage during those, are greatly eased by the use of Promise Hardware Identification. As far as the Demonstrator is concerned, operations on components within the Production division are almost identical to operations within the Repair Laboratory.

The scene has been implemented with two RFID Reader stations at different locations tracking 2 different iBAS components. Initialization of the component (process P10) is NOT implemented in the Scene and it was performed manually. Remaining information entering faces are implemented.

#### 2.1.1.1 "Production" Use Case

PCIM is attached to component to be tracked. This PCIM is subsequently read by suitable PCIS in order to Track the component whenever it is transferred to another "area" within the Production facilities. This "reading" also offers a human user of the PCIS, the chance to retrieve component-relevant information from backend systems as well as alter this information. Whenever a component is sent to the Repair Laboratory, the same identification process takes place.

#### 2.1.1.2 “Production” Constituent Processes[1]

P7 implemented.
P8 modified to include only ID on tag.
P9 NOT implemented due to insufficient functionality of performing needed operation to PDKM through the middleware.

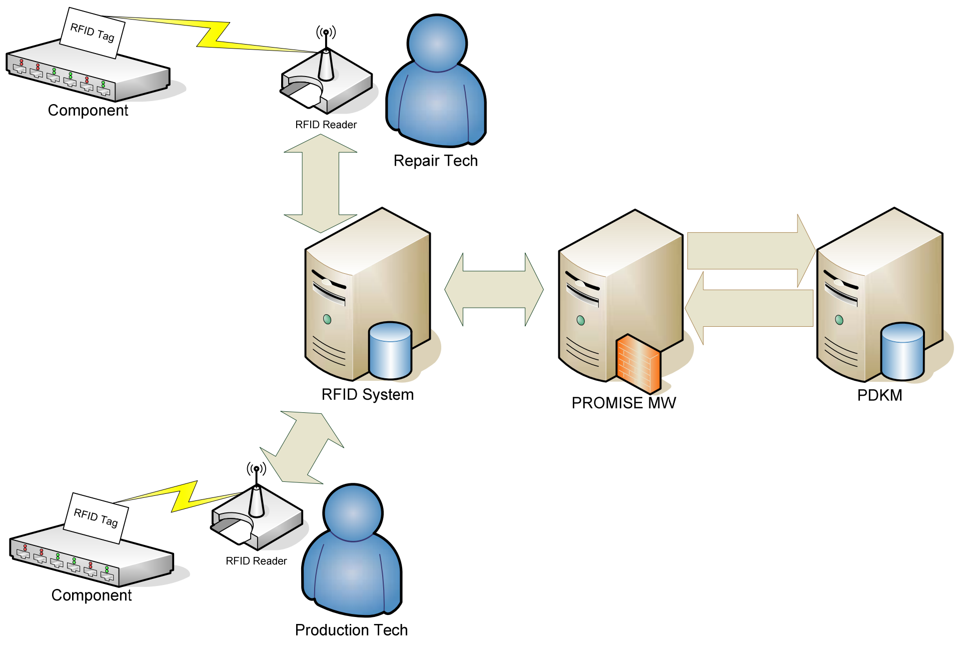

**Figure 1 “ Production” Demonstrator**

### 2.1.2 “Onsite - iBAS Operation Area” Scene

Comprises every entity active in the area where the iBAS product is actually installed and operating. It is usually an indoor or enclosed space where several iBAS racks are located, in a controlled or uncontrolled environment.
The iBAS unit is installed and configured by the Field Technician, who uses the PPIS to insert selected site data to the PPIM. Thereafter, iBAS remains in operation and its APU transfers all vital fault and performance data to the Promise PPIM. The Field Technician connects later to the Promise Middleware and transfers collected PPIM data from the PPIS to the PDKM.

#### 2.1.2.1 “Onsite - iBAS Operation Area” Components

Field Technician, iBAS APU, Promise Product Identification Media, Promise Product Identification System, Promise Middleware, Promise PDKM

#### 2.1.2.2 “Onsite - iBAS Operation Area” Use Case

The FT visits the installation site and installs the product onsite as per Product’s Type. ProductType contains all information about configuration. As soon as the product is set in operation, the FT uses the PPIS to communicate with the Promise PPIM and Write location data. He then uses the Reader to read data from the PPIM in order to Verify that it is communicating with the APU as planned. When communication with the Promise Middleware can be established

[1] For ALL Processes mentioned in this and similar subsequent sectionsn by their identifier only (e.g. P7, P8, P9 in the present case), please see later in Page 14 of this document **Table 1: Input-Output & Organizational Information for Figure 10 Processes**.

after returning to the Company premises, the FT uses the PPIS, connects to the Promise Middleware and Uploads all collected data to the PDKM.

The INSTANTIATE use case developed is simplified. The MWAuthenticate use case developed is simplified as per MW development.

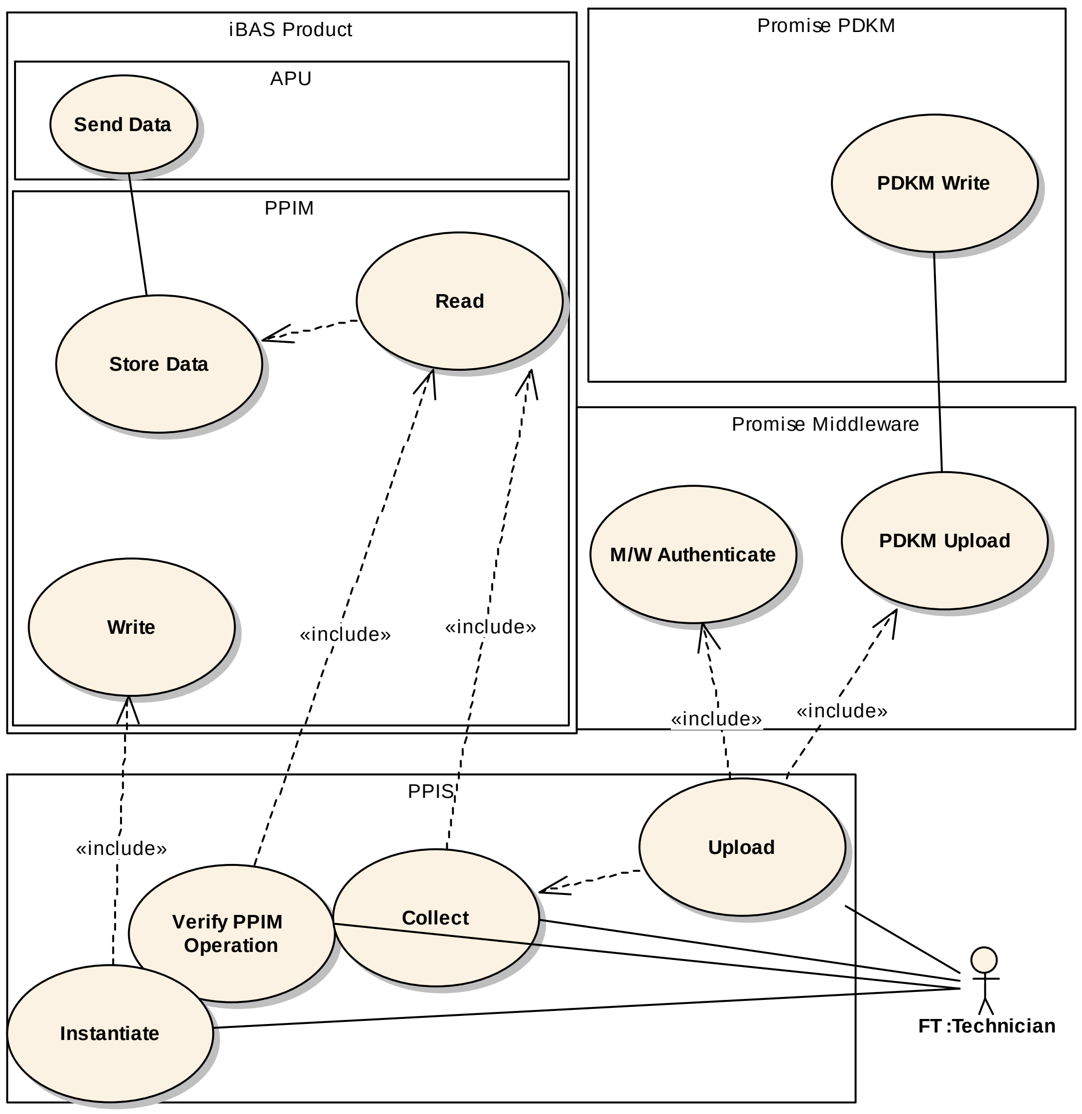

**Figure 2 “*Onsite - iBAS Operation Area*” Use Case**

### 2.1.2.3 “Onsite - iBAS Operation Area” Constituent Processes

P1, P2, P4 Implemented.

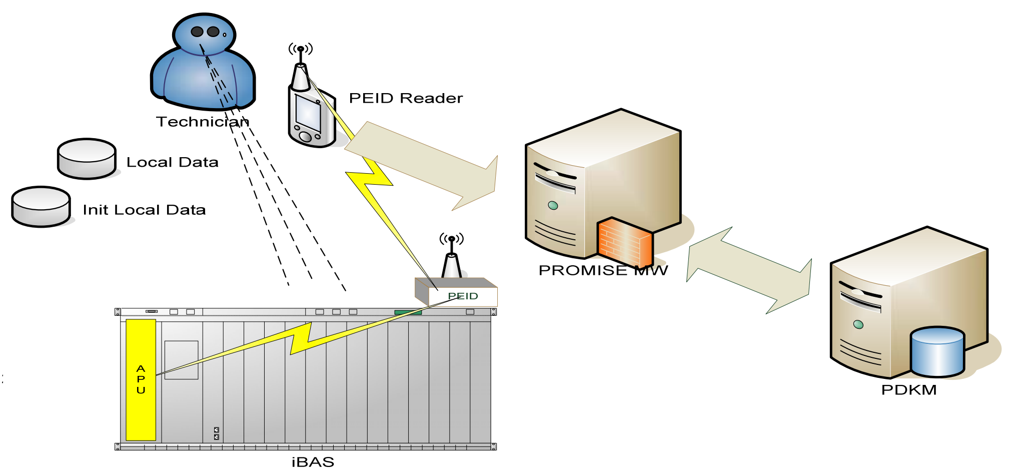

**Figure 3 "Onsite - iBAS Operation Area" Demonstrator**

### 2.1.3 "Company Systems" Scene

ADARES & SIS backend systems are inserting data to the PDKM.
At predetermined periods (batch), the Proxy server gets new data from backend systems.
The Proxy Server Connects to the PDKM for Upload of all data. Transferred data are Stored in appropriate repositories in the PDKM.

#### 2.1.3.1 "Company Systems" Components

Technician on Duty, Backend Systems Proxy Server, Promise PDKM.

#### 2.1.3.2 "Company Systems" Use Case

Data from BE systems (component maintenance/upgrade data in SIS and support call workflow, events and outcome from ADARES) are transferred to a Backend System Proxy Server that is used for communication of those systems with the Promise infrastructure.

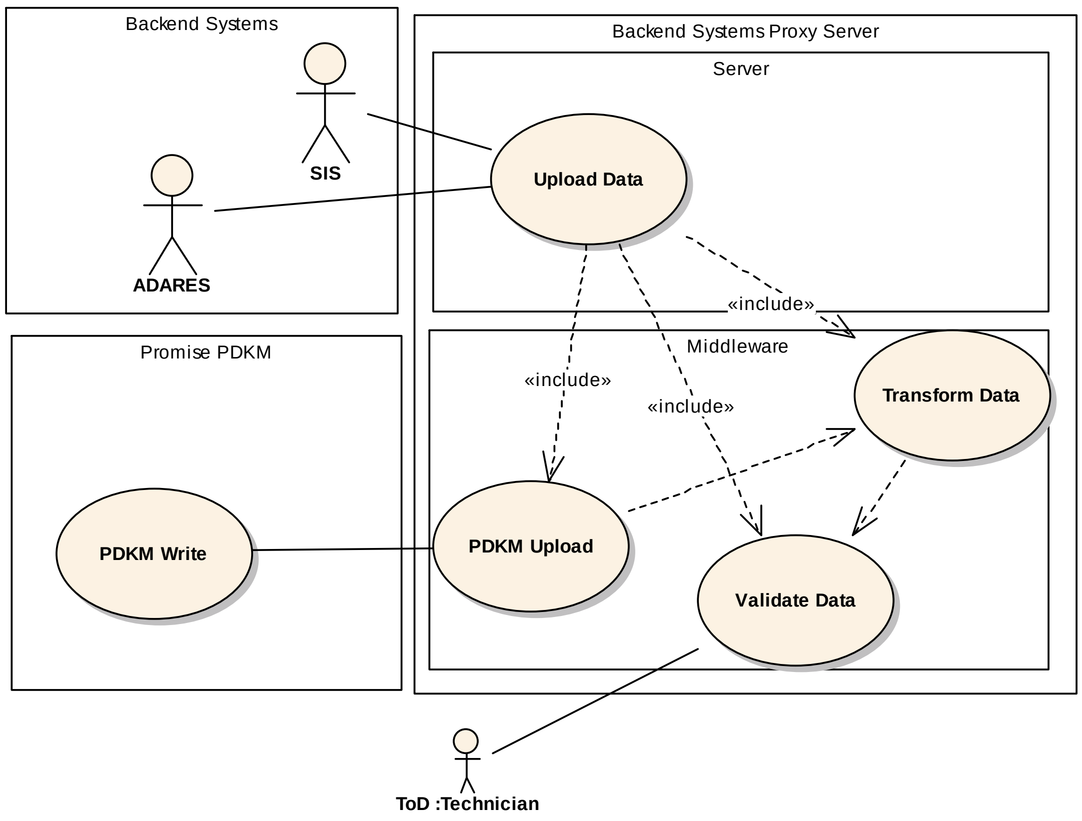

**Figure 4 “*Company Systems*” Use Case**

These transfers take place in batch mode on user request, based on data specified in DA9.3 & 4. SIS’s physical implementation does not allow for easy transfer functionality. Furthermore, both systems are under constant revision and any changes need to be migrated in Promise specific code which is not feasible. The Proxy Server comprises a Server part which receives the uploaded data and stores locally in persistent storage, and a TRANSFORMATION part that is responsible for:

- Initial validation of the uploaded data
- Basic transformations so they can be transferred to the PDKM
- The actual Upload to the PDKM.

LOG validation and recording was not implemented.

### 2.1.3.3 “*Company Systems*” Constituent Processes

P6

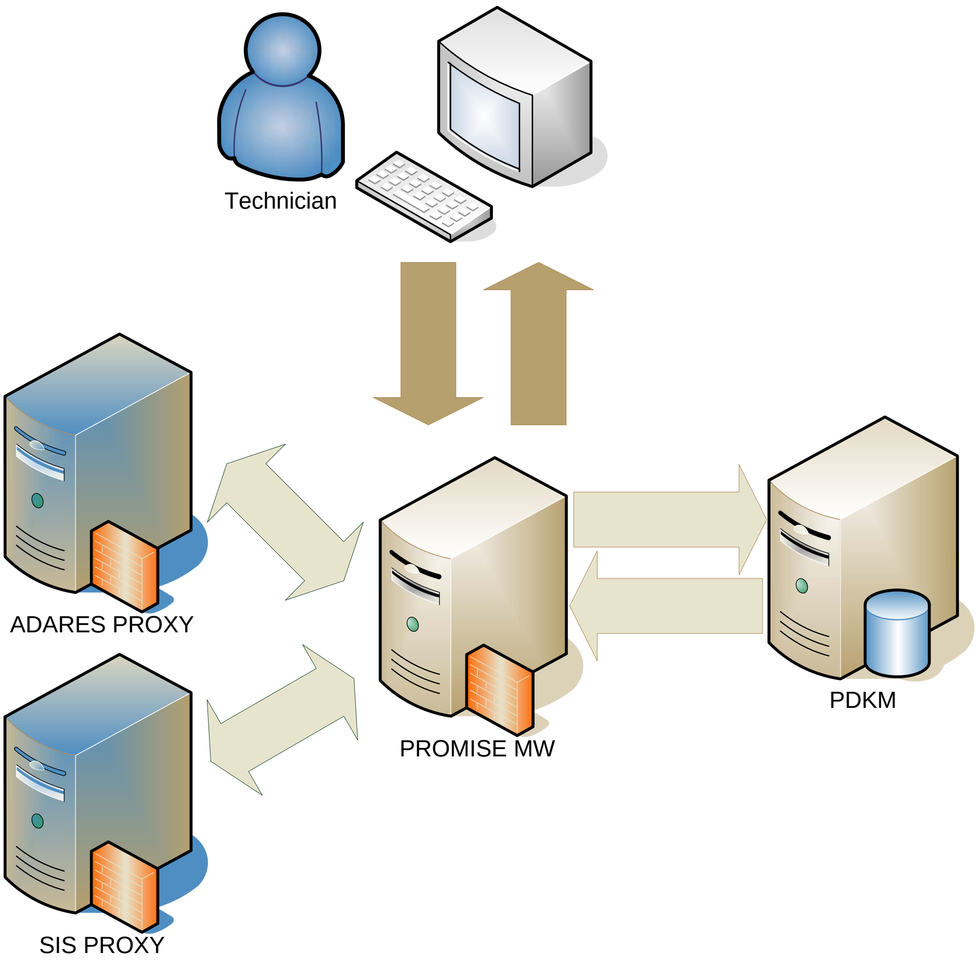

**Figure 5 “*Company Systems*” Demonstrator**

PROMISE MW shown above (and in DA9.4 & DA9.3 in same figure) is not used. The MW design does not allow for transfer of back-end data. Instead, a remote connection to the PDKM and uploading of PMI-XML files generated by the Proxy is used.

### 2.1.4 “Onsite - iBAS Support Visit” Scene

The main actor here is the Field Technician who visits the site responding to a call for support on a malfunctioning iBAS unit or simply on a routine preventive maintenance visit. The FT uses the portable *PPIS* and read the PPIM from the product.
In the case of a Malfunction response visit, the FT a can either proceed unassisted in the repair of the malfunctioning system or, if needed, seek expert assistance on the potential course of action for resolving the problem.

#### 2.1.4.1 “Onsite - iBAS Support Visit” Components

Field Technician, IBAS APU, Promise Product Identification Media, Promise Product Identification System, Promise Middleware, Promise PDKM, Promise DSS

#### 2.1.4.2 “Onsite - iBAS Support Visit” Use Case

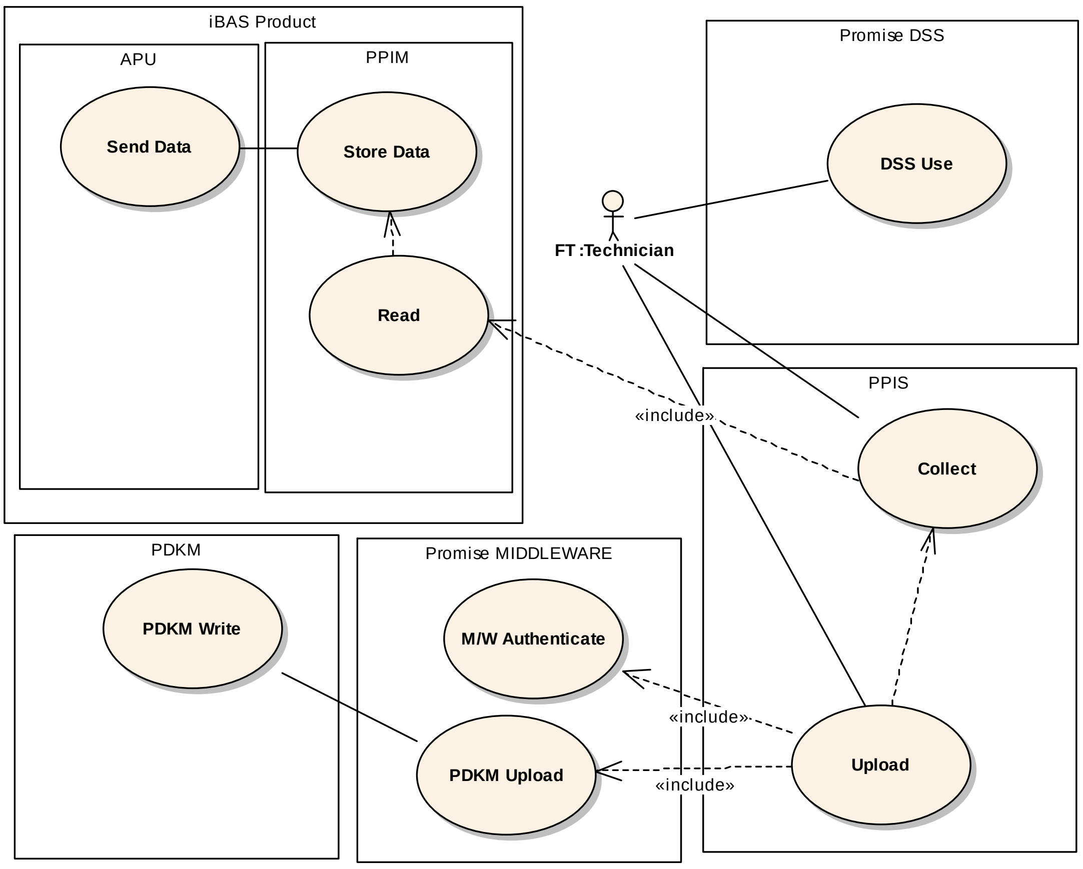

**Figure 6 “*Onsite - iBAS Support Visit*” Use Case**

DSS Use above is described as a distinct Scene in the next sections.
All other UCs are implemented.

### 2.1.4.3 “*Onsite - iBAS Support Visit*” Constituent Processes

P1, P2, P3, P4, P5, P9

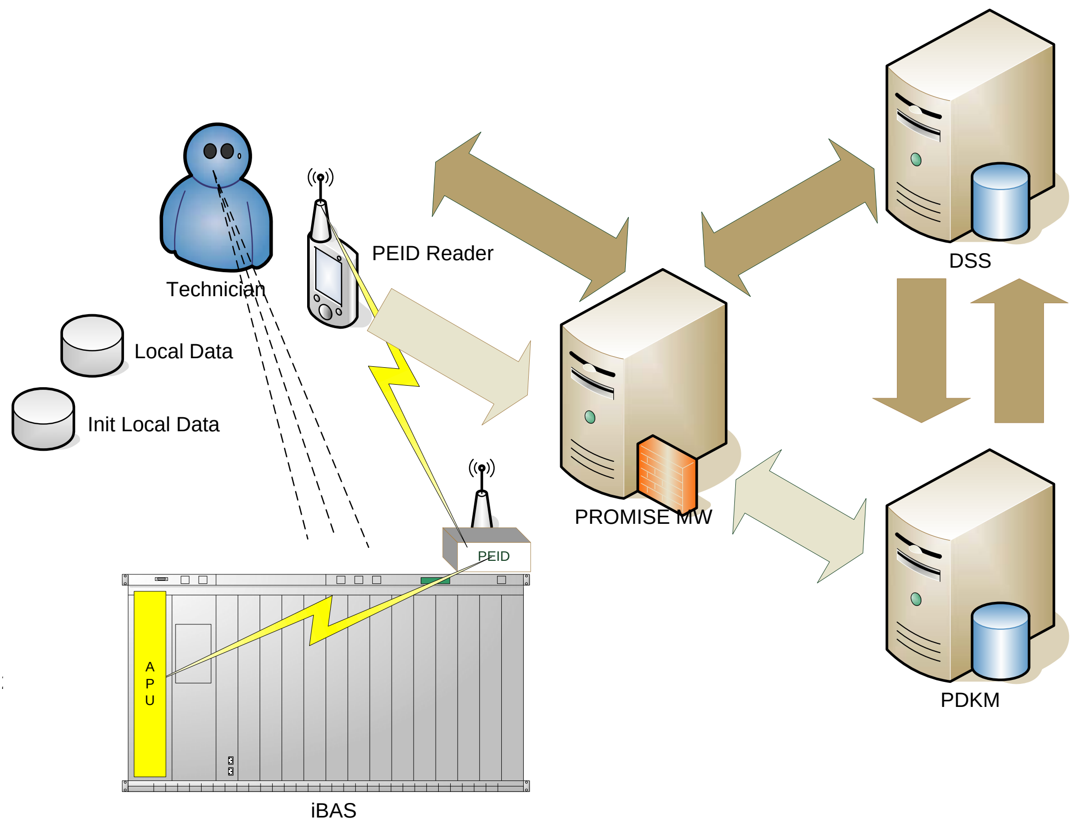

**Figure 7 " *Onsite - iBAS Support Visit*" Demonstrator**

### 2.1.5 "DSS Use" Scene

This scene is not temporally or spatially restricted since DSS use can happen in various modes and in various times, for example:

- locally, from a dedicated desktop terminal by a Duty Technician assisting a Field Technician
- locally by an expert Technician trying to assess the indicators provided by the DSS
- Remote access to the DSS by FT is not implemented.
- Product-improvement related access not implemented.

The DSS user seeking assistance uses a suitable user interface to describe the problem at hand as accurately as possible and in as many "dimensions" as possible. Then he chooses whether he is seeking a suggested problem solution or a presentation of distilled information regarding "similar" situations.

#### 2.1.5.1 "DSS Use" Components

DSS User, DSS Access Terminal, Promise PDKM, Promise DSS

#### 2.1.5.2 "DSS Use" Use Case

The DSS user seeking assistance uses or any other means available to Describe the problem at hand as accurately as possible and in as many "dimensions" as possible. The DSS provides suggested solutions to the described problem. The DSS User can then Assess the validity/applicability of the DSS-Suggested solution and enter the assessment information in the DSS. "Manual" DSS Search Narrowing implemented by choice of attributes for search..

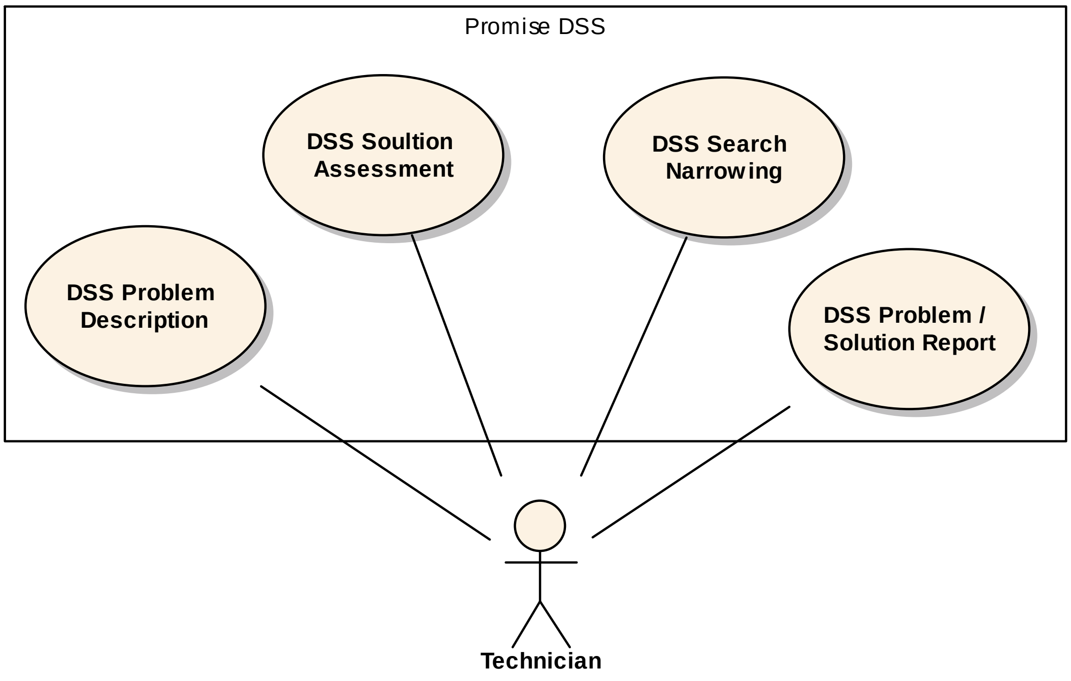

**Figure 8 "*DSS Use*" Use Case**

DSS Problem Report and the DSS Solution Report implemented through PDKM data entry.
DSS failures are implemented in the PDKM data schema for later use but not used in the Demonstrator.

### 2.1.5.3 "*DSS Use*" Constituent Processes

P5, P11

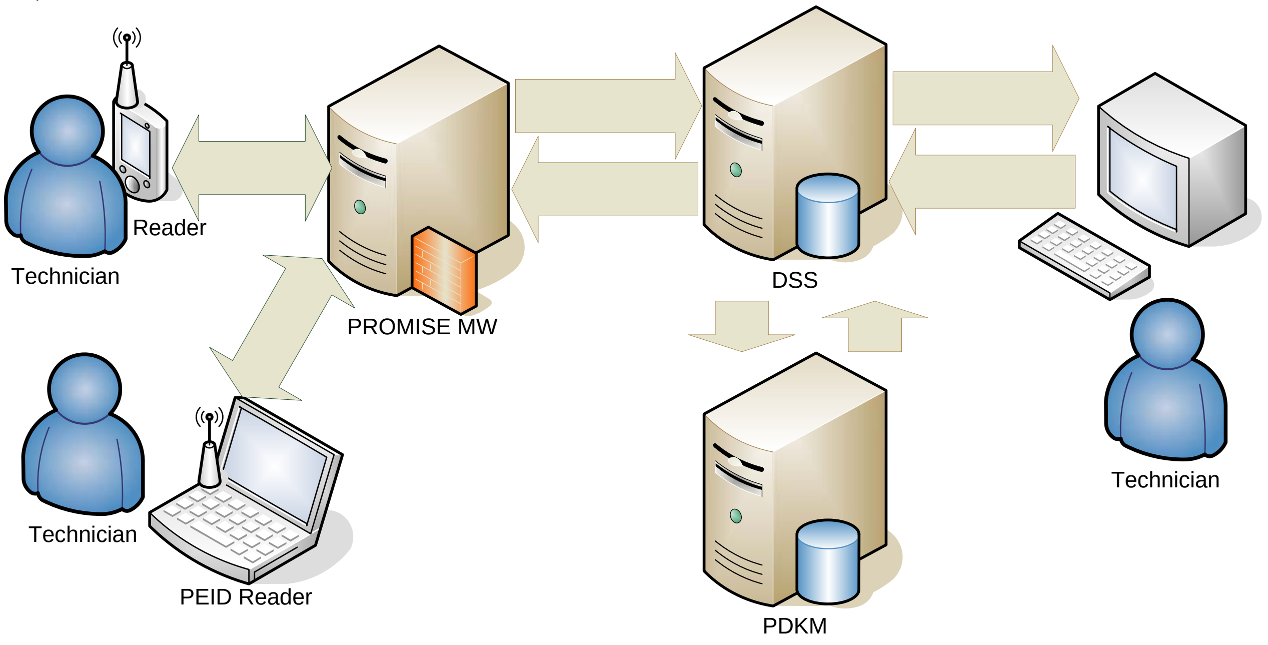

**Figure 9 "*DSS Use*" Demonstrator**

Readers shown above NOT used in demonstrator due to insufficient functionality of MW to transfer required data types to the PDKM/DSS. Readers transfer ony performance/fault data from PPIMs to the PDKM.

## 2.2 PROMISE components used

This section summarizes the actual Promise components that are used in the aforementioned scenes and highlights where components were not developed, used, integrated. It also presents a summary of distinct components that were not mentioned exclusively in previous deliverables but were developed during integration and testing to assist the demonstrator. Emphasis is given in specific A9 characteristics of the components and is not repeated for Promise-wide components.

- **Passive HF RFID tags** (TAGSYS) attached to components. and TAGSYS **RFID readers**
  - **RFID device controller** provided by INDYON.
  - Software components written in C# (dotNET)
  - A Server implemented as dotNET web service, handles communication with peer, accepts PMI requests from client and handles all synchronous requests. Handles HTTP, SOAP1 and SOAP2 message. Can secure the communication trough HTTPS and expose public interfaces in form of WSDL.
  - A daemon implemented as command line program which polls the RFID reader und forwards requested subscription data to the corresponding peer (as a web service client).
  - Both components use a same set of library functions (DLL's) – for PMI processing
  - Persistent data and interprocess communication between the 2 components is realized with a **SQLite database**.
  - PMI requests restricted to only readmetadata-requests and readdata-requests (synchronous or as subscription). No recursion within a request is allowed
  - Only 1 "<targetdevices>" and 1 "<nodes>" is allowed.

- **Sindrion PEID** attached to iBAS product provided and developed by Infineon.
  - - **CorePAC UPnP device controller** interacting to **SAP Middleware** for transfer of performance and fault data collected from iBAS (written in Java).
  - **Ethernet-to-RS232 Protocol Converter** (Axis product) for connecting APU to Sindrion.
  - **APU Data Transformation component** for interrogating the APu and converting SNMP responses to coded format suitable for handling by Sindrion.
  - **PEID Reader application** for interaction with PEID data and later transfer to the PDKM
- **Back-end Proxy persistent storage** and required applications that act as internmediaries between actual back-end data and the PDKM. Storage implemented in MS SQLServer and applications in C# (.NET)
- A9-specific **DSS algorithms for similarity searching**, developed by EPFL
- A9-designed **PMI-XML** developed by **INTRACOM** for backend data transfers to PDKM
- **PDKM**, developed by InmediasP on SAP-PLM infrastructure and hosted by SAP
- **DSS**, developed by Cognidata and hosted by SAP, integrates the EPFL algorithms
- DSS and PDKM **GUI** developed by SAP on SAP NetWeaver platform.

## 2.3 Eventual other modifications with respect to DA9.3 and DA9.4

In the following sections, we highlight workflow changes that resulted from integration and testing efforts with respect to previous DA9.# deliverables. Again, same subsection titles and order are used as before to assist the reader.

### 2.3.1 EPC Diagram (Level 1)

Same EPC as presented before..

**Figure 10 :Level 1 EPC diagram**

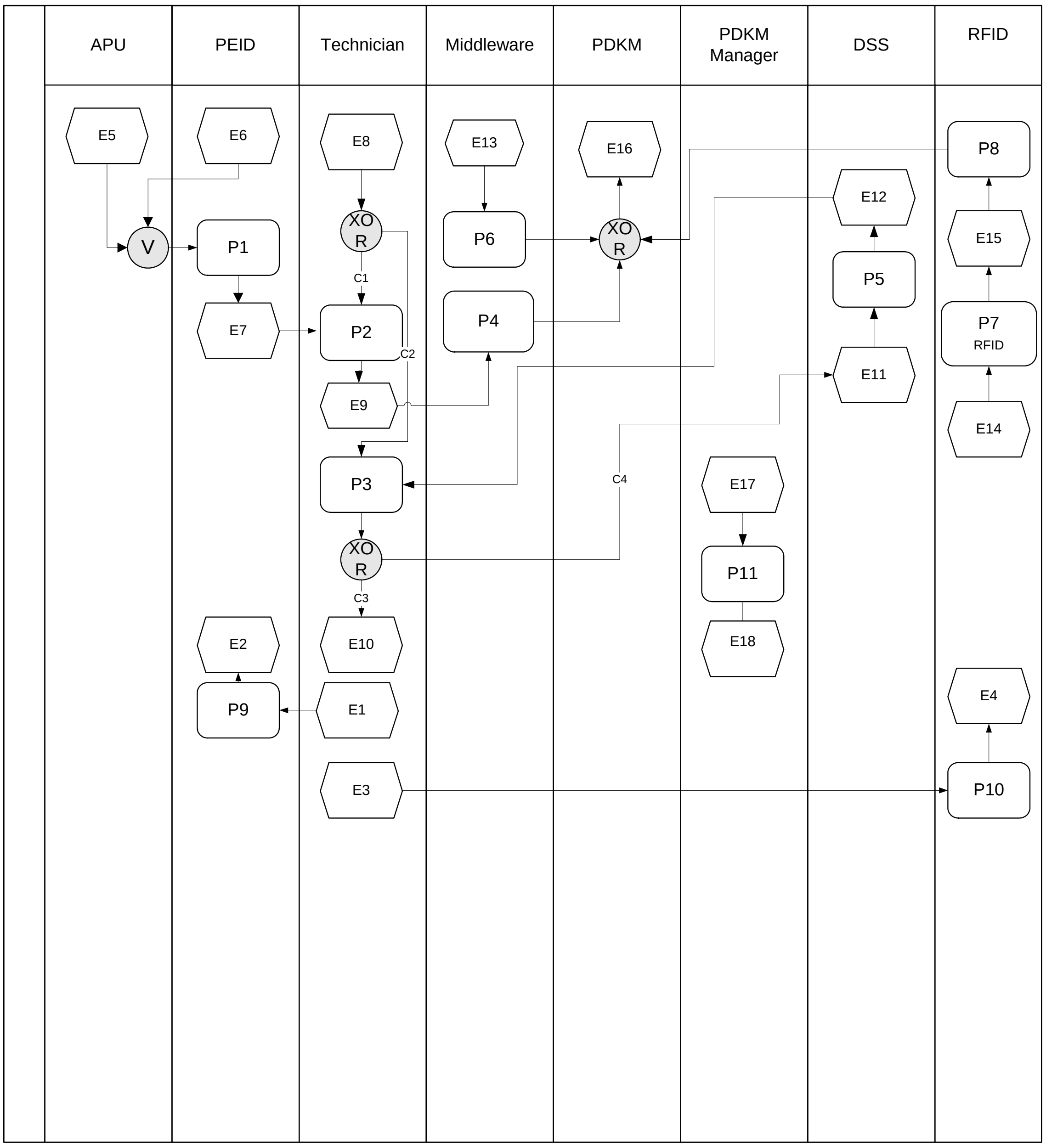

**Table 1: Input-Output & Organizational Information for Figure 10 Processes**

| Process | Input | Output | CHANGES – COMMENTS |
|---|---|---|---|
| P1 | APU Data (I1) | Dynamic PEID Data (I2) | Implemented via 3 components:<br>APU, Protocol Converter, Sindrion PEID |
| P2 | Dynamic PEID Data (I2)<br>Static PEID Data (I8) | Reader Local data (I3) | Implemented through Sindrion PEID, PEID Reader and corresponding application.<br>Middleware NOT involved. |
| P3 | Problem Information (I4), Potential Solutions (I5) | (none) | Implemented in DSS terminal.<br>Problem information entered in DSS GUI.<br>No PEID Reader used, no Middleware used.<br>**Actors**: Technician, DSS, PDKM. |
| P4 | Reader Local Data (I3) | PDKM Data (I13) | Local to PDKM and Middleware infrastructure<br>Implemented through PEID-Reader, Middleware, PDKM |
| P5 | Problem Information (I4), PDKM Data (I13) | DSS Potential Solutions (I5)<br>Case and Solution (I6) | Local to DSS infrastructure.<br>Implemented through DSS<br>This process and I/O data are a complete subcomponent of P3 above. |
| P6 | Backend data (I12) | PDKM Data (I13) | Local to INTRACOM facilities.<br>Implemented through Back-end Proxy Server and associated applications, A9 PMI-XML, PDKM. |
| *P7* | *RIFD tag data (I10)* | *Logistics Data (I11)* | *Local to INTRACOM facilities.*<br>*Implemented through RFID Tags, RFID-Reader (and/or RFID System), RFID Device controller.* |
| *P08* | *Logistics Data (I11)* | *PDKM Data (I13)* | *Local to INTRACOM facilities.*<br>*Implemented through RFID Tags, RFID-Reader (and/or RFID System), RFID Device controller.*<br><br>*P7 and P8 are simplified by restricting the data stored on the RFID tag to a simple ID. No rewriting or exctra information stored on RFID tag and transferred to/from the PDKM.* |
| P09 | PEID Install Info (I7) | Static PEID Data (I8) | Implemented trough PEID, Protocol Converter, PEID Reader and associated application.<br><br>Note that PEID initialization cannot be executed as described due to restricting functionality in MW and PDKM implementation. Instead, the initialization process pre-supposes pre-creation of relevant structures (Measuring Points) in the PDKM and proactive transfer of “back-end” PDKM data to the PEID Reader. |
| P10 | RFID Init Info (I9) | RIFD tag data (I10) | NOT Implemented.<br>Could not be implemented through MW existing MW functionality. |
| P11 | PDKM Data (I13) | DSS Data (I14) | Not implemented in integrated Demonstrator.<br>Restriction was the ability to only define a reasonable subset of possible parameters in the |

|  |  |  |  |
|---|---|---|---|
|  |  |  | PDKM for later searching. Performed offline via searching Backend Proxy data. |

### 2.3.2 Modification of process P01: APU->PEID

Process P01 has been described in the *DA9.3: 2.2.2.3*.
The central processing card that handles all IBAS product configuration and operation procedures interacts with the PEID. When the connection is established data transferred from APU to PEID according to request.
The significant implementation note here is that connection between the APU and Sindrion PEID is implemented through an Enternet-to-Rs232 protocol converter. Furthermore, an A9 specific component was created that converts SNMP-formatted data to a compact encoded format suitable for transfer to the PEID Reader. The process is initiated by the Sindrion PEID which periodically request data from the APU and stores them locally until a PEID Reader requests them.

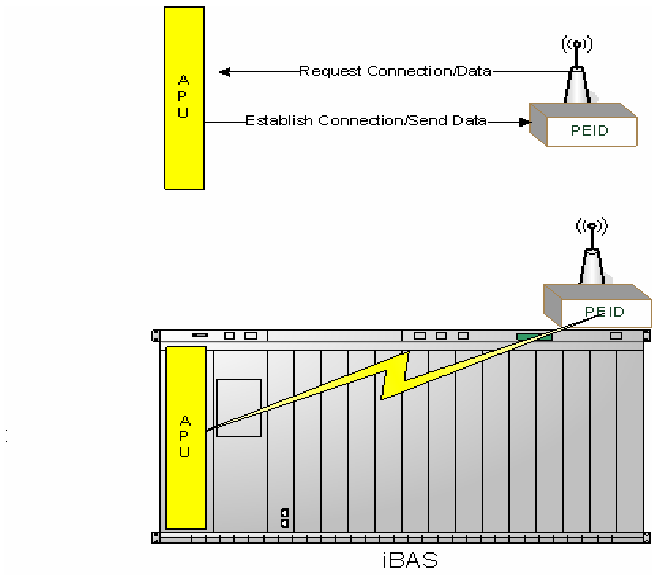

**Figure 11: Communication between APU and PEID**

A brief textual description of salient implementation notes for sub processes is given below (see DA9.3 and DA9.4 for table or diagrammatic format).

#### 2.3.2.1 Sub-process P1: Initiate Communication with APU

Implemented Through Protocol Converter.

#### 2.3.2.2 Sub-process P2: Retrieve Data

Retrieval of APU data by the PEID, initiated by PEID. . Three different types of data (performance, configuration, and fault) stored to PEID component and which are handled differently by the PEID.

MW architecture and current state o f implementation limit the amount of different fault/performance data to 5 per card ( ~ 100 in total per iBAS).

#### 2.3.2.3 Sub-process P3: Process Data

Implemented Through PEID and Protocol Converter..

#### 2.3.2.4 Sub-process P4: Update Configuration Data

Implemented Through special A9 PEID-Reader-based application..

#### 2.3.2.5 Sub-process P5: Append Fault Data

Not Implemented. All data is Updated as in sp-P6.

#### 2.3.2.6 Sub-process P6: Update Performance Data

After the successful completion of performance/fault data processing, data are stored to PEID.

### 2.3.3 Modification of process P02: PEID<-> Reader

Six sub-processes compose this process. P2 is also described in *DA9.3: 2.2.4.3* scene.

The FT uses the PEID reader and connects with the PEID component. The Reader then Collects all the information stored in the PEID and FT inspects it for apparent correctness.

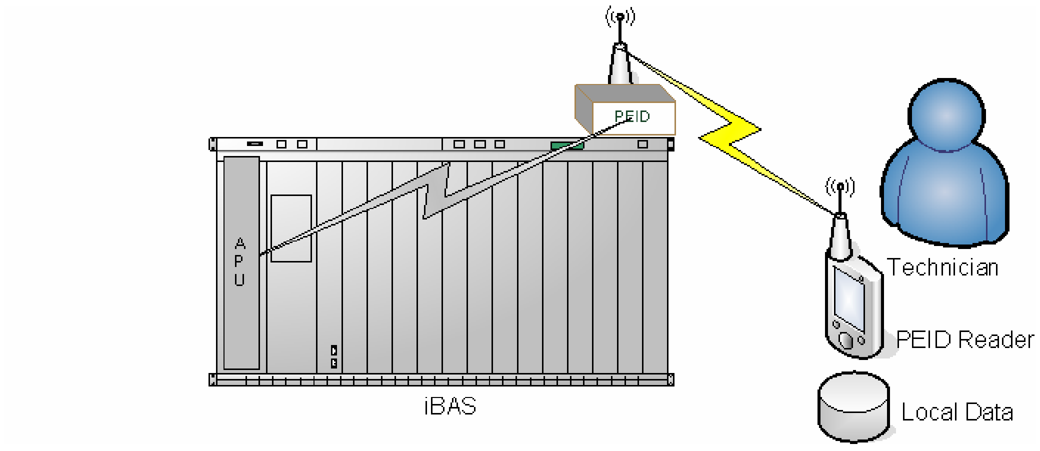

**Figure 12: PEID reader interacts with PEID component**

#### 2.3.3.1 Sub-process P1: Reader seeks PEID

Implemented over a PEID-specific wireless link via Infineon UPnP CorePACK DC. .

#### 2.3.3.2 Sub-process P2: Connect to PEID

Implemented over a PEID-specific wireless link via Infineon UPnP CorePACK DC. .

#### 2.3.3.3 Sub-process P3: Retrieve data

PEID-Reader requests data from PEID, PEID initiates APU interrogation.

Data transformation from PEID-friendly to Middleware/PDKM friendly formats is NOT performed here as was planned but in a NEW sub process P7 added below.

#### 2.3.3.4 Sub-process P4: Store data locally

Data are stored permanently in PEID-reader non-volatile memory.

#### 2.3.3.5 Sub-process P5: Reconfigure PEID

Process indicates mostly new information entered by the user to the PEID, or clearing PEID memory after retrieving data. Initiated in PEID-reader, data finally stored/updated in PEID.

#### 2.3.3.6 Sub-process P6: Update Performance Data Next Action Selection

Still connected to PEID via the PEID-reader, user must choose whether to finish the PEID connection OR continue interacting with the PEID.

#### 2.3.3.7 Sub-process P7: Convert PEID Data to PDKM-friendly format

Conversion of data in PDKM-friendly format (i.e. transformation of encoded format to <targetdevices>, <nodes>, <infoitem> format).

### 2.3.4 Modification of process P03: Problem Solving

Implemented through DSS GUI locally in DSS location. No FT can transfer data to DSS unless he is networked with the DSS and can use DSS GUI. Middleware not used.

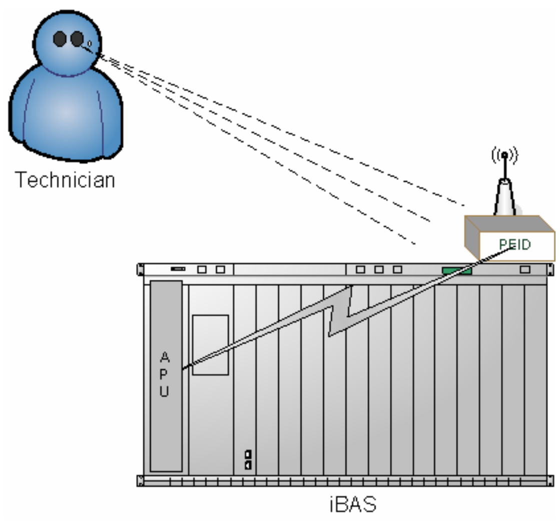

**Figure 13: Technician inspects to identify the problem**

**A**ll Input/Output data relevant to the Promise system in this sub-process are analyzed in Process P05. P5 is a subset of this P03 and describes the interaction of the Technician with the DSS.[2]

Note that Figure 4 representing the physical aspect of this process does not contain ANY interaction with Promise components and only displays inspection (visual in the figure) of the product by the Technician.

#### 2.3.4.1 Sub-process P1: Problem Solving

The Technician after the FT's inspection identified the problem and solved it.

#### 2.3.4.2 Sub-process P2: Additional Problem Investigation

The FT under Technician guidance tried to solve the identified problem but the solution applied didn't work. Consequently the Technician proceeds to additional problem investigation.

### 2.3.5 Modification of process P04: PEID reader->PDKM

Process P04 composed by seven sub-processes. This process was described in *DA9.3: 2.2.2.3.*

PROMISE PEID Reader connects MW to transfer the collected performance/fault data from the Reader to the PDKM.

[2] P3 was instantiated in order to be able to incorporate other possible sub processes in the Problem-Solving workflow if/when needed.

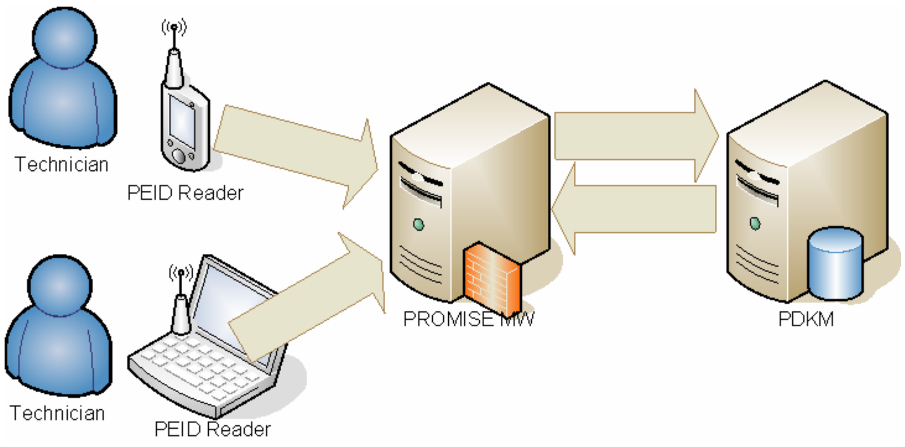

**Figure 14: PEID Reader to PDKM**

#### 2.3.5.1 Sub-process P1: Reader seeks Middleware

Identical to all other "*[Component] Seeks Middleware*" processes in DA9.3-4.

#### 2.3.5.2 Sub-process P2: MW Identifies & Authorizes Reader

Using the appropriate interface, connection between MW and PEID Reader is established and verified by MW.

#### 2.3.5.3 Sub-process P3: Reader Uploads Data

The Reader after connected with the MW proceeds to data uploading. All the information gathered by the PEID Reader transferred to MW.

#### 2.3.5.4 Sub-process P4: MW Verifies & Processes Data

Not specifically implemented in A9. Data integrity is verified at the source (Reader).

#### 2.3.5.5 Sub-process P5: User/MW Corrects Data

See above.

#### 2.3.5.6 Sub-process P6: Transfer Data to PDKM

Subscription functionality used for transfer.

#### 2.3.5.7 Sub-process P7: Next Action Selection

Not specifically implemented. Transfers are performed under user supervision one by one.

### 2.3.6 Modification of process P05: DSS<-> FT

Process P05 composed by seven sub-processes. This process was described in *DA9.3: 2.2.5.3.*
It describes the procedure followed when technicians seeking assistance to solve a problem by using the DSS.
In that case they have to describe the problem at hand as accurately as possible and in as many "dimensions" as possible. The description is by entering in DSS GUI all relevant data as captured from the product's PEID via the PEID reader.
The DSS provides suggested solutions to the described problem. The DSS User can then assess the validity/applicability of the DSS-Suggested solution and enter the assessment information in the PDKM[3].

[3] This was planned to be performed in the DSS but it is implemented in PDKM.

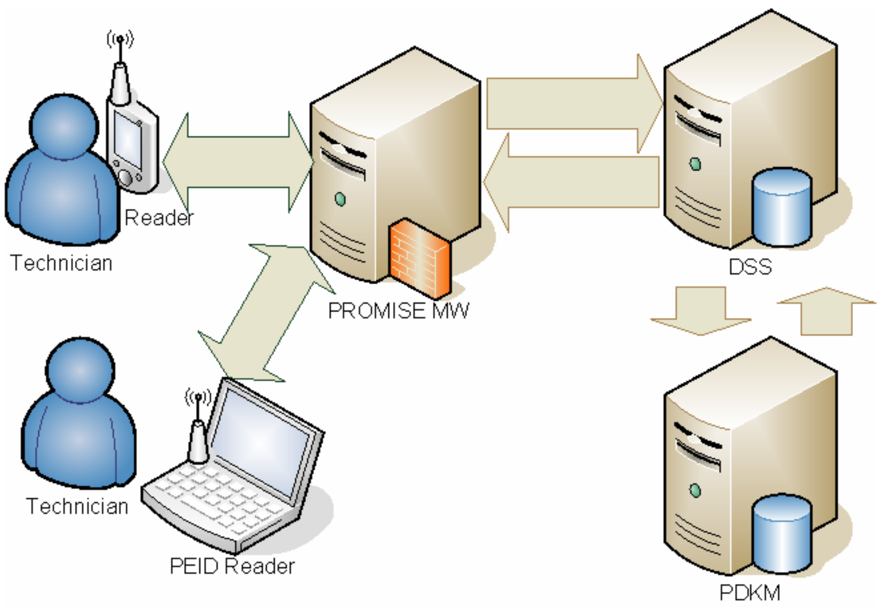

**Figure 15: DSS user seeking assistance**

#### 2.3.6.1 Sub-process P1: Describe problem

FT describes the problem at hand as accurately as possible and in as many “dimensions” as possible in order to find a similar solution. Technician enters data in DSS GUI.

#### 2.3.6.2 Sub-process P2: Search for Similar Cases

DSS searches for a similar solution.

#### 2.3.6.3 Sub-process P3: Describe Case (Problem Description)

Not exclusively implemented. Equivalent to entering search data in Different dimensions.

#### 2.3.6.4 Sub-process P4: Identify Potential Solution

Presentation of list of potential solutions.

#### 2.3.6.5 Sub-process P5: Solution Assessment

The technician selects the potential solution and tests for its success. If the solution is successful then he reports the solution to PDKM[3].

#### 2.3.6.6 Sub-process P6: Manual Solution Searching

Not exclusively implemented. Equivalent to entering search data in less detailed dimensions..

#### 2.3.6.7 Sub-process P7: Report Case OR Solution

Reporting Solution or Case is performed by appropriate entry in the PDKM[3].

### 2.3.7 Modification of process P06: Back End -> PDKM

Process P06 composed by five sub-processes. This process was described in *DA9.3: 2.2.3.3.* Company’s backend systems considered here are the *SIS* system used by the Repair laboratory Personnel and the *ADARES* system used by the Call Center personnel. Data from BE systems are transferred on user action (batch input) to Backend System Proxy.

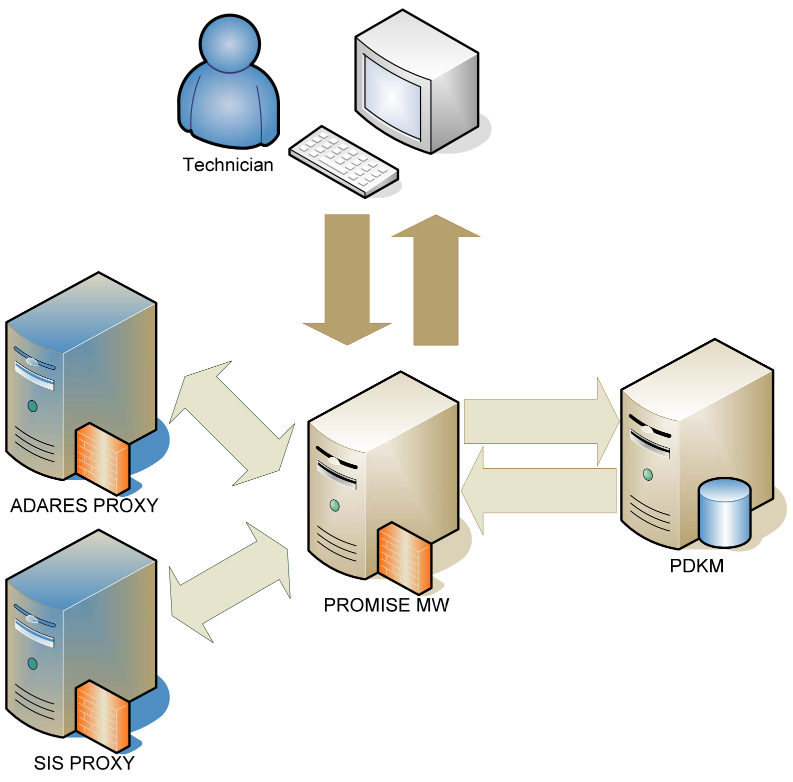

**Figure 16: Backend data transferred to PDKM**

ADARES Proxy and SIS Proxy are combined in one physical system.

#### 2.3.7.1 Sub-process P1: Backend Proxy seeks Middleware

NOT Implemented. Backend Proxy data are not transferable via Middleware.
Instead, the Backend Proxy Server directly connects to the PDKM.

#### 2.3.7.2 Sub-process P2: MW Identifies & Authorizes BE Proxy

See above. ID and authorization via PDKM facility.

#### 2.3.7.3 Sub-process P3: BE Proxy Uploads Data

Directly to the PDKM instead on going through the MW.

#### 2.3.7.4 Sub-process P4: MW Verifies & Processes Data

Not applicable. Data integrity is verified at their source and via internal PDKM mechanisms..

#### 2.3.7.5 Sub-process P5: Transfer Data to PDKM

Uploading of PMI-XML data.

### 2.3.8 Modification of process P07: RFID & associated data entry

Process P07 composed by five sub-processes. This process was described in *DA9.3:*

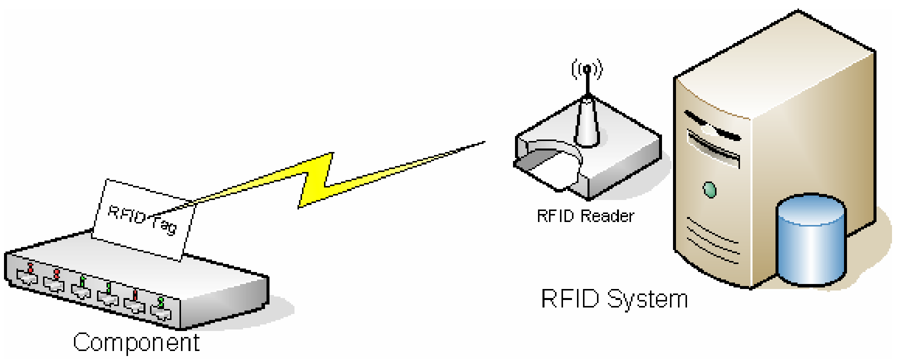

**Figure 17: RFID Systems**

This process has been very simplified during the implementation. Due to possible inconsistencies with data stored in the PDKM about the component, the plan to include comprehensive data in the RFID tag attached to the component was abandoned and only the ComponentInstanceID (EquipmentID in PDKM terms) is stored on the tag. The only data added are Comments (textual format) by the RFID-Reader operating technician, if appropriate.

### 2.3.9 Modification of process P08: RFID-> PDKM

Process P08 composed by five sub-processes. This process was described in *DA9.3:*

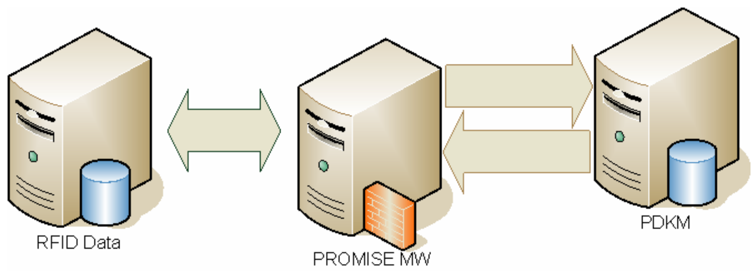

**Figure 18: RFID to PDKM**

#### 2.3.9.1 Sub-process P1: RFID Reader seeks Middleware

Implemented via Indyon RFID DC.

#### 2.3.9.2 Sub-process P2: MW Identifies & Authorizes Reader

Implemented via Indyon RFID DC.

#### 2.3.9.3 Sub-process P3: Reader Uploads Data

Implemented via Indyon RFID DC.

#### 2.3.9.4 Sub-process P4: MW Verifies & Processes Data

Not implemented in current version of MW. Data integrity verified at the source. In case a component that is unregistered in the PDKM is scanned, the PDKM will reject entry of this item.

#### 2.3.9.5 Sub-process P5: Transfer Data to PDKM

Implemented via Indyon RFID DC.

### 2.3.10 Modification of process P09: PEID Init

Process P09 composed by five sub-processes. This process was described in *DA9.3:*

Implemented via Infineon-supplied CorePACK UPnP DC and custom code.

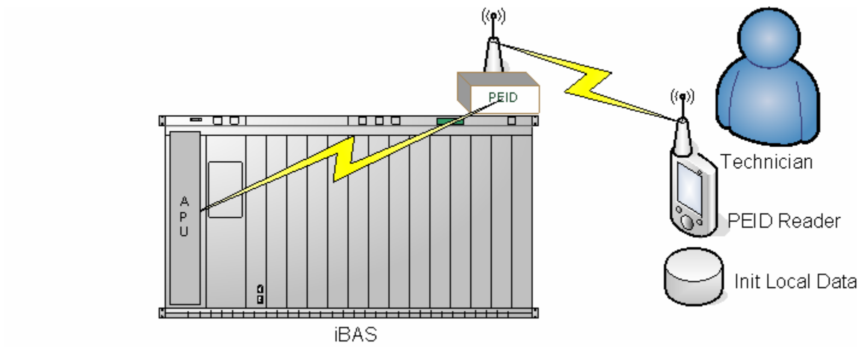

Figure 19: PEID Initalization

#### 2.3.10.1 Sub-process P1: Reader seeks PEID

Implemented via Infineon-supplied CorePACK UPnP DC.

#### 2.3.10.2 Sub-process P2: Connect to PEID

Implemented via Infineon-supplied CorePACK UPnP DC.

#### 2.3.10.3 Sub-process P3: PEID configuration

Implemented via Infineon-supplied CorePACK UPnP DC and Custom reader-based application.

#### 2.3.10.4 Sub-process P4: PEID operation verification

Implemented via Infineon-supplied CorePACK UPnP DC and Custom reader-based application.

#### 2.3.10.5 Sub-process P5: Next Action Selection

Not exclusively implemented. Application can perform either of the tasks until it is closed.

### 2.3.11 Modification of process P10: RFID Init

NOT exclusively implemented. All RFID tags are pre-written and attached to the component used for the demonstrator.

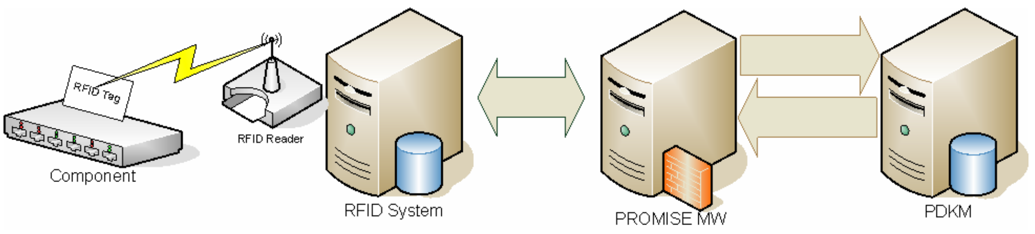

Figure 20: RFID Initialization

### 2.3.12 Modification of process P11: DSS<-> PDKM Mgr

Not implemented in integrated Demonstrator. The major restriction was the ability to only define a reasonable subset of possible parameters in the PDKM for later searching. Fault and performance data are the only data that can be transferred via MW to the PDKM and the actual implementation needs manual definition of every singly measuring point (data point) for every single component. Furthermore, possible changes to existing data or measuring points are “restricted” by the underlying SAP-PLM.

Process was performed offline via searching Backend Proxy data. A description of the procedures that were followed is given below:

### 2.3.12.1 Sub-process P1: Select Sample of PDKM Data for DSS Parameter Tuning

This was not feasible since all data needed processing before they were stored in the Backend Proxy or the PDKM. Data clearing led us to only upload data for which there was complete and useable information so the selection of a subset had no meaning in this context.

### 2.3.12.2 Sub-process P2: Select Factors/parameters for Evaluation

All available parameters were tested on Backend Proxy using the EPFL algorithms. Field data was selected in “quintets” since this was the number of measuring points per component implemented.

### 2.3.12.3 Sub-process P3: Factor Impact Analysis

See above. Running the algorithms resulted in the weights eventually used. Weights were entered in the DSS data and code.

### 2.3.12.4 Sub-process P4: Pareto Analysis

Performed via initial weight assignment to factors by experienced field and support technicians.

### 2.3.12.5 Sub-process P5: Update DSS with New information

Not applicable.

# 3 Analysis of results obtained in the Activities A9.6.y

## 3.1 Test portfolio of stand-alone RFID solutions in machine environment

Tagging solutions tested in the PROMISE A9 Demonstrator are the following:

1. *Solution 1:* Passive HF RFID tags and readers (supplier: TAGSYS) attached to components for logistics and maintenance information tracking
2. *Solution 2:* Active Sindrion PEID (supplier: Infineon) and PEID-Reader attached to the iBAS product and connected to the iBAS APU via Ethernet-to-RS232 converter.

RFID solutions in the framework of the A9 Demonstrator are used for 2 distinct purposes, one per solution as described above:

**1.** Track location and operations performed by human operators on components that needed some attention (repair, inspection, renewal, etc.)
**2.** Monitor the performance, fault and operational conditions of iBAS products while they are in operation and store this information for retrieval when needed by a human operator so that a complete picture of the MOL face of the product can be formed in thePDKM.

The potential problem for Solution 1 was the ability of the reader and selected tag to be read accurately in its intended environment.

Potential problems for Solution 2 were the ability of the PEID to communicate with the APU, retrieve accurate data and store them for later retrieval, and all this in an environment where extreme RF interference and obscene environment conditions might exist.

### 3.1.1 Passive RFID tags in actual environment

RFID is an automatic identification technology with digital data encoding in a suitable tag and scanning of the tag by a reader using radio waves. RFID systems can work at various radio frequencies, such as:

- Low Frequency (LF): 125/134 kHz
- High Frequency (HF): 13.56 MHz
- Higher-other-Active tags

Although active RFID tags were also tested during the project, their size, attachment options and cost made them unsuitable for the intended purposes of A9. Furthermore, the A9 RFID usage scenario only needs scanning in controlled environments and via an operator that knows what she is doing, hence readability requirements were reduced

Taking under consideration needs and potential cost, several different types of passive RFID tags by TAGSYS were tested and we used the smallest and cheapest that proved suitable.

Note however that ALL passive RFID tested, failed to work at specific positions on the component (card) used! "*Failed*" as in returning no response at all every single time they were scanned from any distance. There are only two locations in the majority of the card where they are operational, and only ONE location on the card in some of the more densely populated cards.

*Position is then critical for operation of the tag on the specific components A9 is using. In the case of newer planned components that will be even more densely populated, the only suitable position may be at the Back of the component, outside of the actual product.*

*As they were, RFID tags and readers used, proved readable in the tested conditions from a distance of 20cm and the operator holding/sweeping the component above the reader.*

### 3.1.2 PEID in Actual Environment

The major consideration for the Sindrion PEID was that it needed to accurately communicate with both the APU and the PEID Reader in an environment buzzing with Radio Frequency interference if the worst kind!
IT should further be able to accurately without errors transfer all data to and from the PEID Reader in sessions of interaction with the reader.
Communication with the APU is rather robust since it was implemented via a wired link in the Demonstrator (and is designed to be implemented on board the product when later commercial Sindrion generations come in a SMD form factor. The current implementation uses the existing APU Ethernet port and connects on it an Ethernet-to-RS232 protocol converter. The converter is then connected on the Sindrion Serial bus. Although both products operate exposed to the RF interference, there were no errors from this source during testing.
The wireless link to the Reader is implemented via an Infineon designed wireless HW and protocols and is suitable for the task.
Finally, the need to store all data in persistent storage and the need to verify the accuracy of the data on the destination, necessitate the development of a robust data/file transfer protocol over the serial RS232 link. The present implementation is based on a very simple protocol and has proven reliable but there were cases where the data was corrupted and further processing failed downstream of the PEID. Although this was simply alleviated by a simple re-reading of the Sindrion, there is always the possibility that data corruption may occur to a degree that makes it undetectable.

Concluding, the current implementation has proven that data can be transferred from the APU and stored successfully on the PEID until it is read by a PEID-Reader.

## 3.2 Back-end Systems and Data transfer to the PDKM.

The A9 PDKM developed by InMediasP proved capable of representing the A9 logical structures within the confines of a commercial robust PLM system. There were several cases where detours had to be taken in order to utilize the system with some efficiency but they did not detract or hindered the Demonstrator.

## 3.3 Development and test of the stand-alone DSS (v2)

The development and tests of the stand-alone DSS were presented in DR8.8 and DR8.9. The integrated DSS is presented in DR8.11. The reader should refer to these documents for any detail on the DSS.
We should mention here that the nature of the DSS Similarity Searching algorithms and the underlying data, guarantees that the most similar cases will always be presented to the user.

Testing however proved inconclusive regarding the extent and depth of the criteria that need to be used in order to get an accurate and useable solution suggestion. The existing data do provide such solution suggestions but they are cleaned and processed before entered in the PDKM so that they actually represent cases where complete info was available.
The DSS failed in treating cases with incomplete information, which are cases where data where not entered in the format designed during Promise but were entered "to the closest approximation according to our understanding". Contrary to "complete cases", in such cases, the selection of multiple criteria for similarity matching led to unusable suggestions or no suggestions at all, i.e. the DSS (correctly) found no "similar cases"! Relaxation of the search criteria makes the outcome contaminated by results from "somewhat similar cases" reducing the usability of the DSS.

It is obvious that this issue can only be alleviated by ensuring that data fed to the DSS are all complete in terms of search parameters, something that will need modification of the processes and sources of the primary data (ADARES data and processes).

## 3.4 Integration tests

Integration of the following components (in bold the components specifically developed in PROMISE) has been performed:

- **Passive HF RFID tags** attached to components. and read by **RFID readers**
    - **RFID device controller** provided by INDYON with Software components written in C# (dotNET) and Persistent data and inter-process communication between the 2 components is realized with a **SQLite database**.
- **Sindrion PEID** attached to iBAS product provided and developed by Infineon.
    - - **CorePAC UPnP device controller** interacting to **SAP Middleware** for transfer of performance and fault data collected from iBAS, **Ethernet-to-RS232 Protocol Converter** (Axis product) for connecting APU to Sindrion, **APU Data Transformation component** for interrogating the APu and converting SNMP responses to coded format suitable for handling by Sindrion.
    - **PEID Reader application** for interaction with PEID data and later transfer to the PDKM
- **Back-end Proxy Server** and required applications that act as intermediaries between actual back-end data and the PDKM. Storage implemented in MS SQLServer 2005 and applications in C# (.NET)
- A9-specific **DSS algorithms for similarity searching**,
- A9-designed **PMI-XML** developed by **INTRACOM** for backend data transfers to PDKM
- **PDKM**, developed by InmediasP on SAP-PLM infrastructure and hosted by SAP
- **DSS** integrates the EPFL algorithms

# 4 Conclusion

This deliverable documents the final implementation of the A9 Demonstrator and this section documents in a concise way the results obtained.

A major goal for A9 was to include most of the Promise technologies so that their applicability to MOL maintenance tasks can be investigated and demonstrated. This obviously created a multitude of issues that needed to be addressed, stemming from the variety of interfaces that needed to be implemented and tested, as well as the variety of data sources and data types that needed to be integrated.
A second major goal was to investigate the efficacy of a DSS solution using the data collected from Promise technologies to assist in maintenance tasks by suggesting appropriate potential solutions to new problems.
Both goals have been achieved to a satisfactory level (see Results in previous section) and the main technical problems have been either solved or identified as needs for future work in order not to become barriers.

## 4.1 Results

A9 successfully integrated within the limitations of an R&D project the selected technologies.
iBAS APU was integrated via the SINDRION PEID to the Demonstrator and performance/fault data during gits operation can be retrieved from the PEID regardless of the state of the product itself.
Component maintenance reporting via RFID tags was successfully integrated with the Demonstrator.
Backend system data were integrated via the Backend Proxy Server.
Finally, all varying data in the PDKM proved to be usable from the DSS and capable of providing successful suggestions when the problem was described correctly.

## 4.2 Technical Problems and Technology Barriers

The main issues that remain as barriers for the future coincide almost perfectly with technical problems that arose during the Promise efforts for A9, even those that were resolved during the project.

1. **New Products and technologies**
    - Sindrion PEID is a central part of the A9 architecture and its development and commercialization timeline forms a critical path of implementation and success for the product-end of commercial A9 Application. The existence or adaptation of standard development tools and protocols on the Sindrion platform can go a long way in paving the path for A9-like applications, but is not guaranteed if SINDRION finds a lucrative niche market (e.g. Automotive) and is not further developed as a multi-application product.
    - Promise MW worked in the framework of Promise but there are extensive and important enhancements that are needed in order to be useful as a component in commercial development projects and products. Although it's Specification and design is clean and supports those needs, the current implementation lacks functionality in important areas

(e.g. data not considered *“field-data”* cannot utilize the current MW) and usability features.

**2. Demonstrator-specific products and technologies**

- The iBAS product has limited horizon of production as all telecom products these days. It has already passed the maturity phase (its middle of life!) and its original design poses limitations that make Promise integration for this particular product rather cumbersome. The current generation has limited free resources in terms of memory and APU/PPU power since the extensions of the product have carried the design beyond its original limits. Implementation of the APU-to-PEID code within the APU is not a task to be taken lightly, especially when considering the computing needs and QoS requirements of a real-time embedded system. Future generations of similar products will not suffer from such limitations but this issue makes critical the availability of new products and technologies within a reasonable timeframe.
- The utilization of SAP-PLM as the basis for the PDKM, created an array of issues and delays during the project. Application and technology partners were trying to perform R&D tasks that require multiple changes and testing when so indicated, using a commercial product that earned its reputation by being specifically designed to discourage such “behaviour”. The lack of expertise and lack of appropriate access of most application partners to perform themselves such required interventions, resulted in overloading the technology partners that were responsible for the development. Sometimes, it led to the (always rational) decision to implement a common solution instead of a demonstrator-specific solution, lest the integration would take forever.
- The previous issue will carry over to the future for potential commercialization of Promise demonstrators. The procurement, configuration, development and support of a similar system in order to exploit work performed in Promise are by no means a tractable undertaking.

## 5 References

DA9.3: Design of the MOL demonstrator for Telecommunication equipment
DA9.4: Process model workflow description for the demonstrator
…