Enabling the business-based
Internet of Things and Services

(FP7 257852)

D8.6 Integration of Physical World in Traceability Scenario

Published by the ebbits Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme
Objective ICT-2009.1.3: Internet of Things and Enterprise environments

Document control page

Document file: D8.6 Integration of Physical World in Traceability Scenario.docx
Document version: 1.0
Document owner: Michael Jacobsen (TNM)

Work package: WP8 – Physical World Sensors and Networks
Task: T8.3 – Network implementation and interfaces in traceability
Deliverable type: Report

Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Michael Jacobsen (TNM)	2011-12-16	Initial ToC
0.2	Michael Jacobsen (TNM)	2012-03-20	Updated ToC
0.3	Mauricio Caceres Duran, Gonzalo Alcaraz (ISMB)	2012-04-23	Contribution to chapter 5
0.4	Michael Jacobsen, Sigurjón Björnson (TNM)	2012-04-23	Feeding computer API descriptions added.
1.0	Michael Jacobsen, Sigurjón Björnson (TNM)	2012-04-30	Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Peeter Kool (CNET)	2010-04-26	Minor comments
Ferry Pramudianto (FIT)	2010-04-26	Minor comments

Legal Notice

The information in this document is subject to change without notice.

The Members of the ebbits Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the ebbits Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Possible inaccuracies of information are under the responsibility of the project. This report reflects solely the views of its authors. The European Commission is not liable for any use that may be made of the information contained therein.

Index:

1. Executive summary

In this deliverable we describe how to integrate a number of systems, relevant to the traceability scenario, into ebbits with the help of the Physical World Adaptation Layer.

One area of focus is the integration with various systems at the farm. As has been detailed before, a typical farmer uses many controllers and automation system to automate the operation. The feeding system control the feed stuff supplied to the animals and is therefore of importance to traceability.

There are no leading standards for data exchange for control systems on farms so usually every manufacturer has their own protocol for communication. Data exchange protocols for three feed controllers from two manufacturers are described.

- Skiold DM-5000 uses VIGO system for data exchange (OLE Automation COM-object)
- Skiold DM-6000 uses file based data exchange (comma separated values)
- Big Dutchman NT-99 uses NETIPPC interface

Two standardization projects are discussed, ISOagriNET and Datastandard. The Datastandard API is described as an example of how standardized interfaces work. These standards are not widely implemented.

Chapter 4 describes RFID tags and readers. RFID tags are being applied in ever more applications and are being used for traceability in many domains. RFIDs are standardized by GS1 organization and they provide the EPCglobal Architecture Framework. This framework standardizes the components and interfaces in systems using RFIDs in order to use them globally and across enterprises.

For implementing RFID readers and tags in ebbits the PWAL/ebbits core needs to support the Low-Level Reader Protocol, Reader Management and Discovery Configuration. Chapter 4.1 describes these protocols and their responsibilities.

EPCglobal certified tags come in 4 classes, from basic Passive-backscatter tags to Active tags with integrated microcontrollers and sensors. A range of RFIDs especially practical for traceability are tags with integrated temperature sensors and memory for data logging. These sensors could then verify that the cold-chain is unbroken by the consumer in the shop.

2. Introduction

2.1 Purpose, context and scope of this deliverable

This deliverable describes an initial set of interfaces and solutions used to integrate the physical world in the traceability scenario.

A number of devices are selected for this first part of the iterative work. The work will be extended in the following deliverables *D8.7.1 Integration of new sensors in traceability scenario 1* and *D8.7.2 Integration of new sensors in traceability scenario 2*.

It should be mentioned that most data of relevance to traceability in the strict sense is found in databases as the information is historic by definition. However, interfacing to these databases is more suited for work package 6 "Mainstream Business Systems" and in particular the deliverables *D6.7 Analysis and design of services to interconnect with centralized enterprise information systems* and the iterative deliverables *D6.8.1 Implementation of integration services in the ebbits platform 1* and *D6.8.2 Implementation of integration services in the ebbits platform 2*.

2.2 Background

In the deliverable *D8.2 A Survey of the Physical World in manufacturing and traceability* the scenarios where described in terms of the available sensors and actuators.

This deliverable goes into detail on how to integrate a number of these systems into the ebbits framework. In particular, it describes the integration of these devices with the "Physical World Adaptation Layer" (PWAL), described in deliverable *D8.3 Physical World Adaptation Layer*.

2.3 Structure of deliverable

The next sections describe the selected devices, sensors and subsystems. In section 3 three different feeding systems are described. Section 4 describes how ebbits may interface to RFID tags not only at the farm but throughout the traceability chain which in the future may include the smart home.

3. Process Computers at the Farm

Next to the historic information in the management system, the most valuable source of information and control on a farm is the feeding system. The feeding of the animals is arguably the most expensive aspect of animal farming. At the same time proper feeding is essential to obtain animals with the most value. Thus, the feeding system is a point where feedback and optimization may provide the highest benefits.

In the field of feed systems, several vendors and systems exist and it is not feasible to interface to them all or to explain about them all as no common interface is implemented. Thus, a number of representative systems are selected to illustrate the possible interfaces that can be met.

As described in the next sections all the process computers are connected to and controlled by a PC. It is safe to assume that the PC is already connected to some kind of standard local area network. Thus, electrical standards, interference, EMC pollution and cable characteristics are out of scope for ebbits.

For each API a simple example of extracting and updating information is given in order to give a sense of the steps needed. In all cases we look at extracting information regarding the feeding setup for a valve. The examples use a shortened form of C#.

The degree of control provided by the feeding controllers varies by type and installation. However, the basic operation is to open and close valves and thereby control the amount of feeding stuff provided to each pen, see also Figure 1. Hereafter we will use the terms valve and pen for the single unit of control.

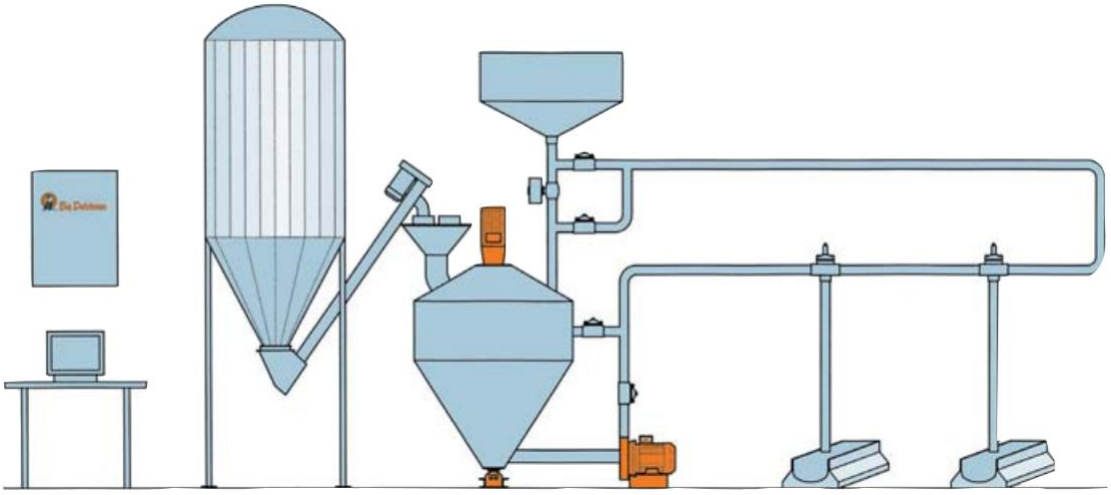

Figure 1 A schematic for a valve feeding system with (from left to right) a controller attached to a computer, feed silo, mixer and two valves. The system is controlled by the controller. Feed is taken from the silo, mixed with water and pumped to the feeding trays to the right according to the setup for the individual valve. Each feeding tray may serve more than one pen/sty. The illustration is from Big Dutchman[1] but the schematic is the same for the other systems.

3.1 Skiold DM-5000

The Skiold DM-5000 system is based on process computers and I/O modules from Proces-Data A/S all interconnected through P-NET (International Standard Fieldbus IEC 61158 Type 4), see Figure 2. The particular controller used for Skiold DM-5000 has output for a standard VGA monitor and input for mouse and keyboard. It is thus possible to control the process computer without an external computer.

[1] http://www.bigdutchmanusa.com/resources/Swine-Downloads/Feeding/WetMIX-gb-small.pdf

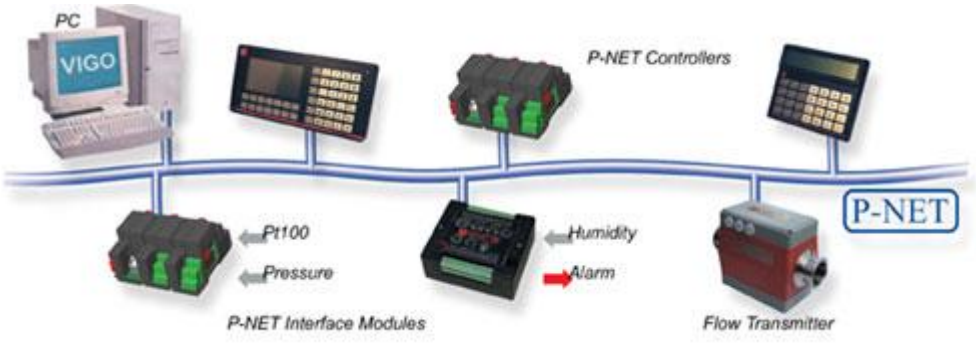

Figure 2 The P-NET Fieldbus system various elements attached (http://www.proces-data.dk).

However, in order to access data in the controller across the P-NET the so-called VIGO system is needed. VIGO communicates to P-NET through either an Ethernet or parallel port bridge. VIGO is implemented as an OLE Automation COM-object[2] that allows for easy communication with the process computers and devices on the P-NET Fieldbus. The VIGO system is essentially equivalent to OPC[3] which also is a COM object model to access data on in this case PLCs.

The layout of a particular control system is described in a Manager Information Base (MIB) as a tree of variables. Each node of the tree, including the variables, has a "physical id" attached to it. These physical ids are needed to access and modify data.

Table 1 shows a list of selected physical ids from a DM-5000 system. All entries are given for valve 1.

Physical Id	**Description**
Dm5020:dm2.Sti1[1].StiAStk	Number of animals at valve 1
Dm5020:dm2.Sti1[1].StiAVægt	Weight of animals at valve 1
Dm5020:dm2.Sti1[1].KurveGrise	Feed curve for animals at valve 1
Dm5020:dm2.Sti2[1].SygePct	Temporary deviation from feed curve in order to account for sickness.
Dm5020:dm2.Sti3[1].Foderdage	Days at valve – index into the feed curve table. Increased automatically when new day begins
Dm5020:dm2.Sti4[1].BeregnetFE	Computed amount of feed to be fed today
Dm5020:dm2.Sti4[1].UdfodretFE	Feed fed today (until now)

Table 1 Selection of physical ids available in Skiold DM-5000. The number within the [] designates the valve number. Here it is valve 1.

To access for example the number of animals at valve 1 the following steps needs to be taken.

1. Create the COM object

   ```
   vigo = new VigoStdClass();
   ```

2. Set the "physical identifier" property of the object to "dm5020:dm2.Sti1[1].StiAStk"

   ```
   vigo.PhysId = "dm5020:dm2.Sti5[1].StiAStk";
   ```

3. Read the value from the value property of the COM object. It necessary to know the type of the requested value and cast appropriately

   ```
   value = vigo.Value;
   ```

[2] COM (common object model) is a binary-interface standard defined by Microsoft in 1993. OLE is short for Object Linking and Embedding.
[3] OPC is short for OLE for Process Control.

In order to modify a parameter the order of the assignment in step 3 is simply reversed. That is, `vigo.Value = value`

Because the VIGO COM-object is of the OLE Automation type it must be created and accessed on the main thread of a program in a logged in session.

Thus, in order to integrate a Skiold DM-5000 system with ebbits and the PWAL it is necessary to make a module similar to the OPC module (described in "D8.4 Integration of Physical World in manufacturing scenario") that interfaces through VIGO instead of OPC.

The VIGO interface does not provide a subscription interface and it is therefore necessary for the PWAL to periodically extract values and detect changed values. This functionality is already included into the current PWAL. The interface to the VIGO system must be implemented as a PWAL driver. Additionally it is necessary to develop a setting file containing which parameters and variables (i.e. physical identifiers) that should be exposed through the PWAL.

3.2 Skiold DM-6000

The coming feed computer system from Skiold is denoted DM-6000 and is based on a proprietary and in-house design. The process computer is attached to a computer via a serial port connection. On the computer it is necessary to install the feed management system program which handles the communication with the process computer itself.

The management system provides a file based interface. In order to retrieve data from the system it is necessary to write a file to a specific place. That signals the system to query the actual controller for data and write them to a number of comma separated files.

Steps for exporting data from the DM-6000 computer

1. Create an empty file named "wakeup.dat" in "c:\DM6000\DataTransfer" folder. This starts the data export/import procedure (Optionally another empty file named "totals.dat" can be created if consumption data is needed in addition to configuration data)
2. The export process takes a while to gather all the data and writes it to various comma separated files in the same location. These files are
    a. Componants.dat: Feed components information with nutrition data.
    b. CurvePig.dat: List of feeding curve* names for piglets and slaughter pigs.
    c. CurveSow.dat: List of feeding curve* names for breeding sows.
    d. Nutrientname.dat: Names of nutrients.
    e. Rec.dat: Recipe names.
    f. Sti.dat: Configuration for every valve.
    g. totals_Acum.dat: Consumption of every feed component in every valve (only if "totals.dat" file was created)

* Feeding Curves indicate the amount of feed the controller should provide based on the average animal weight in the pen(s) served by the valve. The weight is updated each day according to a growth model such that the amount of feed is automatically adjusted as the animal grows.

The most important files are "sti.dat" and "totals_acum.dat". "sti.dat" file contains a line for every valve with the relevant parameters. The textbox below shows example from 2 valves and Table 2 explains the most important fields in the outputted text.

402;0;K;80;100;0;0;100.3;304;0;1;1;0;2;0;0;0;2.56;7.707;0;0;0;0;108.21;05/07/11;1;6;324;
403;0;K;100;100;0;0;100.3;306;0;12;1;0;2;0;0;0;3.2;9.637;0;0;0;0;141.25;05/07/11;1;6;318;

No.	Description	Access
1	Valve number	Read only
2	External control allowed	Read/Write
3	Ordering number	Read/Write
4	Feed amount modulation (deviation from feeding curve)	Read/Write
5	Temporary feed amount modulation (deviation from feeding curve)	Read/Write
6	Number of days for temporary feed modulation	Read/Write
7	Stage in cycle (Sows)	Read/Write
8	Weight (Pigs)	Read/Write
9	Age (Pigs) / Days from start of cycle (Sows)	Read/Write
10	Number of sows	Read/Write
11	Number of pigs/piglets	Read/Write
12	Feed curve	Read/Write
13	Technical group	Read/Write
14	Main recipe number	Read/Write
15	Auxiliary recipe number	Read only
16	Portion of Auxiliary recipe	Read only
17	--	Read only
18	Daily feed energy (Unit is Fe/Kg where 1 FE/KG is the calories in 1 kg of barley)	Read only
19	Daily feed amount	Read only

Table 2 Description of data exported from DM6000

Similarly it is possible to write data to the feeding system. To do this it is necessary to write a properly formatted file named "NewSti.dat". The data should be in the same format as the exported data but prefixed with 12 binary numbers indicating which field should be updated. The textbox below shows how to change the number of pigs at valve 402 to 10. This new information is then imported when "wakeup.dat" file is created.

```
000000000100;402;;;;;;;;;;10;;;
```

Because the interface is file based and has no other mean of notifying of changes the PWAL interface needs to use its built in services with respect to generating events from polled data. That is, in order to interface ebbits to Skiold DM-6000 is a PWAL driver and a setting file containing what variables should be exposed.

3.3 Big Dutchman NT-99

The Big Dutchman NT-99 system is a proprietary system built completely in-house by Big Dutchman and it is controlled via a PC. It connects via a serial or USB interface to a PC where the internal protocols are handled by a "NETIPPC" service that allows other PCs to access the NT-99 over the network. Big Dutchman provides a programming interface described in the following that allows reading and writing values to the system via the NETIPPC interface.

The Big Dutchman API is built around the notion of groups which can be compared to sheets in a spread sheet. A group has a number of rows and fields (comparable to columns).

The API is developed in .NET. In order to connect to a NT-99 system it is necessary to create

1. A client connection for which the IP and port is needed. Here the local machine and the standard port have been specified.

```
var clientConnection = new IPCSystem.ClientConnection();
clientConnection.Connect("127.0.0.1", 5103, "ADMIN", true);
clientConnection.WaitForDevices(1000);
```

2. An IPCCClient connection through the clientConnection

```
ipccClient = new IPCCClient(clientConnection);
```

3. A "Visual Farm" connection (named after the control software) using the IPCCClient

```
vfConnection = new VFConnection(ipccClient, "ADMIN", "ADMIN","fbase");
```

In all statements the default passwords have been used. The VFConnection object is needed to read and write information on the NT-99 as seen in the following.

Each NT-99 is able to run several feed programs called devices. One device could handle feed mixing, another valve feeding and a third electronic sow feeding. Assume that device 1 is the valve feeding device/program. In order to read information from a NT-99 a vfConnection is needed

1. The proper device must be selected

```
vfConnection.SelectDevice( 1 );
```

2. The group (type of data) and a number of fields must be selected before creating a query object and data adaptor object. The identifiers for groups and fields corresponds to those of Table 3.

```
group = vfConnection.GetDialogGroupById(1);
fields = new [] {
                group.GetFieldById(4),
                group.GetFieldById(6),
                group.GetFieldById(9),
                group.GetFieldById(10),
                group.GetFieldById(12)
};
query = new VFSelectQuery(group, fields, 0, maxRows);
adaptor = new CFDataAdapter(query, vfConnection);
```

3. Finally, the data can be extracted into a ADO.NET dataset

```
dataset = new DataSet();
adaptor.Fill(dataset)
```

To control the feeding system it is necessary to write to fields. Writing is accomplished as follows

1. The proper device must be selected

```
vfConnection.SelectDevice( 1 );
```

2. Group, row and field must be specified for an update query

```
group = vfConnection.GetDialogGroupById( groupId );
field = group.GetFieldById( fieldId );
query = new VFUpdateQuery( group, row, field, value );
```

3. Finally, the query must be executed

```
vfConnection.ExecuteQuery( query );
```

In order to access the feed settings and current consumptions the group id 1 is used. Each row of the result corresponds to a valve and each field is some setting or measurement. For the sake of

comparison Table 3 lists the field ids for some of the fields that were also seen in the two other feeding systems.

Field Id	Description
4	Feed curve
6	Fixed deviation from feeding curve
9	Number of animals
10	Average mass of an individual animal
12	Feed fed today

Table 3 A selection of field ids for group id 1 for the Big Dutchman NT-99 system

Similarly to the other feeding system APIs the NT-99 API does not provide any means for change notifications. Thus, if needed the PWAL should provide the functionality via polling.

3.4 Alternatives

As shown in the previous sections the interface to process computers can be very diverse. And that is only for feeding systems from two different vendors. When other types of systems such as climate control are considered the diversity increases. The diversity also makes it difficult to interchange information between the various systems installed on the farm. This fact has also spurred interest in defining standards for interconnecting such systems and it would also be possible to interface to the systems via these standards.

Currently, two proposals for data exchange on the farm are being developed. One is ISOagriNET[4] and the other is Datastandard[5]. They are both based on the same definition of data called the agricultural data element dictionary (ADED), that is, the identification and structure of information is common, see description in (ISO 11788) or at the Datastandard homepage for an introduction. The largest difference is the choice of communication protocol where ISOagriNET is based on a line based protocol send over TCP or via UDP multicasting. The protocol of Datastandard is web-service on a small set of remote procedure calls transferred via SOAP and http.

Because ISOagriNET does not have an API we will not describe it in any further details and focus on Datastandard. The details of ISOagriNET can be found in (ISO 17532).

3.4.1 Datastandard

Datastandard provides a relatively simple SOAP based RPC interface to various data sources. Using its definition files most development environments can create an API where the protocol, serialization and deserialization are abstracted away and hidden from the developer.

In order to retrieve data from a service the request data call is used. In order to retrieve information about feed settings for a group (equivalent to a valve) the steps are as follows

1. Create the web service proxy and set the url

```
service = new Datastandard();
service.Url = "http://127.0.0.1:9000";
```

2. Make a request for the feed settings for a group entity. The call requires user name and password which needs to be configured. Here we use "test" for both. The third argument, 903008, is the entity id for group feed settings and the fourth argument can be used to request data with filtering. A null filtering is the same as no filtering. The fifth argument will after

[4] http://www.isoagrinet.org/
[5] http://datastandard.dk/

successful completion contain the requested data. The errorState signals if the call was successful or an error code if something did not work out.

```
errorState = service.RequestData("test","test", "903008", null, out results);
```

Updating data is more complicated as we need to

1. Create service proxy and set the URL

```
service = new Datastandard();
service.Url = "http://127.0.0.1:9000";
```

2. Setup the search criteria such that only a specific valve is changed

```
search[0] = new Search();
search[0].DD = "901002"; // Location ID item DD
search[0].In = new [] { "1" }; // Specify the valve id for search criteria
```

3. Setup an entity with the proper new values. The entity DD shows which type of data that should be updated.

```
newEntity = new Entity();
newEntity.DD = "903008"; // Group feed setup
newEnitty.Items[0] = new Item();
newEntity.Items[0].DD = "900350"; // Number of animals DD
newEntity.Items[0].Value = value; // New Number of animals
```

4. Send the update request

```
errorState = service.UpdateData("test","test",search,newEntity,out results);
```

5. The errorState signals whether the update call ended without errors. The results argument contains the entities that were updated by the call.

The Datastandard also provides functionality to subscribe to events. However, it is optional to support the subscribe functionality and no devices or systems support that functionality as of the time of writing.

4. RFID Tags and Readers

Among the devices found within the traceability scenario, one of the key ones that need to be integrated are Radio Frequency Identification (RFID) **tags** and their respective **readers**. RFID is the term used for all wireless non-contact systems which transmit data from a tag using radio-frequency waves, for the purposes of automatic identification (in the form of a serial number) and tracking; part of the broad category of Automatic Identification (Auto-ID) Technologies. The high-level principle of RFID technology can be seen in Figure 3.

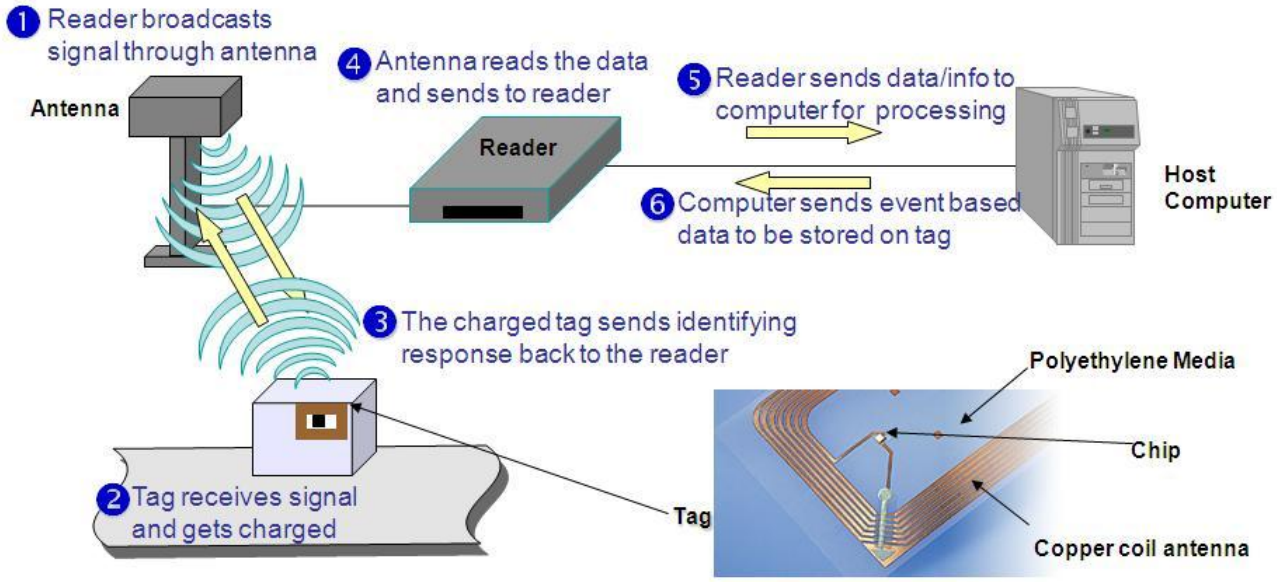

Figure 3: High-level principle of RFID technology[6]

Among their characteristics and capabilities, RFID systems provide three main features that have driven their adoption in many applications: first, some tags require no battery and are powered by the electromagnetic fields used to read them; second, tags can be read from up to several metres away without the need to be within line of sight of the reader; and third, tags can be embedded in the tracked object. Thanks to these features, RFID has become the *de facto* technology for several applications like: access management, tracking of goods/animals/persons, logistics and asset management, contactless payment, machine readable travel documents and many other applications.

Within the food traceability scenario, RFID tags are used mainly for tracking animals and products along the whole production and distribution chain. Thus, the ability to read them, or in other words, to capture and process identification events in the ebbits platform is crucial for many of the applications foreseen in the project. Some of these applications include the identification and tracking of animals within farms and slaughterhouses, cold-chain monitoring in transportation, nutrition facts and information for consumers, and in general tracking the meat in all the steps of its production for an automatic, complete, reliable and fast traceability (for instance when recalling a flawed product). More details on the role of RFID in traceability can be found on deliverable *D8.2 A Survey of Physical World in Manufacturing and Traceability Scenario.* In a future smart home scenario an intelligent refrigerator may use RFID technology to track it contents and adjust cooling accordingly.

The proliferation of different commercial RFID solutions, has led industries in the last decade to join forces in the global not-for-profit standards organization (GS1) to push the definition and adoption of

[6] Taken from http://octaware.blogspot.it/2010/11/rfid-technology-many-of-us-have-heard.html

a worldwide royalty-free set of standards: the EPCglobal Architecture Framework[7]. As stated in the standard (EPCglobal, 2010a), such framework "is a collection of interrelated standards for hardware, software, and data interfaces, together with core services that are operated by EPCglobal and its delegates, all in service of a common goal of enhancing the supply chain through the use of Electronic Product Codes (EPCs)". An overview of the architecture can be seen in Figure 4, showing how the different components of the framework fit together to form a cohesive whole.

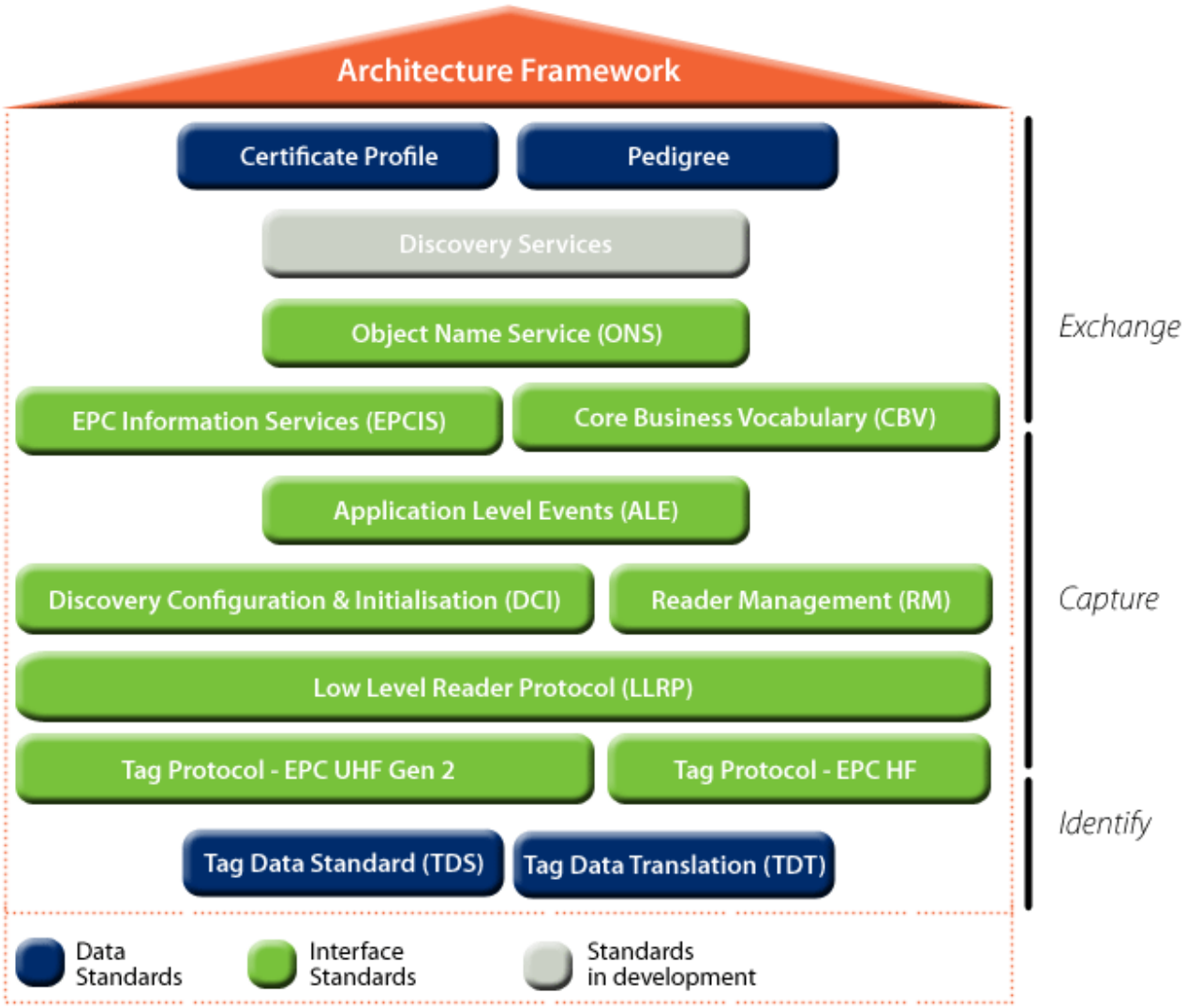

Figure 4: EPCglobal Architecture Framework – Standards Overview[8]

A fundamental principle of the EPCglobal Architecture Framework is the assignment of a unique identity to physical objects, loads, locations, assets, and other entities whose use is to be tracked. By "unique identity" is simply meant a name, such that the name assigned to one entity is different than the name assigned to another entity. In the EPCglobal Architecture Framework, the unique identity is the Electronic Product Code (EPC), defined by the EPCglobal Tag Data Standard. Moreover, such framework specifies different hardware, software and data aspects within the processes of identification capture and exchange of identity information both intra- and cross-

[7] More info on http://www.gs1.org/gsmp/kc/epcglobal/architecture

[8] Taken from http://www.gs1.org/gsmp/kc/epcglobal

enterprise. These aspects include for example the format of storing identities within the RFID tags, the physical channel and protocols to inquire them, the interfaces to manage and control RFID readers, as well as other upper-layer interfaces, such that a RFID tag can be uniquely identified.

However, the particular interest of the WP8 in EPCglobal is focused to the interaction and abstraction of the physical world through RFID, i.e. to integrate readers and handle tag data in a standardized way. This could be achieved within the framework developed and presented in *D8.3 Physical World Adaptation Layer*, by developing a PWAL driver which interacts with some of the interfaces defined in the EPCglobal Architecture Framework at capture layer. However, before making this implementation, a deeper study on the involved EPCglobal standard interfaces has to be made in order to identify the key functionalities needed to enable such interaction.

The rest of this section is organized as follows. First, the integration of RFID readers will be discussed by reviewing the EPCglobal Architecture Framework with a particular emphasis on the standards that specify the interfaces for both event capture and reader discovery and management. Then, the integration of RFID tags will be studied, extending the analysis to different types of tags, in particular those equipped with sensors, from which extra information could be retrieved when reading these tags, targeting at applications in the field of cold-chain certification and passive monitoring in general.

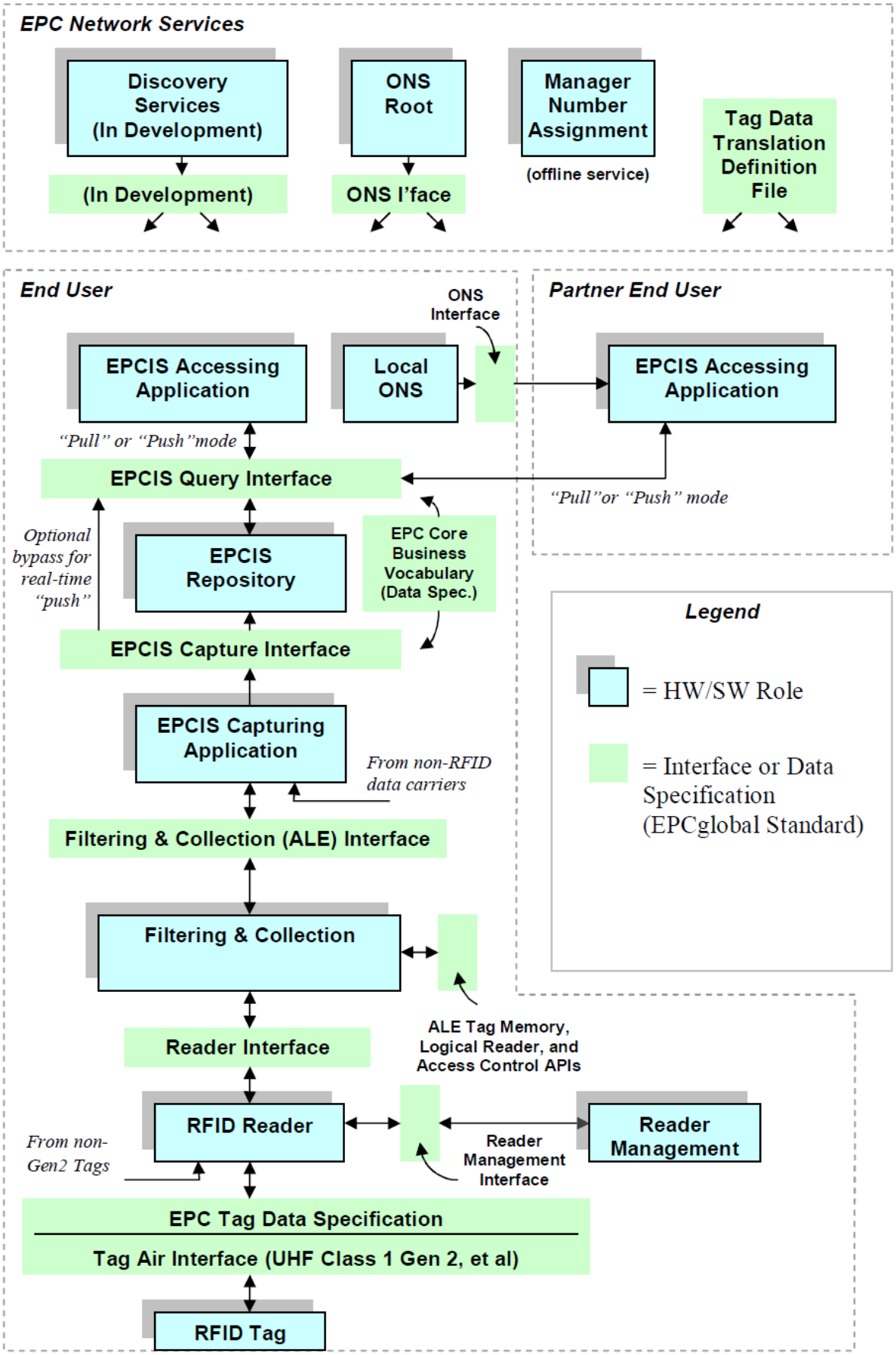

Figure 5: EPCglobal Architecture Framework – Functional View (EPCglobal, 2010a)

4.1 Integration of EPCglobal compliant RFID readers

As it can be appreciated in Figure 4, the EPCglobal Architecture Framework is divided in three main layers: identify, capture and exchange. The role of these layers is better depicted in the functional view of the architecture as shown in Figure 5. In particular, it can be noticed how two *software roles*, namely Filtering & Collection and Reader Management, interact with the reader through two different interfaces. These *roles* are the functionalities that the ebbits platform should provide to its applications, and in particular, to the traceability ones.

Therefore, the integration of these RFID readers, which are in general capture devices, within the ebbits platform require a deeper analysis on the capture and management interfaces of the framework, in particular the Low Level Reader Protocol (LLRP), the Reader Management (RM), the Discovery, Configuration and Initialization (DCI) and the Application Level Events (ALE) standards.

Moreover, a proper integration of RFID readers within the ebbits platform following the EPCglobal Architecture Framework can be split in three main tasks:

1. the PWAL driver must implement an LLRP client in order to interact with the reader;
2. the PWAL framework in conjunction with the Device Discovery Manager (DDM) and the Sensor Fusion capabilities found in the Device Proxies should perform the Filtering & Collection role;
3. and finally, Device Proxies should expose the services and functionalities specified in the Application Level Events (ALE) interface.

In each one of these tasks, it is important to understand the responsibilities of the elements and interfaces involved. Below the Filtering & Collection role, these responsibilities can be classified into three broad functional groups: tag data processing (*Data path*), Reader device management (*Management path*) and Reader control and coordination (*Control path*). For instance, according to EPCglobal, RFID readers are responsible for:

- reading the EPCs of RFID Tags within range of one or more antennas (via a Tag Air Interface) and reporting the EPCs to a client application (via the Reader Interface);
- writing the EPC to a tag (via a Tag Air Interface), when allowed by the tag, as commanded by a client application (via the Reader Interface);
- reading and writing user data (via a Tag Air Interface), when provided by the tag, as directed by a client application (via the Reader Interface);
- operating additional features such as kill, lock, etc. (via a Tag Air Interface), when supported by the tag, as directed by a client application (via the Reader Interface);
- and optionally, providing additional processing such as filtering of EPCs, aggregation of reads, and so forth.

Meanwhile, the Filtering & Collection role coordinates the activities of one or more RFID Readers that occupy the same physical space and which therefore have the possibility of radio frequency interference. It also raises the level of abstraction to one suitable for application business logic. Filtering & Collection is responsible for:

- receiving raw tag reads from one or more RFID Readers;
- processing required to reduce the volume of EPC data, transforming raw tag reads into streams of events more suitable for application logic than raw tag reads, including filtering (eliminating some EPCs according to their identities, such as eliminating all but EPCs for a specific object class), aggregating over time intervals (eliminating duplicate reads within that interval), grouping (e.g., summarizing EPCs within a specific object class), counting (reporting the number of EPCs rather than the EPC values themselves), and differential analysis (reporting which EPCs have been added or removed rather than all EPCs read);
- commanding write, lock, kill, or other operations upon tags by performing the corresponding low-level operations on one or more RFID Readers;

- determining which processing operations as described above may be delegated to the RFID Reader, and which must be performed by the Filtering & Collection role itself, according to the capabilities of associated RFID Readers;
- decoding raw tag values read from tags into URI representations defined by the Tag Data Standard, and conversely encoding URI representations into raw tag values for writing using the Tag Data Translation Interface to obtain machine-readable rules for doing so;
- mapping "logical reader names" and physical resources such as reader devices and/or specific antennas;
- mediating between multiple client application requests for data when those requests involve the same set or overlapping subsets of RFID Readers;
- and optionally, providing decoding and encoding of non-EPC tag data in Tag user memory or other memory banks;
- also optionally, setting and controlling the strategy for finding tags employed by RFID Readers;
- and finally, coordinating the operation of many readers and antennas within a local region in which RFID Readers may affect each other's operation, e.g., in order to minimize interference this role may control when specific readers are activated so that physically adjacent readers are not activated simultaneously, or make use of reader- or Tag Air Interface-specific features, such as the "sessions" feature of the UHF Class 1 Gen 2 Tag Air Interface, to minimize interference.

Finally, the Application Level Events (ALE) interface, whose main responsibility is provide the standard interfaces to the Filtering & Collection role in two planes: a *data plane* and a *control plane*. In one hand, within the data plane, the responsibilities of the ALE interface are:

- providing means for one or more client applications to request EPC data from one or more Tag sources;
- providing means for one or more client applications to request that a set of operations be carried out on Tags accessible to one or more Tag sources, including writing, locking, and killing;
- insulating client applications from knowing how many readers/antennas, and what makes and models of readers are deployed to constitute a single, logical Tag source;
- providing declarative means for client applications to specify what processing to perform on EPC data, including filtering, aggregation, grouping, counting, and differential analysis;
- providing means for client applications to request data or operations on demand (synchronous response) or as a standing request (asynchronous response);
- providing means for multiple client applications to share data from the same reader or readers, or to share readers' access to Tags for carrying out other operations, without prior coordination between the applications;
- and provides a standardized representation for client requests for EPC data and operations, and a standardized representation for reporting filtered, collected EPC data and the results of completed operations.

In the other hand, within the control plane, the responsibilities of the ALE interface are:

- providing means for client applications to query and configure the mapping between logical reader names as used in read/write requests and underlying physical resources such as RFID Readers;
- providing means for client applications to configure symbolic names for Tag data fields;
- and providing means for management applications to secure client access to the ALE interface.

As it can be noticed, the integration of RFID readers following the EPCglobal Architecture approach is quite complex and requires a deeper understanding of the interfaces supported by the readers. For this reason, the following subsections describe the readers capture, management and control interfaces, exploring those functionalities that could be exploited by the PWAL, with a particular attention to the LLRP interface.

4.1.1 EPCglobal Low-Level Reader Protocol (LLRP) Interface Standard [9]

As stated in the EPCglobal Architecture Framework documentation (EPCglobal, 2010a), a Reader Interface provides the means for software to control aspects of RFID Reader operation, including the capabilities implied by features of the Tag Air Interfaces. The EPCglobal Low Level Reader Protocol (LLRP) standard (EPCglobal, 2010b) is designed to provide complete access to all capabilities of the UHF Class 1 Gen 2 Tag Air Interface, including reading, writing, locking, and killing tags, as well as providing control to clients over the use of the RF channel and protocol-specific tag features such as Gen2 inventory sessions. According to EPCglobal, the LLRP is responsible for:

- providing means to command an RFID Reader to inventory tags (that is, to read the EPCs carried on tags), read tags (that is, to read other data on the tags apart from the EPC), write tags, manipulate tag user and tag identification data, and access other features such as kill, lock, etc.;
- providing means to access RFID Reader management functions including capability discovery, firmware/software configuration and updates, health monitoring, connectivity monitoring, statistics gathering, antenna connectivity, transmit power level, and managing reader power consumption;
- providing means to control RF aspects of RFID Reader operation including control of RF spectrum utilization, interference detection and measurement, modulation format, data rates, etc.;
- providing means to control aspects of Tag Air Interface operation, including protocol parameters and singulation parameters;
- and providing access to processing features such as filtering of EPCs, aggregation of reads, and so forth.

The fundamental operations a Reader performs on a tag population are *inventory* and *access*. **Inventory** is the operation of identifying tags, and comprises multiple air protocol commands. Using the singulation scheme, the Reader detects a single tag reply and requests the EPC memory contents from the tag. **Access** is the operation of communicating with (reading from and/or writing to) a tag. An individual tag must be uniquely identified prior to access. Similar to the inventory operation, access comprises multiple air protocol commands. In addition, a Reader can choose a subset of the tag population for inventory and access.

In addition, the LLRP interface between the Client and the Reader facilitates the management of Reader devices to mitigate Reader-to-tag and Reader-to-Reader interference and maximize the efficiency of singulation and data operations over the tag population. This is achieved by enabling the Reader device operation at the full performance level of the air protocol. LLRP is specifically concerned with providing the formats and procedures of communications between a Client and a Reader. Messages from the Client to the Reader include getting and setting configuration of Readers, capabilities discovery of Readers and managing the inventory and access operations at the Readers. Messages from the Reader to the Client include the reporting of Reader status, RF survey, and inventory and access results. A summary of these tasks can be seen in Figure 6, where a typical runtime timeline is displayed.

[9] More info on http://www.gs1.org/gsmp/kc/epcglobal/llrp

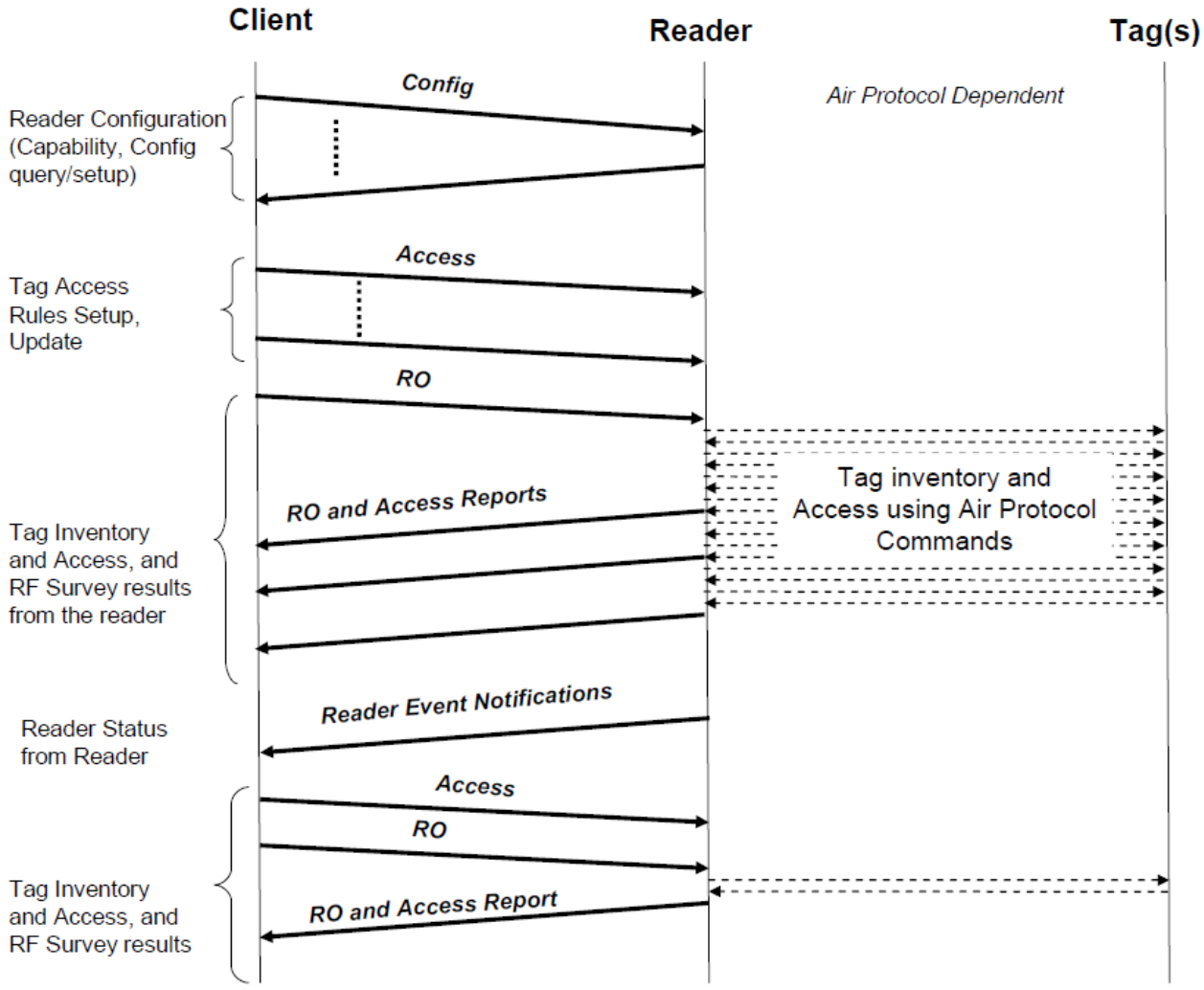

Figure 6: Low-Level Reader Protocol – Typical Runtime Timeline (EPCglobal, 2010b)

LLRP is an application layer protocol and does not provide retransmission, or reordering facilities. State consistency between the Client and the Reader is critical for the correct functioning of the system. Using LLRP messages, the Client updates the Reader state which includes Reader configuration parameters, dynamically created data structures and possibly vendor defined data. For this reason, LLRP requires acknowledgements for the Client to Reader transactions – this provides a fail-safe mechanism at the LLRP layer to cope with network error situations. Also, to cope with intermittent connections, a Client can request a Reader's configuration state to confirm that a Reader's state is consistent with the Client after the Client reconnects. The Reader to Client messages are primarily reports, status notifications or keepalives. Only the keepalives are acknowledged by the Client. As a final remark, the transport protocol between Readers and Clients is TCP, with default port is 5084 as established by IANA[10], but other ports can be used.

The LLRP client within the PWAL driver could be based on some of the open source implementations recommended by EPCglobal (i.e., LLRP toolkit[11]), or an ad-hoc implementation following the standard specification.

[10] See http://www.iana.org/assignments/port-numbers
[11] Found on http://www.llrp.org

4.1.2 EPCglobal Reader Management (RM) Interface Standard [12]

The second interface of interest for the abstraction of RFID readers is the one in charge of exposing their management capabilities. Again, according to EPCglobal, the Reader Management (RM) interface is responsible for:

- providing means to query the configuration of an RFID Reader, such as its identity, number of antennas, and so forth;
- providing means to monitor the operational status of an RFID Reader, such as the number of tags read, status of communication channels, health monitoring, antenna connectivity, transmit power levels, and so forth;
- providing means for an RFID Reader to notify management stations of potential operational problems;
- providing means to control configuration of an RFID Reader, such as enabling/disabling specific antennas or features, and so forth;
- and optionally, providing means to access RFID Reader management functions including device discovery, identification and authentication, network connectivity management, firmware/software initialization, configuration and updates, and managing reader power consumption.

Summarizing, the Reader Management Specification, is targeted at monitoring the health of networked RFID readers. Specifically, RM provides mechanisms to get operational statistics for a broad range of RFID operations - including reading and writing of tags, network availability of the reader, and the status of specific antennas and peripherals that may be connected to the reader.

In this context, heath monitoring refers to information about reader operational states. As a concrete example, the RM specification provides data structures to count tag reads and writes at a per-antenna level, as well as providing data structures to indicate whether the reader believes those antennas are functional or not; how long the radio(s) have been on, what peripherals may be attached to a reader, and so on. Additionally, the RM specification defines optional mechanisms that allow readers to notify management systems of alert conditions based on configurable parameters.

The Reader Management Protocol is specified in three distinct layers, as illustrated in Figure 7:

- the *Reader Layer* specifies the content and abstract syntax of messages exchanged between the Reader and Client. This layer is the heart of the Reader Protocol and the Reader Management Protocol, defining the operations that Readers expose to monitor their health;
- the *Messaging layer* specifies how messages are formatted, framed, transformed, and carried on a specific network transport, as well as, how an underlying network connection is established (i.e., initialization messages required to establish synchronization or security services like authentication, authorization, message confidentiality, and message integrity);
- finally, the *Transport Layer* corresponds to the networking facilities provided by the operating system or equivalent, and is specified elsewhere.

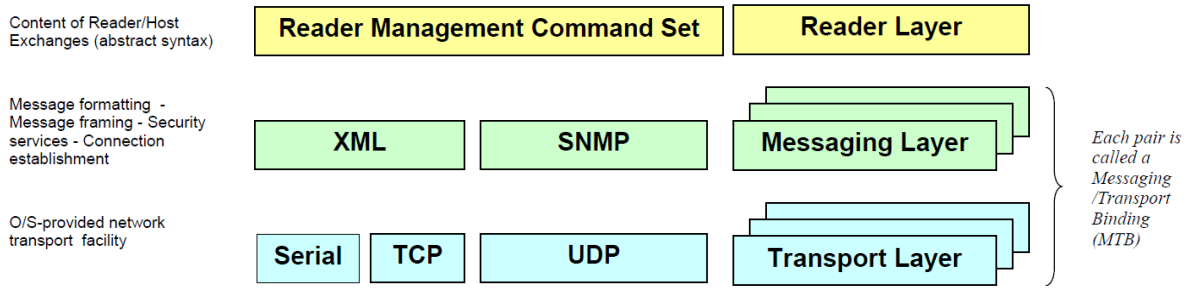

Figure 7: Reader Management – Protocol Layers Mapping

[12] More info on http://www.gs1.org/gsmp/kc/epcglobal/rm

At Messaging layer, RM specifies two bindings, or network protocols, for implementing the standard. One is **SNMP**, a widely used protocol for monitoring network devices and servers. Commercial and open-source network management systems use SNMP, so readers that implement SNMP should be easy to integrate into existing network management systems. The other binding is an **XML-based protocol** designed to extend the binding used by the former Reader Protocol. Both bindings were designed to provide equivalent visibility into reader operations, so as to provide end users and solution provider's maximum flexibility in designing systems to fit their needs.

The interface between the Reader Layer and the Messaging Layer is defined in terms of message channels, each one representing an independent communication between Reader and Client:

- The *Command channel* carries requests issued by the Client to the Reader, and responses to these requests carried from the Reader to the Client. All messages exchanged on the command channel follow this request/response pattern. Most configuration interaction between Reader and Client takes place through the command channel. The Reader MAY support one or more Command Channels.
- The *Alarm channel*(s) carries messages issued asynchronously by the Reader to the Client. Messages on the alarm channel only flow to the Client. The alarm channel is primarily used to support a mode of operation in which the Reader asynchronously delivers health and status alerts to the Client, The Reader controls when alarm channel messages are sent to the Client; the Client does not request alarm channel messages. The Reader MAY support one or more Alarm Channels for the delivery of Management Alarms.
- Finally, the Reader may support one or more *Notification Channels* for delivery of tag data.

4.1.3 EPCglobal Discovery Configuration and Initialization (DCI) Interface Standard [13]

As previously described, the Reader Management standard focuses on monitoring reader's operational status and on notifying management stations of potential operational problems. Instead the Discovery, Configuration and Initialization (DCI) for Reader Operations standard (EPCglobal, 2009) focuses on reader discovery identification, configuration and network connectivity management. These two standards fulfil different and complementary responsibilities of the reader management interface.

The purpose of the DCI interface is to specify the necessary and optional operations of a Reader and Client that allow them to utilize the network to which they are connected to communicate with other devices, exchange configuration information, and initialize the operation of each Reader, so that the Reader Operations Protocols can be used to control the operation of the Readers to provide tag and other information to the Client. According to EPCglobal, this interface is responsible for:

- providing means for the Reader to discover one or more Access Controllers;
- providing means for the Access Controller to discover one or more Readers;
- providing means for the Reader to discover one or more Clients;
- providing means for the Reader and Access Controller to exchange identity information and authenticate that identity information;
- providing means for the Client and Access Controller to authenticate their communications and operations;
- providing means for the Access Controller to configure the Reader, including a means to update the software and/or firmware on the Reader;
- providing means for the Access Controller to initialize the Reader, providing parameters necessary for the Reader to begin operation;

[13] More info on http://www.gs1.org/gsmp/kc/epcglobal/dci

- and providing means for the Reader and Access Controller to exchange vendor-specific information.

DCI utilizes the Control and Provisioning of Wireless Access Points (CAPWAP)[14] protocol which must be implemented by both the Reader and the Access Controller (which could be hosted in the Client machine, or elsewhere in the network) in order to support the DCI features. DCI operation comprises the following phases (as shown in Figure 8): device discovery, device authentication and identity exchange, firmware download if necessary, device configuration and finally device initialization.

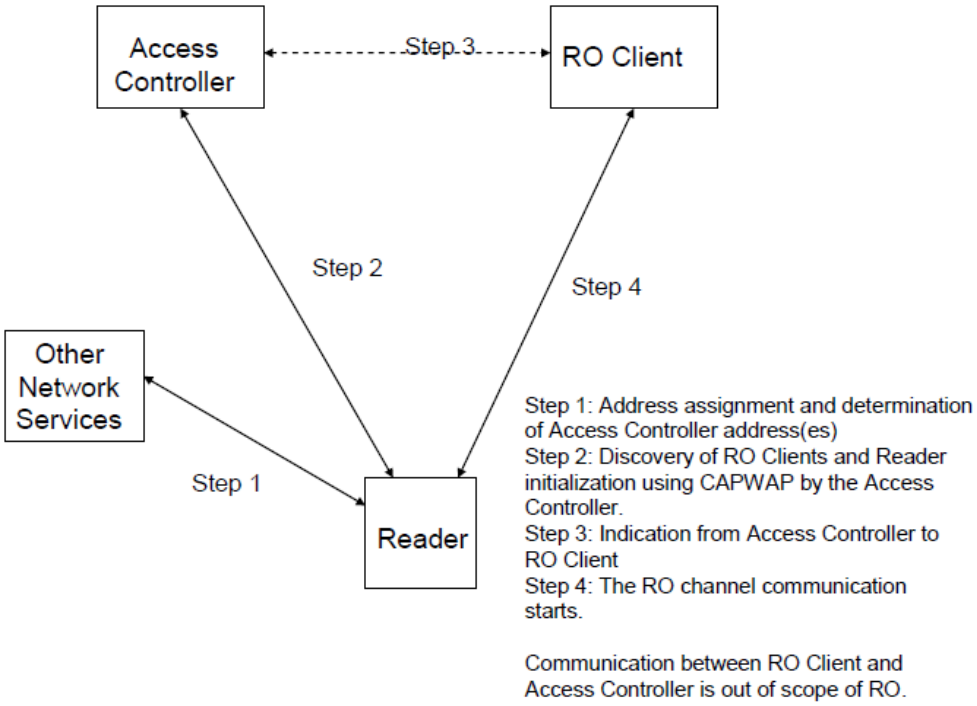

Figure 8: Discovery, Configuration & Initialization – Operation Steps (EPCglobal, 2009)

4.2 Integration of EPCglobal compliant RFID tags and sensors

Having described the integration aspects and interfaces for RFID readers, it is important as well to discuss the role of the different RFID Tags considered within the Architecture Framework. In particular, EPCglobal has defined a tag classification system to describe tag functionality. The responsibilities of the RFID Tag role based on classification are shown below.

Class-1 – Identity Tags: Passive-backscatter Tags with the following minimum features:

- an EPC identifier, optionally writeable;
- a Tag Identifier (TID) that indicates the tag's manufacturer identity and mask ID;
- a "kill" function that permanently disables the Tag This feature may involve additional data stored on the tag such as a kill password;
- an optional extended TID that may include a unique serial number and information describing the capabilities of the tag;
- an optional recommissioning of the Tag;
- an optional password-protected access control;

[14] More info in http://datatracker.ietf.org/wg/capwap/charter

- and optional user memory (for application data apart from the EPC).

Class-1 Tags are relevant for animal identification on the farm. The extra features as seen for the other classes are most likely handled by other means and the extra cost of an advanced tag is not justified. For instance, temperature control of the environment is handled by the climate system.

Class-2 – Higher-Functionality Tags: Passive Tags with the following anticipated features above and beyond those of Class-1 Tags:

- an extended Tag ID as described above (required in Class-2, as opposed to optional in Class-1);
- an extended user memory;
- an authenticated access control;
- and additional features as will be defined in the Class-2 standard.

Class-3 – Battery-Assisted Passive Tags: Semi-passive Tags with one or more of the following anticipated features above and beyond those of Class-2 Tags:

- a power source that may supply power to the Tag or to its sensors;
- and sensors, with or without sensor data logging.

Class-3 Tags still communicate passively, meaning that they (i) require a Reader to initiate communications, and (ii) send information to a Reader using either backscatter or load-modulation techniques. Tags of this type could, if cost allows, be of interest on the farm. For example, a tag could monitor the temperature of the animal and detect infections.

Class-4 – Active Tags: Active Tags with the following anticipated features:

- an EPC identifier or other identifier;
- an extended Tag ID;
- an authenticated access control;
- a power source;
- communications via an autonomous transmitter;
- an optional User memory;
- and optional sensors, with or without sensor data logging.

Class-4 Tags have access to a transmitter and can typically initiate communications with a Reader or with another Tag. Tag Protocols may limit this ability by requiring a Reader to initiate or enable Tag communications. Because active tags have access to a transmitter, of necessity they have access to a power source. Class-4 Tags shall not interfere with the communications protocols used by Class-1/2/3 Tags.

Thus, depending on the type of Tag employed, the associated *physical world information* that they may access or represents change as well. In particular, Class-3 and Class-4 tags include additional features which extend the RFID tag capabilities in order to provide information coming from attached sensors. These sensors, for instance, could be used to have a record of the products temperature during its journey. The data gathered along the trip, then serve to trace the product's cold chain and control temperature variations.

There are several commercial products available in the current RFID market. An overview of some companies producing relevant products is provided below.

GAO RFID Inc. [15]

It produces a series of RFID tags combining LEDs for visual recognition and sensors for temperature monitoring (ranging from -50°C to 150°C). These tags include 32 MB for data storage, identification, tracking and tracing and location, all in real time. According to GAO, this line of products is ideal for temperature tracking and agricultural purposes. Data transmission can reach distances up to 30 meters from handheld readers or up to 90 meters from fixed interrogators, with particular models combined with active RFID readers.

Figure 9: GAO 2.45 Ghz Active RFID Temperature Sensor Tag

Intelleflex [16]

It provides a particular RFID tag, named TMT-8500, which is a RFID tag combined with high sensitivity temperature sensing functions. Within its extended memory (60kbits), the tag can store up to 3,600 temperature samples. In addition, the tag can be configured to gather the data in intervals from 1 minute to 5 days. It is worth mentioning that, the tag is based on the ISO 18000-6.1 and EPCglobal C1G2 protocols. Support for these protocols (ISO and EPCGlobal) enables the TMT 8500 to provide extended RFID capabilities including enhanced read/write sensitivity equating to read ranges in excess of 100 meters, superior battery power management and the full functionality and configurability of a microcontroller-based temperature sensing application.

Figure 10: Intelleflex TMT-8500 Temperature Monitoring Tag

[15] http://www.gaorfid.com
[16] http://www.intelleflex.com

Sealed Air [17]

Has developed several products related to RF monitoring systems, including the TurboTag® System. This latter, is a temperature monitoring system use to have a complete record of the products temperature during a certain time frame. The card, while moving throughout the distribution cycle, captures and stores time and temperature data points. The data can then be synchronized to specific software for review. Some of the Key Features/Benefits described by the manufacturer are:

- Low cost of consumables (tags) and accessories (readers and software)
- No compromising on accuracy and quality assurance (every tag is calibrated)
- Wireless (RFID) interface for rapid tag initialization and data capture, even through Sealed packaging
- Instant electronic data capture, analysis, and printouts in a single mobile process
- Unique digital kinetic shelf-life estimation capability as well as upper/lower limit Monitoring
- Compliance with EPC data standards and global ISO RFID standards
- Remote, high speed reading of the complete temperature record
- Reusable

In addition to commercial products, an interesting project has been developed within the research community regarding RFID sensors combined with microcontrollers performing computer tasks and controlling additional components such as light sensors or temperature sensors for instance.

Figure 11: TurboTag® Card

WISP [18]

WISP stands for Wireless Identification and Sensing Platform. The term "Identification" comes from "Radio Frequency Identification" (RFID). WISPs are a wireless, battery-free sensing and computation platform conceived as collaboration between Intel Research Seattle and the University of Washington. The platform is powered by harvested energy from off-the-shelf UHF RFID readers. WISP is just a normal EPC gen1 or gen2 tag including a 16-bit general purpose microcontroller operated by the harvested energy. The included microcontroller can perform a variety of computing tasks, including sampling sensors, and report that sensor data back to the RFID reader.

Figure 12: WISP G2.0

[17] www.sealedairspecialtymaterials.com
[18] http://www.seattle.intel-research.net/WISP/

5. Conclusion

This deliverable introduced two basic technologies used in food production and traceability. The feeding controllers on the farms automate and log the feeding of the animals. These systems are the key in providing detailed information about the first part of meat production. The other technology described is RFID tags and readers. RFIDs are increasingly becoming important part in the supply chain and food production is no exception. The primary use is in tracking and localizing goods. The RFID technology is also a likely element of the smart home of the future.

There are three feeding controllers introduced that all have different methods for data exchange. Skiold DM-5000 uses VIGO system for data exchange, Skiold DM-6000 uses file based data exchange and Big Dutchman NT-99 uses NETIPPC interface. This is the reality in the farming environment where no dominant standards have emerged as of now. In order to integrate the feeding controllers PWAL drivers need to be implemented.

RFID tags and readers have gained a lot of success and partially because their use is standardized by the GS1 organization. There is an Architecture framework available to insure compatibly with other RFID systems. Ebbits will need to implement the **Low-Level Reader Protocol**, the **Reader Management** and the **Discovery Configuration and Initialization** interface standards.

Tags that have been extended with sensors, microcontrollers and memory will in the future have large role in the food production and supply chain because the products are generally sensitive to environmental conditions. These tags are not used on a wide scale at the moment but with widespread use could improve food safety and information accuracy considerably. A good example would be RFID tags with temperature sensor that could verify that the cold chain is not broken.

6. References

(EPCglobal, 2007) EPCglobal Inc. 2007. Reader Management (RM) – Ratified Standard v1.0.1. http://www.gs1.org/gsmp/kc/epcglobal/rm

(EPCglobal, 2009) EPCglobal Inc. 2009. Discovery, Configuration and Initialization (DCI) for Reader Operations – Ratified Standard v1.0.1. http://www.gs1.org/gsmp/kc/epcglobal/rm

(EPCglobal, 2010a) EPCglobal Inc. 2010. The EPCglobal Architecture Framework v1.4. http://www.gs1.org/gsmp/kc/epcglobal/architecture

(EPCglobal, 2010b) EPCglobal Inc. 2010. Low Level Reader Protocol (LLRP) – Ratified Standard v1.1. http://www.gs1.org/gsmp/kc/epcglobal/llrp

(ISO, 1998) ISO 1998, ISO-11788-1, Electronic data interchange between information systems in agriculture – Agricultural data element dictionary – Part 1: General description.

(ISO, 2007) ISO 2007, ISO-17532, Stationary equipment for agriculture – Data communications network for livestock farming.