

# RFID: Numbering and Services

Mobile and Ubiquitous Computing

George Roussos  
g.roussos@bbk.ac.uk

---

# Overview

---

- RFID Systems Architectures
- Middleware functionality and operational model
- API and examples
- EPCglobal services
- EPC Information Service
  - Profiles

# RFID Addressing

- Identifiers in RFID
- A brief history of object numbering schemes
- Object identifiers
  - EPCglobal Electronic Product Code
  - Ubiquitous ID
  - Other object numbering schemes
- Addressing objects
- The Internet of Things

## Identifiers in a Gen2 tag

- Tag identification (TID) memory bank
  - An 8-bit ISO 15963 allocation class identifier
    - For EPCglobal Tags it is 0xE2
  - A 12-bit Tag mask-designer ID
  - A 12-bit Tag model number.
  - Manufacturers can also include other information if required e.g. tag serial number
- EPC in EPC memory bank
- User memory bank may contain additional application specific IDs

# ISO 14443 IDs

- ISO 14443-A requires fixed Card Identifier (CID)
- CID uniquely related to tag chip
  - Application Family Identifier (AFI) defines separate spaces for CID
- Used by reader to address a specific card
  - Also used in groups to keep specific cards in a particular state
- In ISO 14443-B can be pseudo –random number
- Application layer identifiers are contained in user data space
  - e.g. Oyster card customer number different from ISO ID

## Addressing objects

- User-space object ID
- Generally no additional context data on tag
- Characteristics
  - Universally unique
  - Sub-domain structure
  - Registrar
  - Ownership
  - Mechanisms for mapping to metadata
- There are already some candidates!

# Numbering Systems for Objects

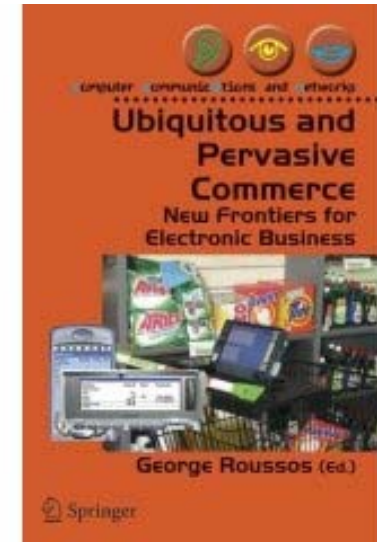
- Barcodes
  - many different types!
- IPv6 addressing
  - too much functionality for objects in many cases
  - requires superior processing capability and >100KB stack)
- Internet 0
  - reduced IP stacks with ISO-1800/IRDA etc link layer
  - Asymmetric, no end-to-end
- Other MAC addresses
  - embedded Zigbee, Bluetooth



UCC/EAN-128, EAN-13, EAN-8, ITF 14.

## Multiple identifiers

- Objects can have multiple IDs in different schemes
  - 658.05 UBI (Dewey Classification Scheme)
  - 1846280354 (ISBN)
  - 9781846280351 (EAN)
  - 6602940 (LIBRI)





---

## Objects are also products

---

- Object manufacturer well positioned to embed ID
- Has been done before at global scale
- Major perceived business benefits in the supply chain
  - logistics, inventory, anti-counterfeiting, demand forecasting, shrinkage
- Possible consumer applications
  - smart things, smart selves, product recalls
- Major technology investment

## Barcodes and the SG1 system

- UPC created in 1973 the first American 10-digit barcode standard (uniform and then Universal product code)
- European Article Numbering introduced in 1977 extended the scheme to the needs of a global market
  - first to separate the data from the data carrier
- Two systems became interoperable in 2005 as EAN.UCC and later SG1 (One Global Standard)
- Under SG1 a variety of standardization activity including RFID within EPCglobal
  - ebXML, Global Data Synchronization Network, Global Standards Management Process, Global Product Classification

## EPC Identifiers

- A global identifier scheme is needed
  - Address allocation, coordination of address space, address semantics, resolution
- EPC is part of SG1 and so has to accommodate existing EAN and related identifiers
- Management of the scheme is via a SG1 subsidiary called EPCglobal Inc
- Protocols are developed in the Auto-ID network of research laboratories

## EPC structure

- EPC tag data standards define “pure identifiers” which are abstract object addresses
- Pure identifiers are stored following the related “physical realization” and “encoding” protocols on the tag
- *Header data* identifies the particular scheme employed in a specific EPC and thus the semantics of the digits
- Current schemes are specific to SG1 and DoD requirements and there is also a general ID

## Encoding schemes

- General Identifiers (GID-96)
- System Identifiers
  - GS1 Global Trade Item Number (GTIN) SGTIN-96 SGTIN-198
  - GS1 Serial Shipping Container Code (SSCC) SSCC-96
  - GS1 Global Location Number (GLN), SGLN-96 SGLN-195
  - GS1 Global Returnable Asset Identifier (GRAI) GRAI-96 GRAI-170
  - GS1 Global Individual Asset Identifier (GIAI) GIAI-96 GIAI-202
- DoD construct (DoD-96) cf. [www.dodrfid.org](http://www.dodrfid.org)

## Types of data

- Serialized Global Trade Item Number (SGTIN) -On item packaging for items where a serial number is used for the unique identification of trade items worldwide within the UCC.EAN System.
- Global Returnable Asset Identifier (GRAI)-On item packaging for items (reusable package or transport equipment).
- Global Individual Asset Identifier (GIAI) -On item packaging for items (used to uniquely identify an entity that is part of the fixed inventory of a company -GIAI can be used to identify any fixed asset of an organization).
- Serialized Shipment Container Code (SSCC)-Items shipped as either pure or mixed case, pallet, (SSCC can be used by all parties in the supply chain as a reference number to the relevant information held in computer database or file).

# Electronic Product Code

016.37000.123456.1000000000			
Header	EPC Manager	Object Class	Serial Number

- *Header*: identifies the length, type, structure, version and generation of EPC
- *Manager Number*: which identifies the company or company entity (today: same as EAN)
- *Object Class*: similar to a stock keeping unit or SKU
- *Serial Number*: which is the specific instance of the Object Class being tagged

- Not specifically related to supply chain applications
- ucode is a 128-bit number
- It is a meta-ID because it can incorporate other numbering schemes
  - provides bindings for JAN, UPC, EAN.UCC, ISBN
- It can be abbreviated for use with low-capacity carriers
  - uses context code
- Distinct domain levels, managed independently
- Registrar is Ubiquitous ID Centre
  - T-Engine Forum, University of Tokyo



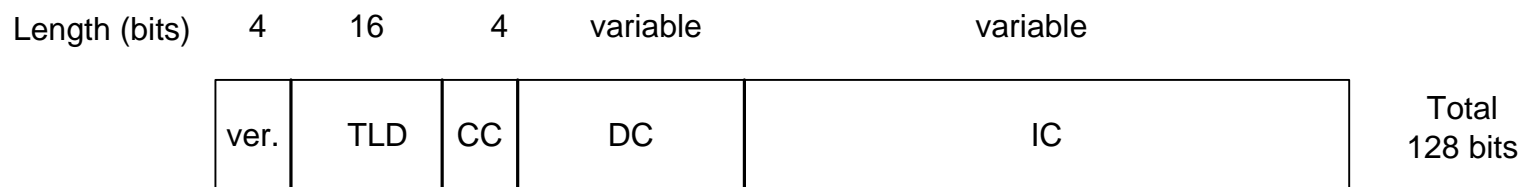
## uID technologies

- Defines specific tag classes
  - also incorporates barcodes
  - microwave, HF and UWB tags
- Defines reader device called the uID Communicator
- Defines software platform
  - Based on TRON
- Address resolution points to uTAD record with object details

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:utad="urn:utad:schema:utad:base:0.0.0#"
  xmlns:pc="urn:utad:schema:pc:example:0.0.0#"
  <rdf:Description rdfabout="ucode:0123....cdef">
    <rdf:type rdfs:resource="urn:utad:schema:pc:example:0.0.0#pc"/>
    <utad:version>0.0.0</utad:version>
```



## unicode structure



- version
- Top Level Domain code
- Class Code specifies the boundary between DC and IC
- Domain Code specifies the type of IC
  - e.g. JAN, ISBN, EPC etc
- Identification Code is the actual object identifier

# Network RFID

- Tags have to minimize cost:
  - very limited storage, i.e. contain ID only
  - very limited computational power
- IDs by themselves are not useful
- Tradeoff: ID is the key to query the network for information
- Need:
  - directory,
  - lookup service
  - (federated) database to hold info
  - associated protocols
- Employ internet and web standards where possible
- Cost and interoperability

# EPCglobal RFID architecture

Discovery	Object Naming Service (ONS)	Discovery of authoritative object manufacturer information
	EPC Discovery Service	Track-and trace chain information discovery (pointers to)
Storage	EPC Information Service	Store and retrieve item and class level usage information
Authentication	EPC Trusted Services	Authentication, authorization and access control

---

# Directory

---

- Map IDs to service locations
  - e.g. map product ID to web service that can be queried for its expiration date
  - does NOT include serial number
- It also maps EPC Manager IDs to EAN.UCC Company prefix
- Requirements: global directory on the internet
- Obvious candidate: Domain Name System

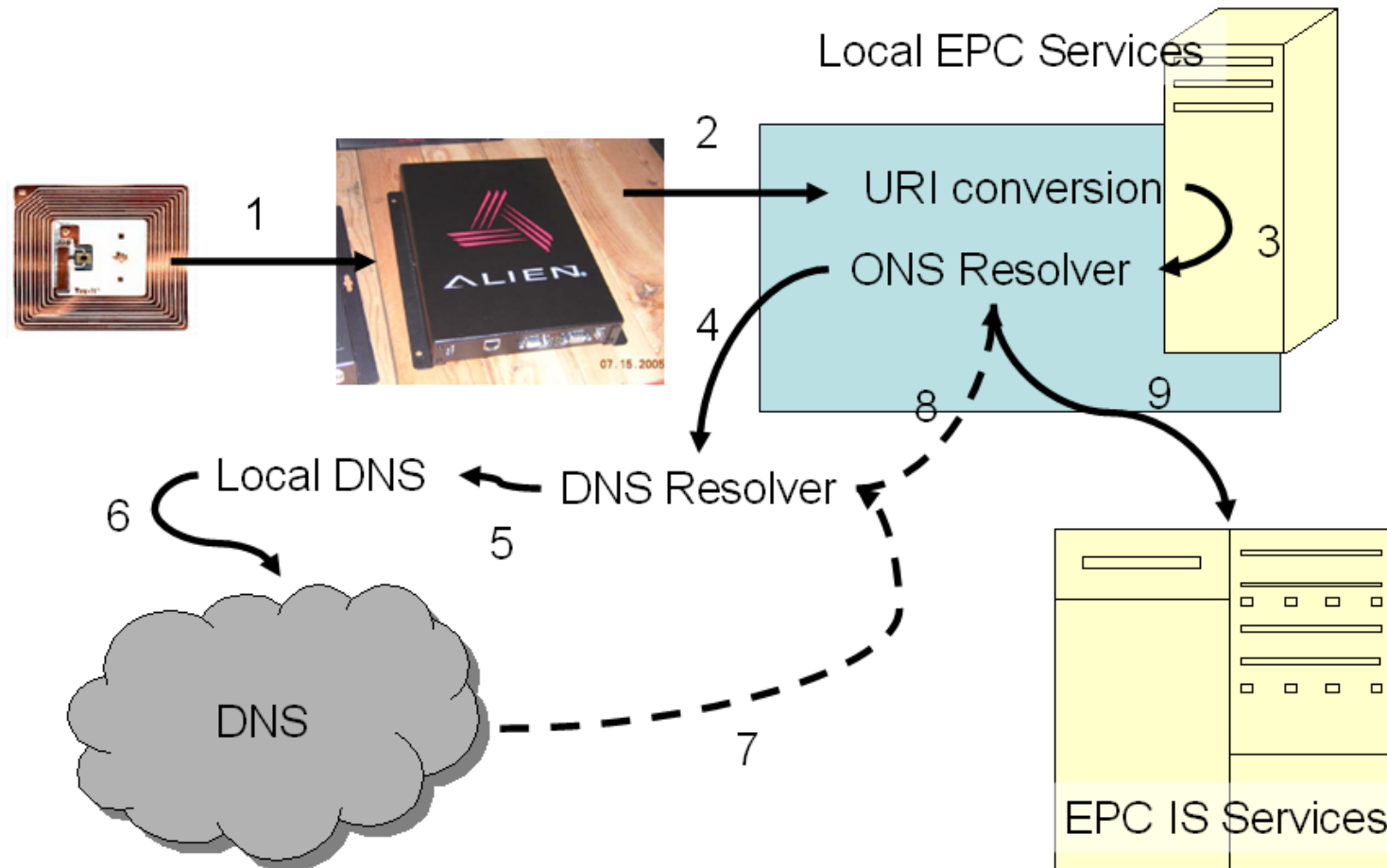
## DNS and X.500

- DNS maps IP numbers to names and vice versa
- In fact, it maintains general Resource Records
- Extensible using NAPTR records
- Well established API and tools
- Efficient lookups, global reach
- Decentralized: location, administration (hierarchical)
- X.500 (ITU) free search but less efficient
- White pages, yellow pages
- Update protocol, security

## ONS lookups

- Using the usual DNS tools
- Two types of DNS resource records
  - NAPTR for EPC codes
  - TXT for company code tables
- Translating the ID into a DNS query
- Follows path to (local to authoritative) onsepc.com through DNS
- Follows path within onsepc.com from root to ID custodian local server

# Query sequence





# Translation

## EPC 64-bit Format:

[10 000 00000000000000 000000000000000011000 0000000000000000110010000]

Step 1: Reader captures and sends to EPC event manager

10 000 00000000000000 000000000000000011000 0000000000000000110010000

Step 2: EPC EM creates URI following Tag Data Standard:

urn:epc:id:sgtin:0614141.000024.400

Step 3: To local ONS resolver:

urn:epc:id:sgtin:0614141.000024.400

Step 4: ONS resolver converts the URI to the equivalent DNS NAPTR query

000024.0614141.sgtin.id.onsepc.com

Step 5: DNS returns result set (redirect to manager domain)

# ONS Resolver

- Remove URI pre-fix

urn:epc:id:sgtin:0614141.000024.400 → 0614141.000024.400

- Remove Serial Number

0614141.000024.400 → 0614141.000024

- Invert

0614141.000024 → 000024.0614141

- Append ONS root

000024.0614141 → 000024.0614141. sgtin.id.onsepc.com

- Issue DNS query e.g.

nslookup 000024.0614141. sgtin.id.onsepc.com (set type=NAPTR)

```
ictx.getAttributes(epcDomainName, new String[]{"NAPTR"});  
(javax.naming)
```

# NAPTR

- Naming Authority Pointer (NAPTR) is a type of DNS Resource Record (RFC 2915)
- Designed for Dynamic Delegation Discovery System (DDDS) applications (RFC 3401, 3401, 3403, 3404)
  - Lazy binding of strings to data
  - Supports dynamically configured delegation
- Uses regular expressions to specify a delegation point within some other namespace
- e.g. used to locate SIP users  
\$ORIGIN 3.8.0.0.6.9.2.3.6.1.4.4.e164.arpa.  
NAPTR 10 100 "u" "E2U+sip" "!^.\*\$!sip:info@example.com!" .

# ONS Result Set

- NAPTR fields:
  - **Order** And **Pref** show priority of this result within the set
  - **Flags** when set to “u” means regular expression containing URI
  - **Service** designates different types of services. The format of this field is EPC+service\_name where service\_name can be pml, html, xmlrpc, and ws
  - **Regexp** specifies a URI for the service being described (for ONS currently it is hostname and additional path information)
  - **Replacement** specifies the replacement portion of the rewrite expression (not used in ONS)

# ONS Result Set Example

Orders	Pref	Flags	Service	Regexp	Replacement
0	0	u	EPC+pml	^.*\$!http://www.epc.dcs.bbk.ac.uk/cgi-bin/epcpml.php!	.
0	0	u	EPC+html	^.*\$!http://www.epc.dcs.bbk.ac.uk/epcpml.jsp!	.
0	0	u	EPC+xmlrpc	^.*\$!http://www.epc.dcs.bbk.ac.uk/exist/epc!	.
0	0	u	EPC+epcis	^.*\$!http://www.epc.dcs.bbk.ac.uk/epc!	.
0	0	u	EPC+ws	^.*\$!http://www.epc.dcs.bbk.ac.uk/ws/epc.wsdl!	.

Service codes:

EPC+pml: Product Markup Language document

EPC+html: Web page description

EPC+xmlrpc: XML Remote Procedure Call interface

EPC+ws: Web Service interface (WSDL)

EPC+epcis: Authoritative EPC IS server

# Example

Solaris 10  
nslookup

Set DNS record  
type to NAPTR

ONS reply



```
hermes{113}% /usr/sbin/nslookup
*** Can't find server name for address 193.61.29.197: Non-existent host/domain
Default Server:  loki.dcs.bbk.ac.uk
Address:  193.61.29.134

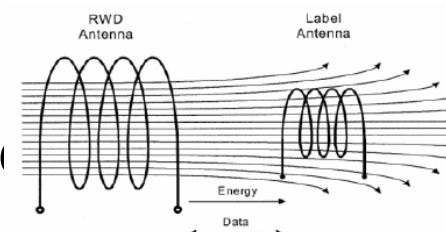
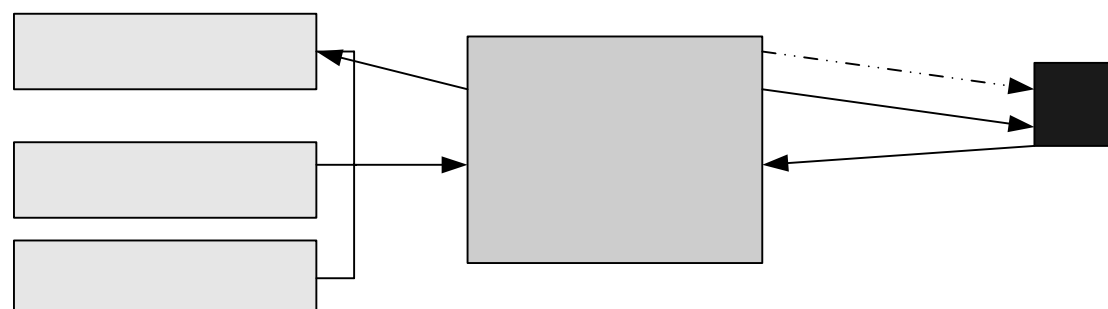
> set type=NAPTR
> 075861.0434687.sgtin.id.onstest.com
Server:  loki.dcs.bbk.ac.uk
Address:  193.61.29.134

Non-authoritative answer:
075861.0434687.sgtin.id.onstest.com      order = 1, preference = 1
      flags = "u"
      services = "EPC+EPCIS"
      rule = "!^.*$!http://reference.verisignepctest.com!"
      replacement = (root)

> █
```

Try test ONS server at [epc.dcs.bbk.ac.uk](http://epc.dcs.bbk.ac.uk)

# RFID Quick Recap



individual

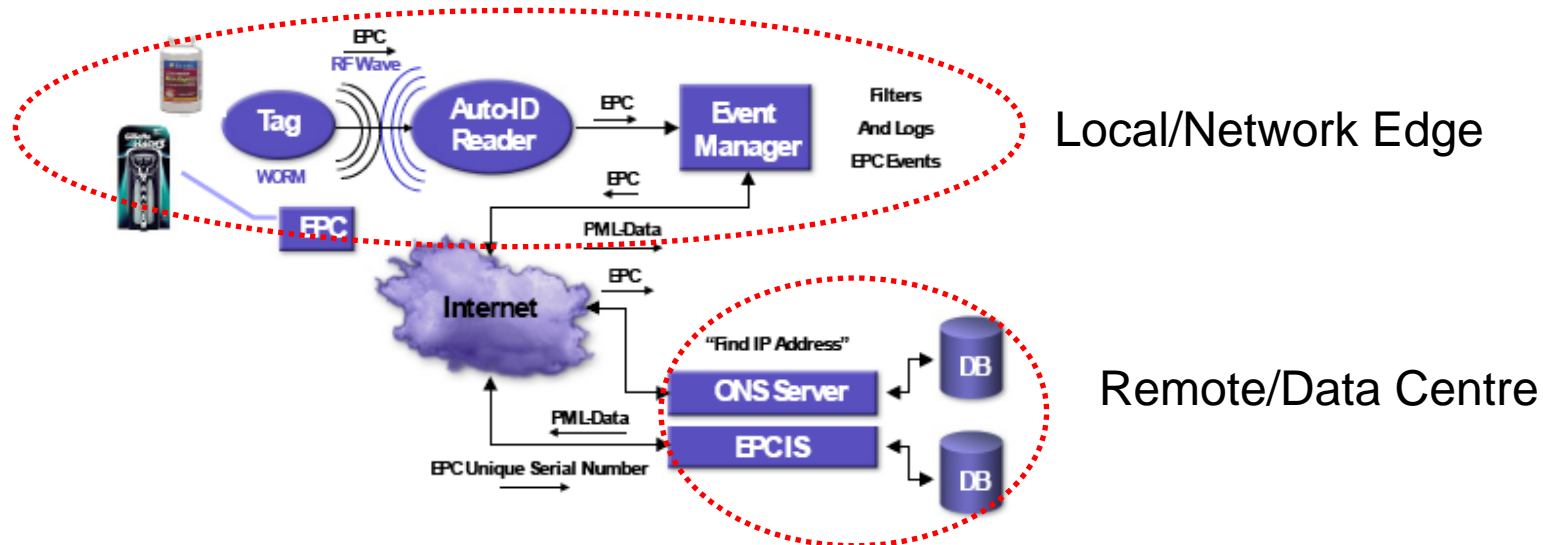
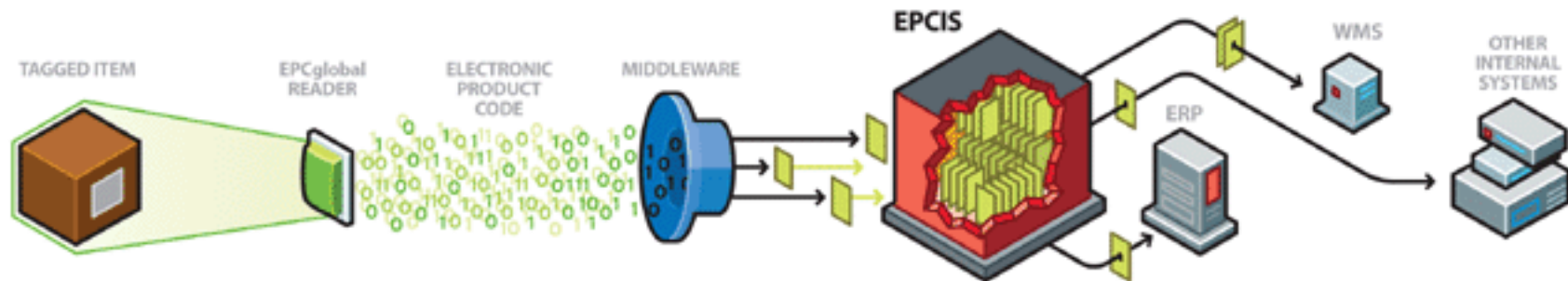
5

1

Network  
management

Reader  
management

# RFID Components

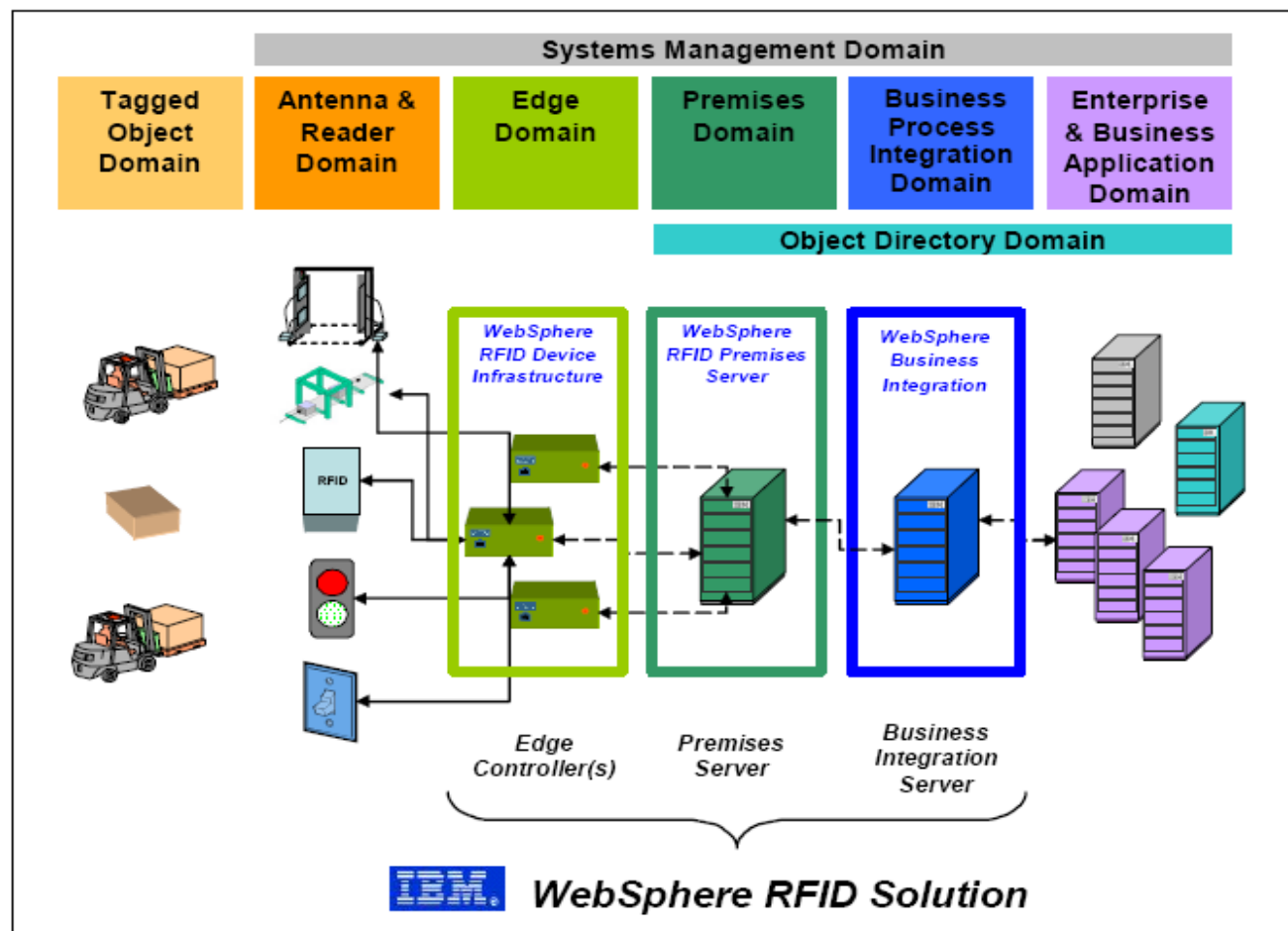




# Tasks in sequence

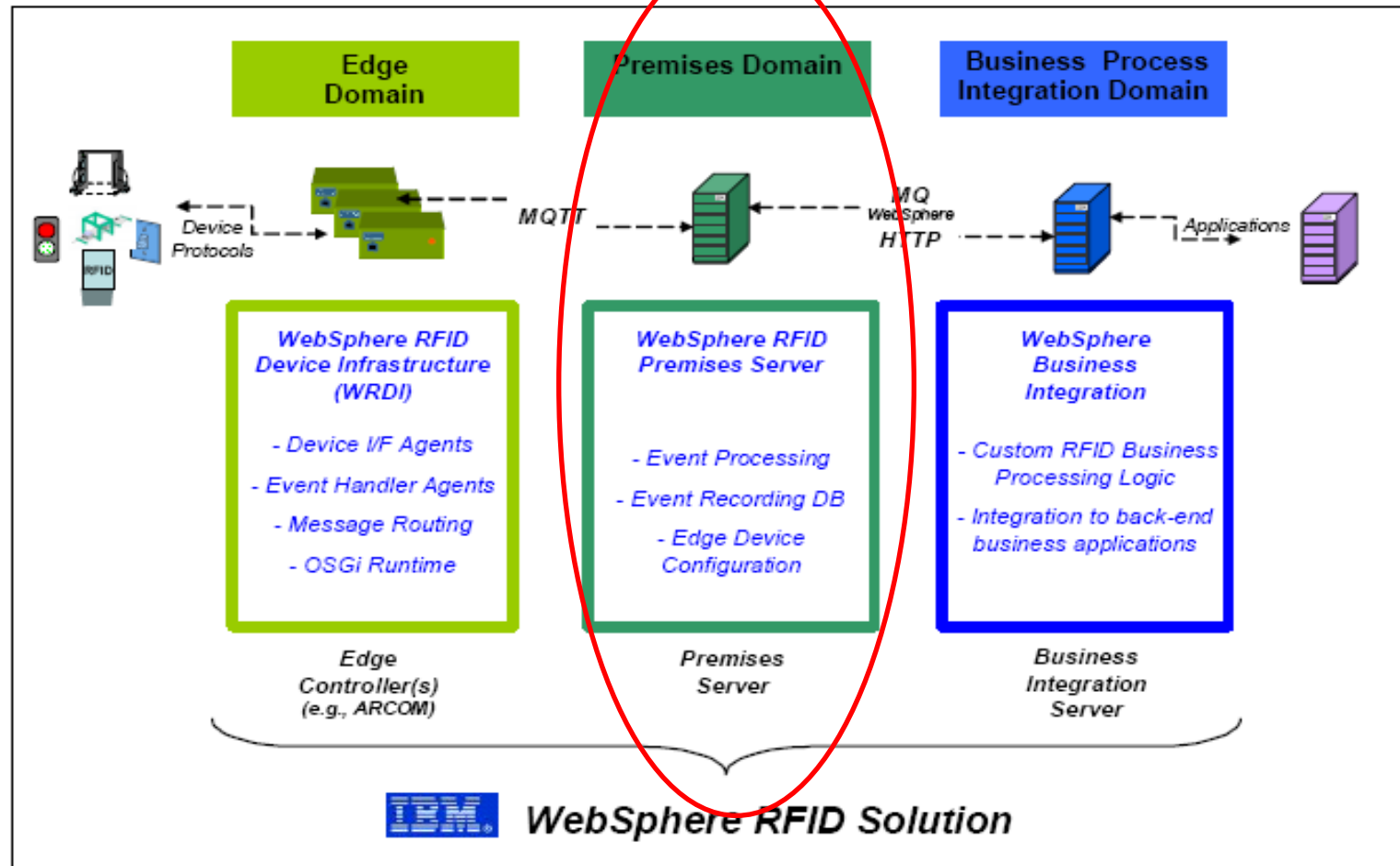
- **Collect Sensor Data**
  - RFID Readers, RFID Label Printers, Temp. Sensors, Laser Diodes, etc
- **Cleanse and Normalize Sensor Data**
  - Cleanse, Normalize, Filter observations
  - Only “Relevant” events are forwarded
- **Dispatch Sensor Data**
  - Deliver Sensor Data to various distribution systems
- **Device Management**
  - Manage and Monitor Sensors and Response Devices
  - Sensors, Light Stacks, Message Boards, Carousels, etc
- **Process Instructions**
  - Local Processing
  - Send instructions to Display/Notification Devices

# RFID System

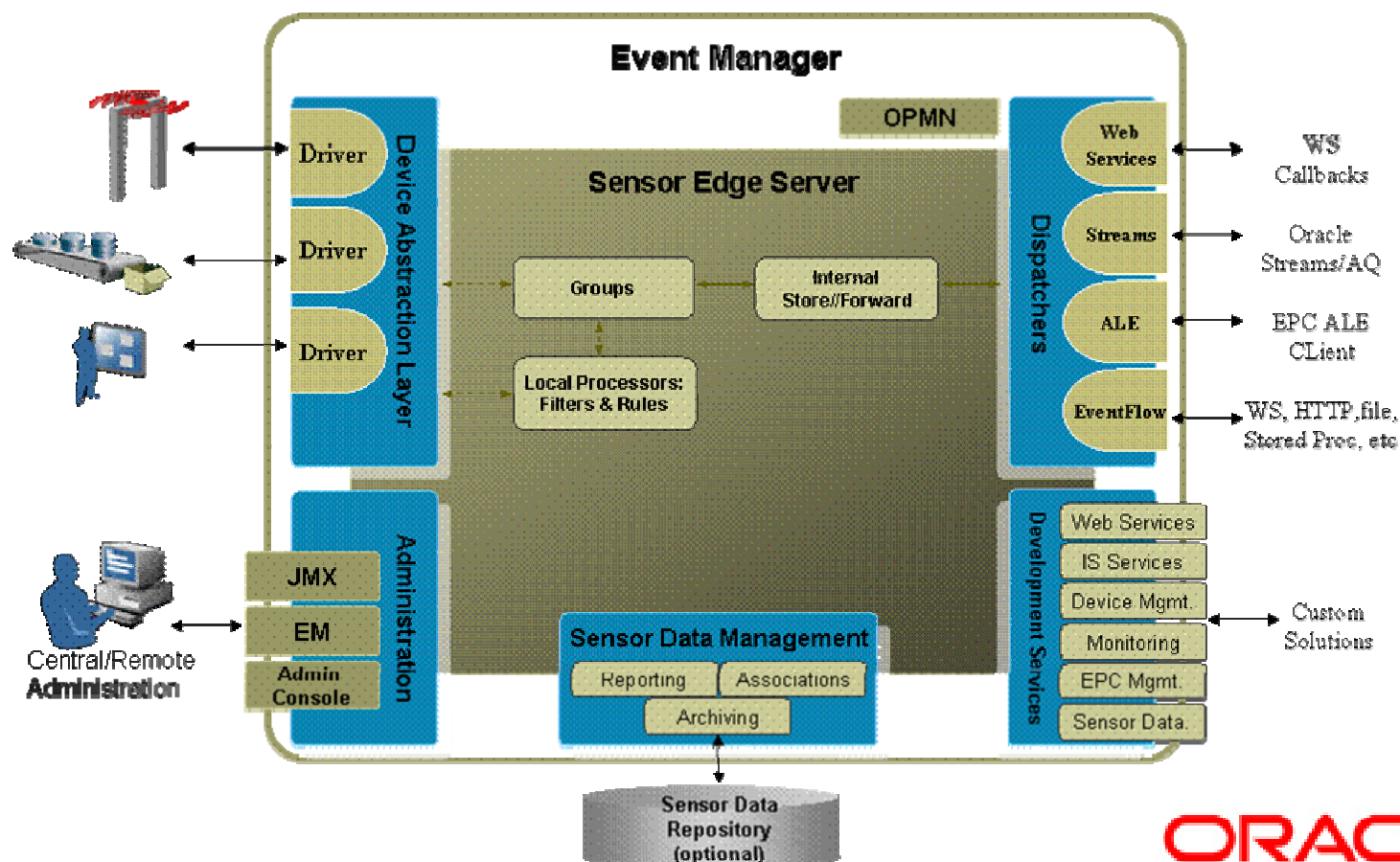


# RFID System (detail)

## Event Manager



# Event Manager Internals

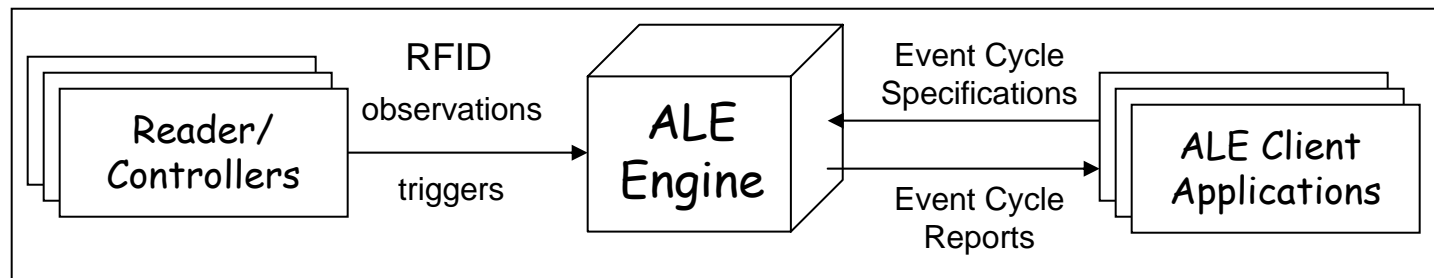


# RFID Middleware

In typical RFID processing systems there is a need to:

- **Reduce** the volume of RFID data that comes directly from RFID readers (and other data sources). Specifically
  - *accumulate* data over specified time intervals
  - *filter* data to eliminate duplicate IDs and IDs that are not of interest
  - *count* and *group* IDs to reduce volume
- **Enhance** application portability and interoperability by decoupling applications from the physical layers of infrastructure through an API
- **Report** in various forms

# Application Level Events



- ALE Middleware Engine processes RFID data coming from readers
- ALE API provides facilities to specify, in a high-level, declarative way, what RFID data they are interested in
  - does not dictate an implementation
  - SOAP bindings map abstract API to Web service implementation
  - does not specify how ALE interfaces with data sources or triggers
- Formal processing model around clients' event cycle specifications

# ALE Terminology

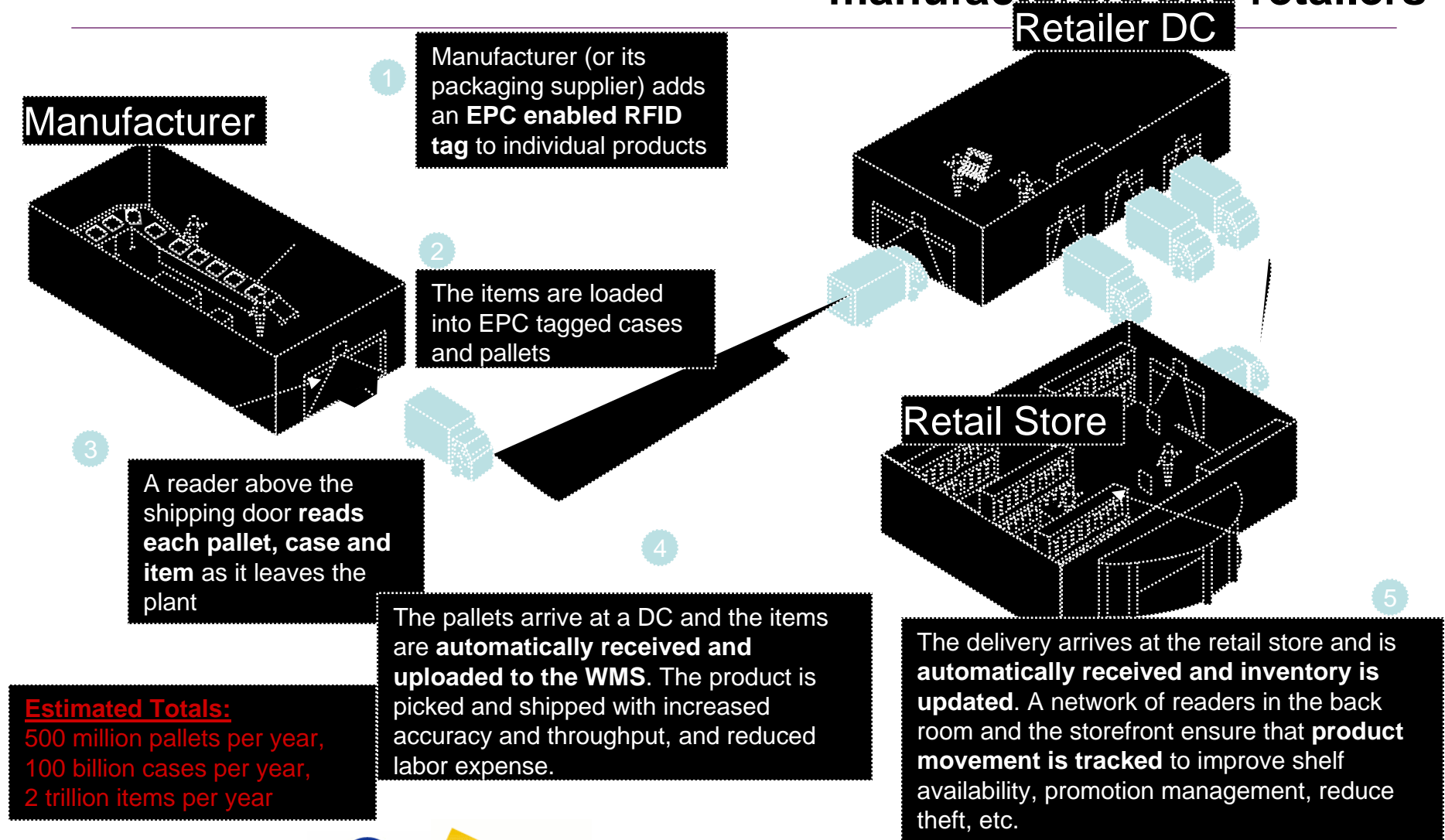
- *Reader*
  - source of raw RFID observations
  - RFID reader, EPC enabled bar code reader, person typing EPC data
- *Read Cycle*
  - smallest unit of interaction with a reader
- *Logical reader*
  - abstract source of EPC data
  - often synonymous with location

## ALE Terminology (cont.)

- **Event Cycle**
  - *smallest unit of interaction between client and ALE*
  - *may consist of one or multiple read cycles*
  - *event cycles are defined by their boundaries*
- **Event Cycle Boundaries**
  - *may extend for a specified interval of time e.g. accumulate reads into five-second intervals*
  - *may occur periodically e.g., report every 30 minutes regardless of the read cycle*
  - *may be triggered by external events e.g. an event cycle starts when a pallet on a conveyer triggers an electric eye upstream of a portal, and ends when it crosses a second electric eye downstream of a portal*
  - *may be delimited when no new IDs are detected by any Reader specified for that event cycle for a specified interval of time*
- **Report**
  - *data about a specified event cycle communicated to a client*



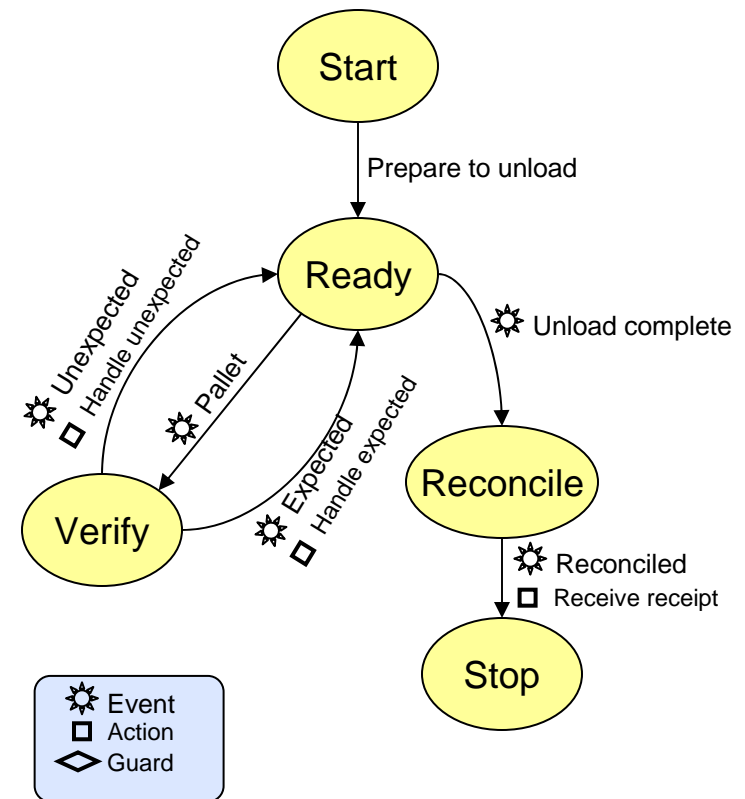
# RFID can make real time tracking of inventory possible and coupled with related process improvements can benefit both manufacturers and retailers



# Example Dock Door Receiving use case and business process

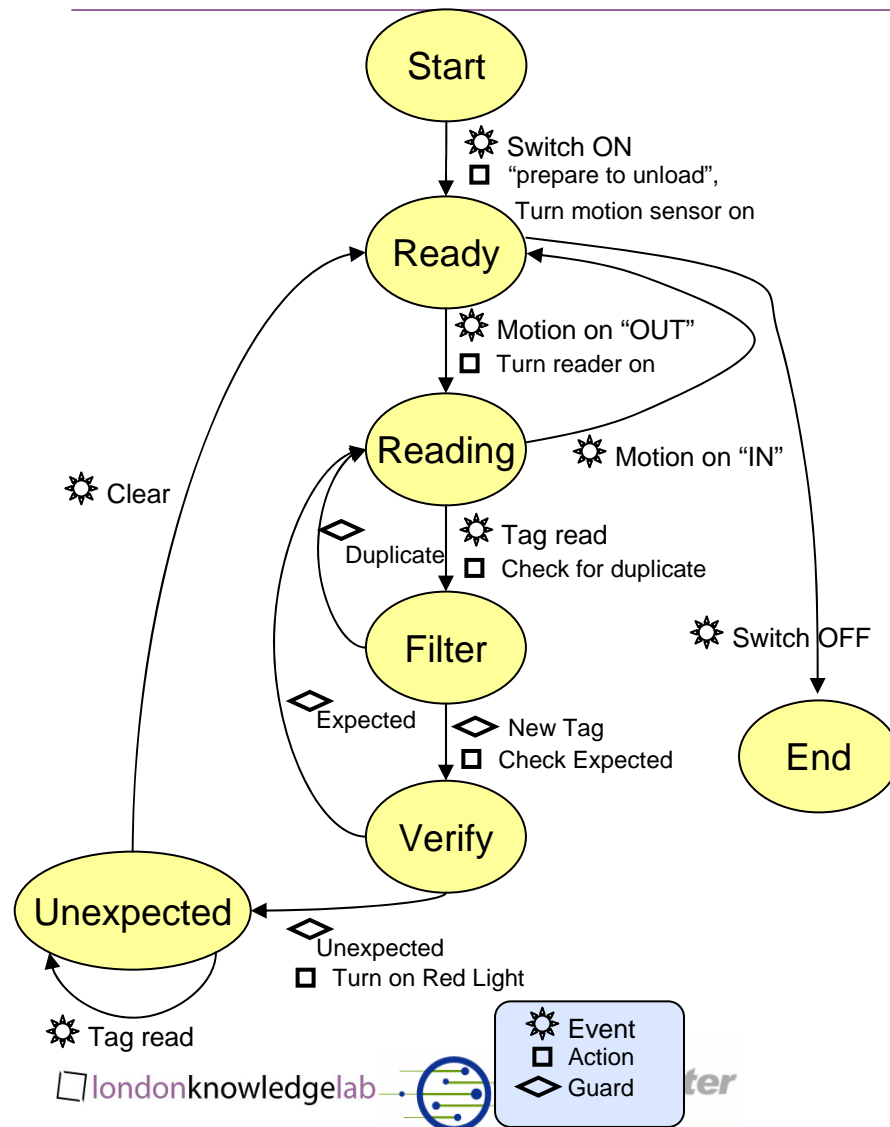
## •Receiving Use Case

- A retailer has previously placed an order for goods with the retail Distribution Center (DC).
- The DC prepares an Advanced Shipment Notice (ASN) at the time of shipping goods and sends it to the retailer.
- At the store receiving dock the operator turns on the dock door switch and starts to unload the truck.
- One pallet at a time is taken through the dock door. Green light indicates dock door is ready, amber indicates the dock door is busy.

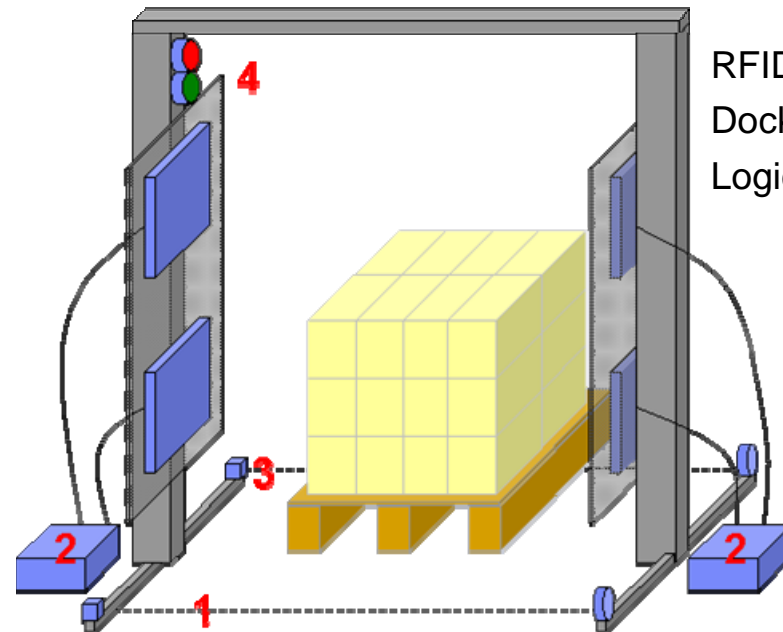


High-level Business Process

# Example Dock Door Receiving RFID Process

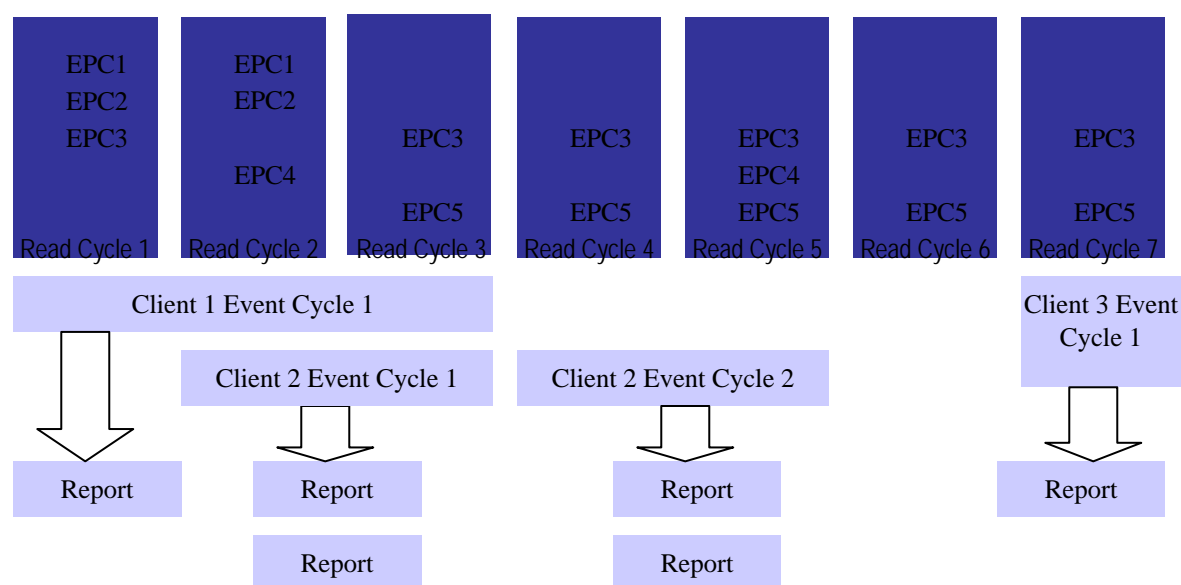


RFID enabled  
Dock Door



RFID enabled  
Dock Door:  
Logical View

# ALE Cycle Example



Source: The Application Level Events (ALE) Specification, Version 1.0

# ALE Core API - Part 1

- **define(specName:string, spec:ECSpec) : void**
  - Define ECSpec to the ALE Engine
- **undefine(specName:string) : void**
  - Undefine specified ECSpec
- **getECSpec(specName:string) : ECSpec**
  - Get ECSpec from engine
- **getECSpecNames() : List**
  - Get names of ECSpecs known by the engine;
- **subscribe(specName:string, notificationURI:string) : void**
  - Subscribe to an ECSpec with a certain notification URI string
- **unsubscribe(specName:string, notificationURI:string) : void**
  - Remove subscription to ECSpec defined with notification URI

## ALE Core API - Part 2

- `poll(specName:string) : ECReports`
  - Poll the ECSpec for reports
- `immediate(spec:ECSpec) : ECReports`
  - Define the spec, poll it and undefine it
- `getSubscribers(specName:String) : List`
  - Who is currently subscribed to this ECSpec
- `getStandardVersion() : string`
  - ALE Standard level supported by this engine
- `getVendorVersion() : string`
  - ALE Engine version number

# Event Cycle Specification

- Event Cycle Specification ECSpec
- **readers : List**
  - A logical reader i.e. any tag source (one or more readers with one or more antennas, also an EPC-enabled bar code reader)
  - For example a location (e.g. a dock door) where read events are captured for all physical readers attached to a location or one specific reader defined for a location
- **boundaries : ECBoundarySpec**
  - Defines a filter and the scope of an event specification
  - It defines the when and how the filter should start and stop and what the filter is
- **reportSpecs : List**
  - Defines the contents and the format of a report
- **includeSpecInReports : boolean**
  - Include ECSpec in report that is returns

## Event Cycle Boundary

- Event Cycle Boundary Spec defines the beginning and the end of an event cycle
- An event cycle starts if one of the following conditions occurs:
  - The specified start trigger is received while an ECSpec is in the Requested state.
  - The repeat period has elapsed from the start of the last event cycle and the ECSpec is still in the Requested state.
- An event cycle ends when one of the following conditions is met:
  - The time interval specified in the duration field expires.
  - The stop trigger is received.
  - The ECSpec transitions to the Defined but Unrequested state.



## Event Cycle Boundary (cont.)

- ECTrigger is a URI that denotes the beginning or end of EC
  - interpretation of this URI is left to the implementation
  - e.g. a motion sensor fires
- ECTerminationCondition specifies how the EC should end
  - TRIGGER: An explicit stop trigger is received.
  - DURATION: Duration expires.
  - STABLE\_SET: EPCs under observation have been stable for a duration.
  - UNREQUEST: there are no requesting/subscribed **clients**.

---

# Filtering and Grouping

---

- Processing of observations for inclusion into report (ECReportSpec)
  - *Filtering* is used to identify specific patterns in the event data (ECFilterSpec )
  - *Grouping* is used to aggregate data collected from different Readers over multiple event cycles (ECGroupSpec)

## Filtering and grouping examples

- Filtering has include and exclude patterns
  - single EPC pattern: urn:epc:pat:gid-96:18.324.7654
  - serial number wildcard: urn:epc:pat:gid-96:18.324.\*
  - item reference range: urn:epc:pat:gid-96:18.[321-326].\*
- Grouping has a single pattern list over which the aggregation is carried
  - group together all observations: urn:epc:pat:gid-96:\*.\*.\*.\*
  - group observations by item type: urn:epc:pat:gid-96:\*.\*.X.\*
  - group per company observations with serial number in range 1-100: urn:epc:pat:gid-96:\*.X.\*.[1-100]

## subscription based example

### ALE Client Application Does the following

#### 1. Creates an event cycle specification

##### –Data sources

- Logical readers : *dockdoor1*

##### –Data aggregation

- Collection period: duration 5 seconds

- Selection criteria

- Exclude filters: exclude *case tags*

##### –Report format (two reports requested)

- Report 1*: Current (tags seen in this cycle)

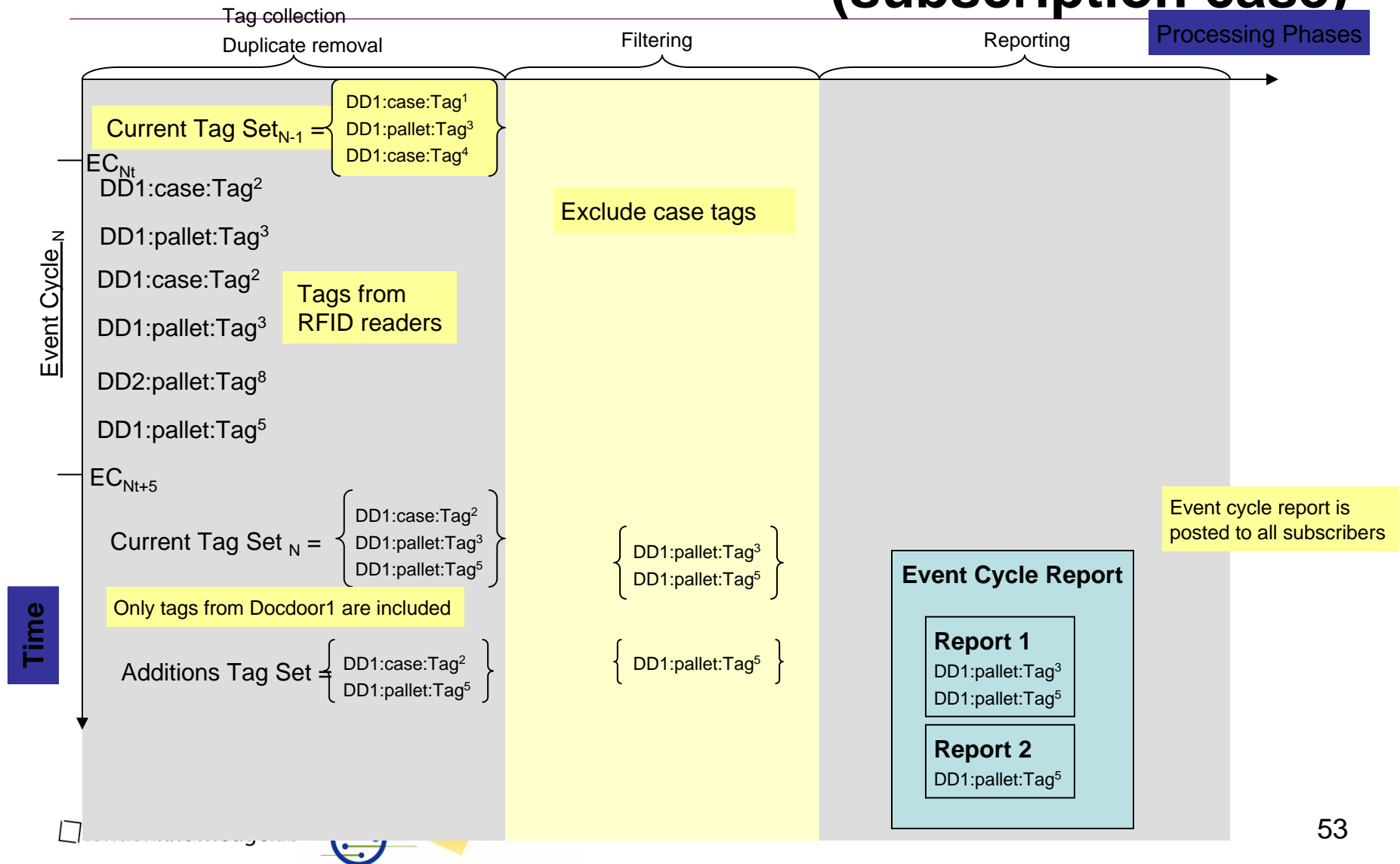
- Report 2*: Additions (tags seen in this cycle, but not in previous cycle)

- Provide a list of EPC Tag URIs

#### 2. Defines (sends) the event cycle specification to an ALE Engine

#### 3. Subscribes to the event cycle specification at the ALE Engine

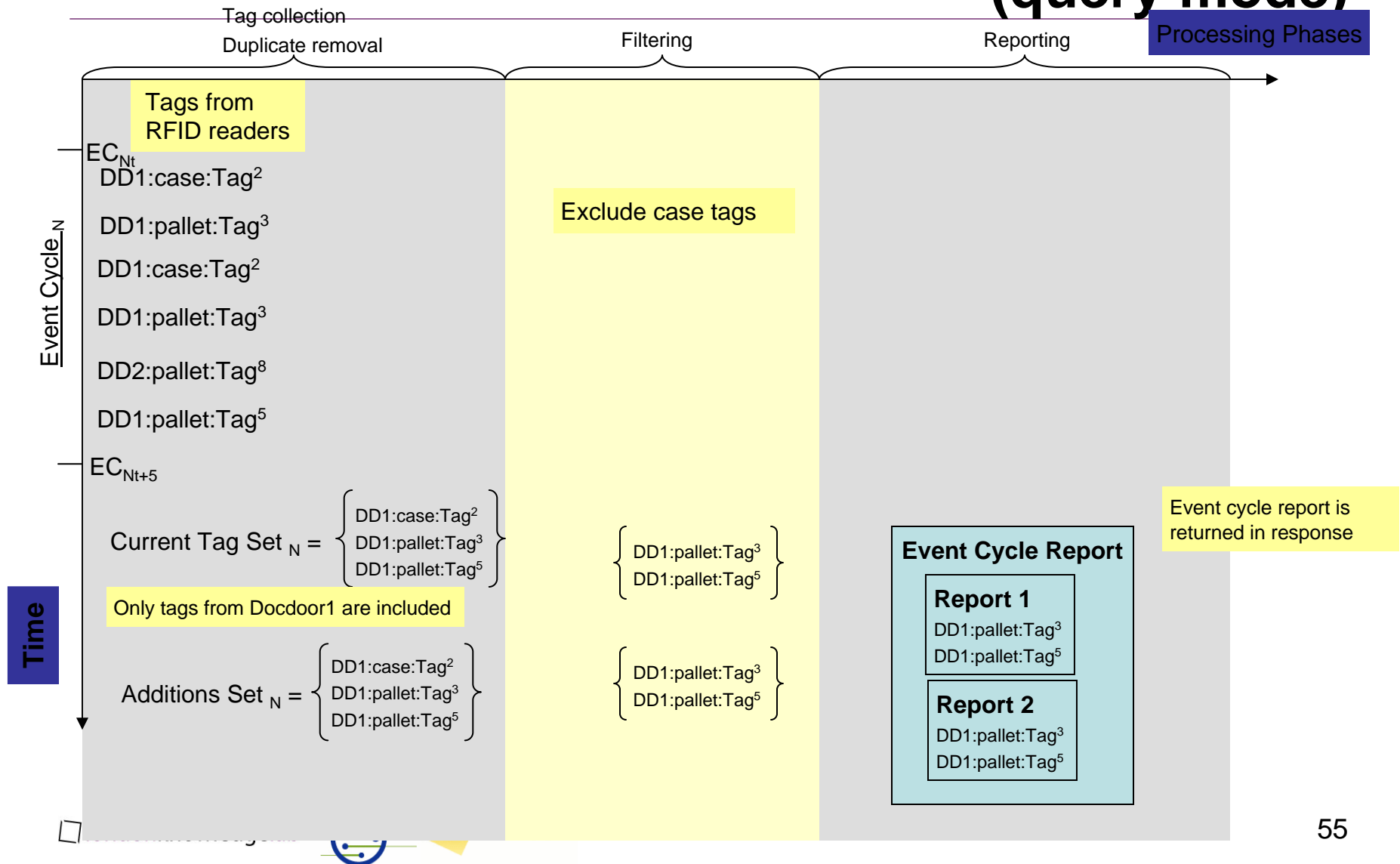
# Example of tag processing by ALE Engine (subscription case)



## Getting EPC data from ALE Engine (query based example):

1. Creates an event cycle specification
  - Data sources
    - Logical readers : *dockdoor1*
  - Data aggregation
    - Collection period: duration 5 seconds
    - Selection criteria
      - Include filters: *include pallet tags*
  - Report format (two reports requested)
    - Report 1*: Current (tags seen in this cycle)
    - Report 2*: Additions (tags seen in this cycle, but not in previous cycle)
    - Provide a list of EPC Tag URIs
2. Sends an *Immediate* request to the ALE Engine
3. Receives event cycle report in response

# Example of tag processing by ALE Engine (query mode)



## Recap: EPCglobal RFID architecture

Discovery	Object Naming Service (ONS)	Discovery of authoritative object manufacturer information
	EPC Discovery Service	Track-and trace chain information discovery (pointers to)
Storage	EPC Information Service	Store and retrieve item and class level usage information
Authentication	EPC Trusted Services	Authentication, authorization and access control

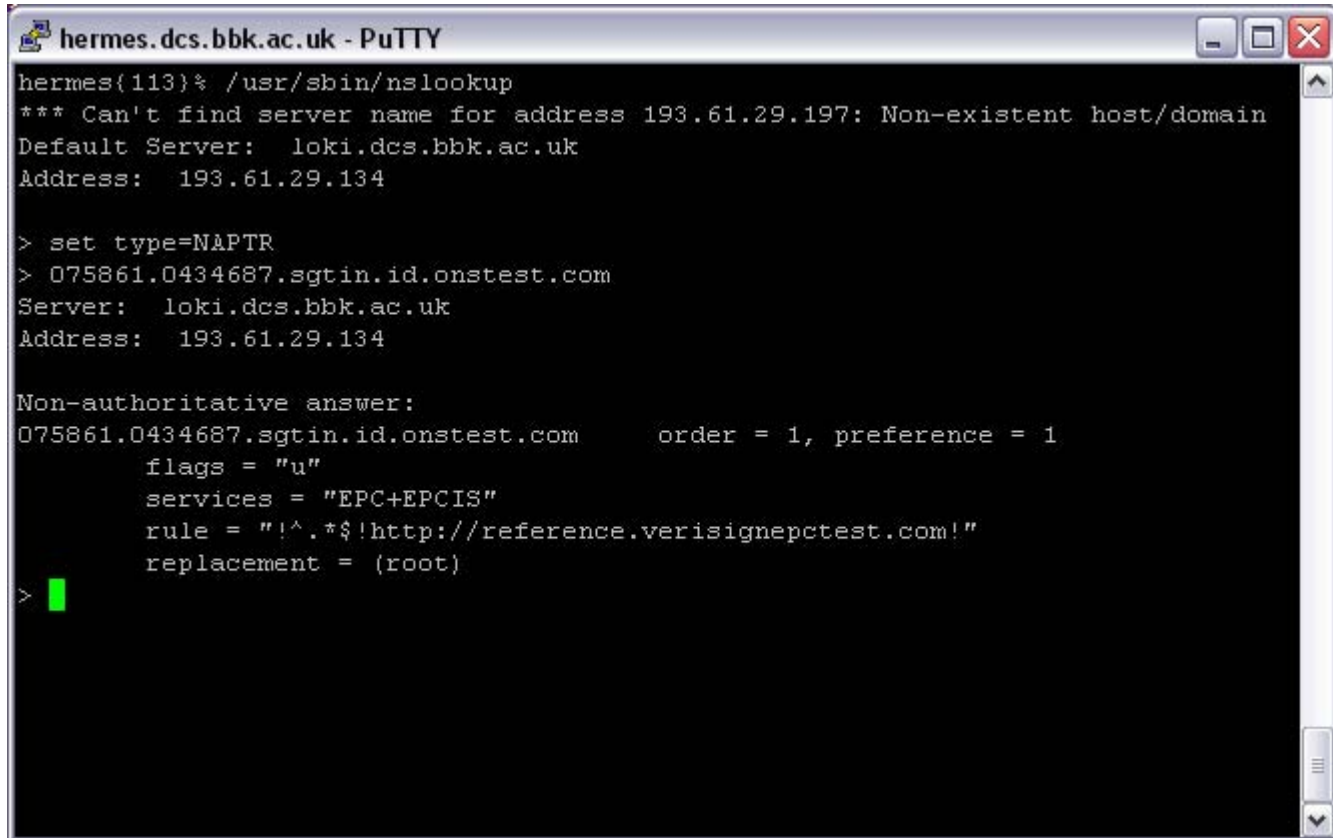


# ONS Recap

Solaris 10  
nslookup

Set DNS record  
type to NAPTR

ONS reply



```
hermes{113}% /usr/sbin/nslookup
*** Can't find server name for address 193.61.29.197: Non-existent host/domain
Default Server:  loki.dcs.bbk.ac.uk
Address:  193.61.29.134

> set type=NAPTR
> 075861.0434687.sgtin.id.onstest.com
Server:  loki.dcs.bbk.ac.uk
Address:  193.61.29.134

Non-authoritative answer:
075861.0434687.sgtin.id.onstest.com      order = 1, preference = 1
      flags = "u"
      services = "EPC+EPCIS"
      rule = "!^.*$!http://reference.verisignepctest.com!"
      replacement = (root)

> █
```

Try test ONS server at [epc.dcs.bbk.ac.uk](http://epc.dcs.bbk.ac.uk)

# Simple EPC Query with XML RPC

```
<methodCall>
  <methodName>lookupEPCDS.getCurrentCustodian</methodName>
  <params>
    <param>
      <value>
        <string>urn:epc:id:sgtin:800900.456.9876</string>
      </value>
    </param>
  </params>
</methodCall>
```

- EPC DS interfaces in flux
- Open source implementation of a possible solution with XML RPC
- Based on the eXist native XML engine and query XQuery processor available via **exist**.sourceforge.net

# Delegation

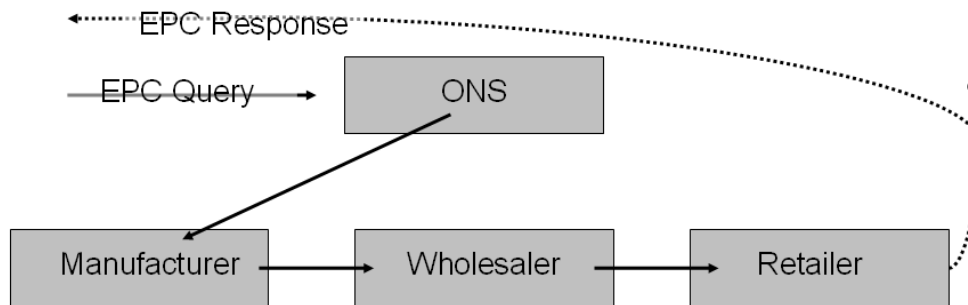
- Domain sgtin.id.onsepc.com controlled by EPCglobal
- Delegation at the EPC Manager layer
  - e.g. domain 0614141. sgtin.id.onsepc.com is delegated to EPC Manager 0614141
- List of EPC managers' EAN.UCC codes (used in bar codes) maintained on ONS
- wget <http://www.onsepc.com/ManagerTranslation.xml>

```
<GEPC64Table date="2006-06-20T08:51:55-05:00">
  <entry index="1" companyPrefix="0037000"/>
  <entry index="2" companyPrefix="0047400"/>
  <entry index="3" companyPrefix="0080878"/>
  <entry index="4" companyPrefix="038004"/>
  <entry index="5" companyPrefix="0036000"/>
  <entry index="6" companyPrefix="0681131"/>
  .....
</GEPC64Table>
```

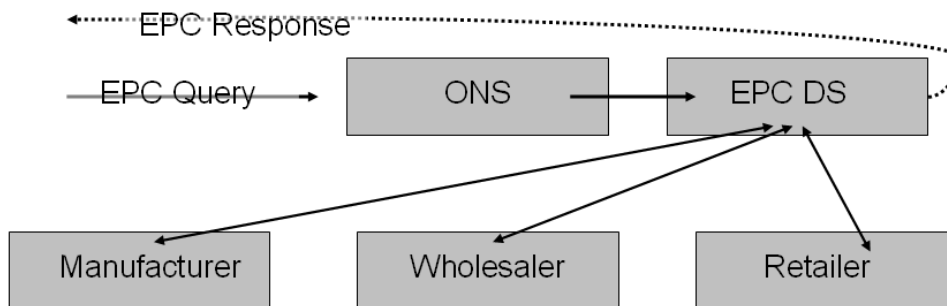
# EPC Discovery Service

- ONS is authoritative at production
  - i.e. ONS points to the originator/manufacture of the object but not subsequent custodians of the EPC (serial) code
  - even more complex if objects are transferred from consumer to consumer
- EPC observation responsibility moves from one custodian to next
  - e.g. from manufacturer, to wholesaler, to retailer
- ONS queries cannot follow through (cf. next slide)
- EPC DS allows track-and-trace applications

# ONS and EPC DS



Track-and-trace with ONS



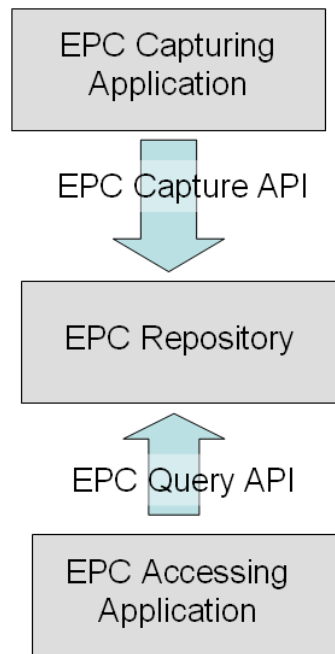
Track-and-trace with EPC DS

- One approach for tractability is to daisy-chain at the ONS from custodian to custodian
- One broken link destroys the sequence
- Solution: keep pointers to each link in the sequence at the EPC DS

## EPC DS records

- change of custodian (arrival / departure)
- change of EPC to track
  - upon aggregation into a container
  - upon re-tagging / re-packaging
- whether the particular EPC is marked for recall
- track forwards to the current custodian
  - to get current information about location/status
  - to determine who to contact about a product recall
- trace backwards to find all custodians which
  - have handled the object and may hold some data on it

# EPC Information Service



- EPC IS: Standard Interface for capture and publication of EPC data (still draft)
- In essence, a distributed database
- Some degree of “semantic” level information
- Provides a common model for location data

## EPC IS records

- Instance level data:
  - Time-stamped observations
- Class level data
  - Classification schemes
- Queries:
  - Which readers saw tag A?
  - Which tags did reader R see?
  - What happened from time  $t_1$  to time  $t_2$ ?



# Full SOAP example

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <AuthInfo>
      <UserToken>
        <UserName>epcisuser</UserName>
        <Password>password</Password>
      </UserToken>
    </AuthInfo>
  </soapenv:Header>
  <soapenv:Body>
    <getCapabilityNSList xmlns="urn:epc:specification:interchange:EPCIS:BaseProfile:xml:wsdl:1"/>
  </soapenv:Body>
</soapenv:Envelope>
```

- EPC IS Authentication profile
- Full SOAP envelope shown
- Any WS framework can be used as client
- HTTP GET is also supported for backward compatibility

## Observation Profile: Log data

- Clients log observations in batch to the EPC-IS
- Same or different locations allowed
- Same or different observations allowed

```
<logEvents>
  <logEvent>
    <location> urn:epc:id:gln:900100.7296</location>
    <observation>
      <DateTime>2005-12-17T09:30:47-05:00</DateTime>
      <Tag>
        <ID> urn:epc:id:sgtin:900100.456.989</ID>
      </Tag>
    </observation>
  </logEvent>
  <logEvent>
    <location>MarinaDelRay</location>
    <observation>
      <DateTime>2005-12-17T09:30:47-05:00</DateTime>
      <Tag>
        <ID> urn:epc:id:sgtin:900100.456.990</ID>
      </Tag>
    </observation>
  </logEvent>
</logEvents>
```

Operation returns status:  
e.g.  
<status>true</status>

## Observation Profile: Query

- Query EPC-IS for observations recorded at a specific location

```
<getEventsByLocation>  
  <location>urn:epc:id:gln:900100.7296</location>  
</getEventsByLocation>
```

Operation returns the full list of observation i.e. exactly the XML on the previous slide

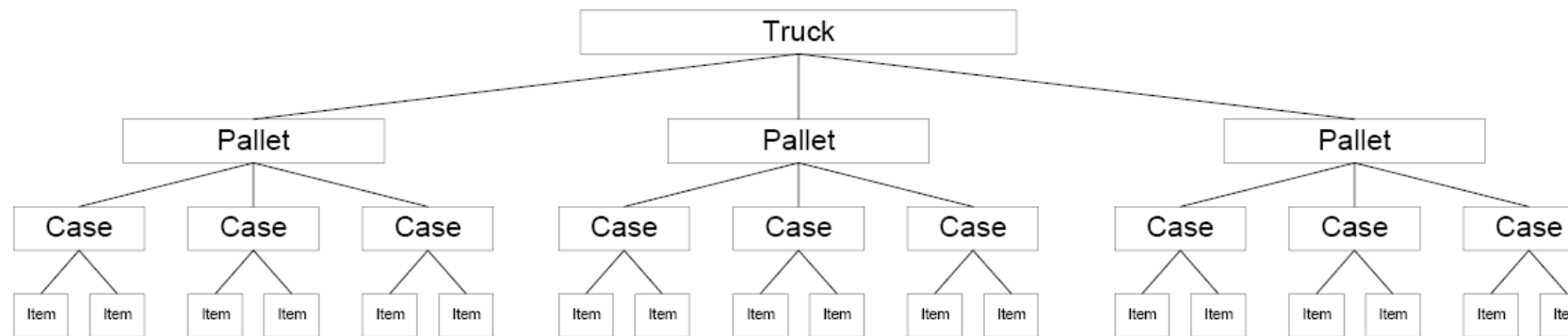
```
<logEvents>  
  <logEvent>  
    <location> urn:epc:id:gln:900100.7296</location>  
    <observation>  
      <DateTime>2005-12-17T09:30:47-05:00</DateTime>  
      <Tag>  
        <ID> urn:epc:id:sgtin:900100.456.989</ID>  
      </Tag>  
    </observation>  
  </logEvent>  
</logEvents>
```

## Observation Profile: Other Queries

- `logEvent(logEvents)`
  - Logs multiple observations
- `getEventsByLocation(location)`
  - Retrieves all observations logged at the specified location
- `getEventsByLocationByTimeRange(location, fromTime, toTime)`
  - Retrieves all observations logged at the specified location between two times
- `getEventsByTimeRange(fromTime, toTime)`
  - Retrieves all observations logged between two times
- `getEventsByEPC(epc)`
  - Retrieves all observations of the specified EPC
- `getEventsByEPCByTimeRange(epc, fromTime, toTime)`
  - Retrieves all observations of the specified EPC between two times
- `deleteEventsByLocationByTimeRange(location, fromTime, toTime)`
  - Deletes all observations made at a location between two times
- `deleteEventsByEPCByTimeRange(epc, fromTime, toTime)`
  - Deletes all observations of an EPC made between two times

# Containment Profile

- Aggregation into larger units



- The aggregation hierarchy is defined implicitly by specifying the items included in a container
- Both container and contents are specified using their respective EPC codes

## Containment Profile: Example

- Containment relationships are time sensitive i.e. they start at a specific time and have a specific end
- Relationships do not exist outside their defined time frames
- Thus, each method in the profile requires a time parameter

```
<setContents>
  <epc>urn:epc:id:sgtin:900100.456.870</epc>
  <time>2001-12-17T09:30:47</time>
  <epcList>
    <epc>urn:epc:id:sgtin:900100.456.871</epc>
    <epc>urn:epc:id:sgtin:900100.456.872</epc>
    <epc>urn:epc:id:sgtin:900100.456.873</epc>
  </epcList>
</setContents>
```

returns

```
<result>
  <status>true</status>
</result>
```

## Containment Profile: Example

- How to find the container of a particular object

```
<getContainer>
  <epc>urn:epc:id:sgtin:800100.432.987</epc>
  <time>2005-12-17T15:32:39</time>
</getContainer>
```

- If there is such a container, then

```
<epcList>
  <epc>urn:epc:id:sgtin:000200.100.900</epc>
</epcList>
```

- If there is not, then an error message

```
<result>
  <status>false</status>
</result>
```

# EPC IS Static Attribute Profile

- Ask for specific attributes of an object

A schema must be defined

```
<getAttributeData>
  <epc>urn:epc:sgtin:800900.321.123</epc>
  <schema>prodDetails</schema>
  <xpath>/NVP/Name[@id='color']/text()</xpath>
</getAttributeData>
```

XPath query

- Returns the requested data

```
<attribute>Black</attribute>
```

- And then change it

```
<setAttributeData>
  <epc>urn:epc:sgtin:800900.321.123</epc>
  <schema>prodDetails</schema>
  <xpath>/NVP/Name[@id='color']/text()</xpath>
  <value>Red</value>
</setAttributeData>
```



---

# Summary

---

- RFID Systems Architectures
- Middleware functionality and operational model
- API and examples
- EPCglobal services
- EPC Information Service
  - Profiles