RFID: Middleware and Web Services

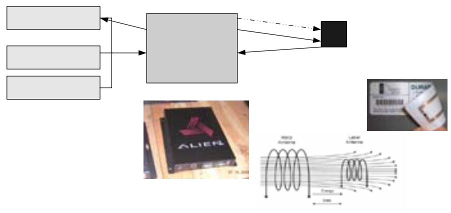
Mobile and Ubiquitous Computing

George Roussos
g.roussos@bbk.ac.uk

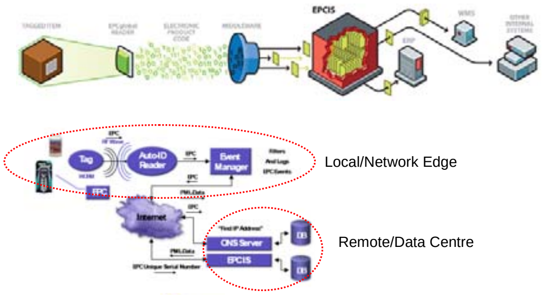
Overview

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

RFID Quick Recap



RFID Components



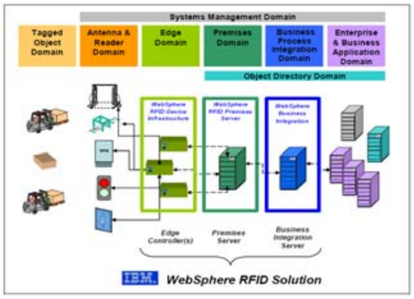
Local/Network Edge

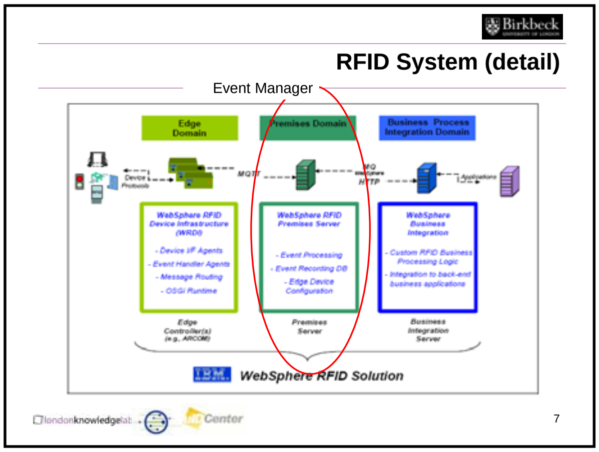
Remote/Data Centre

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

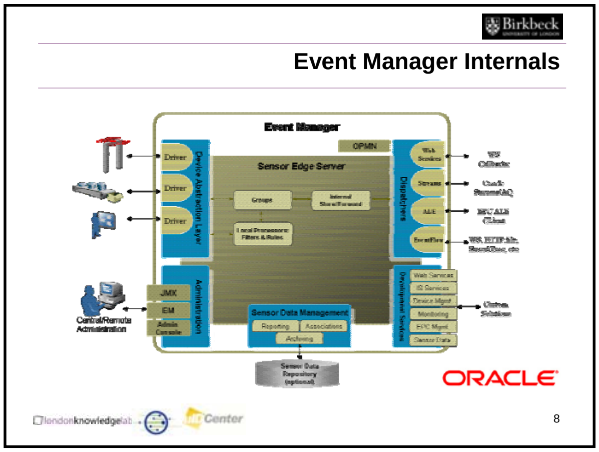
RFID System


Systems Management Domain
Tagged Object Domain
Antenna & Reader Domain
Edge Domain
Premises Domain
Business Process Integration Domain
Enterprise & Business Application Domain
Object Directory Domain
WebSphere RFID Device Infrastructure
WebSphere RFID Premises Server
WebSphere Business Integration
Edge Controller(s)
Premises Server
Business Integration Server
IBM WebSphere RFID Solution

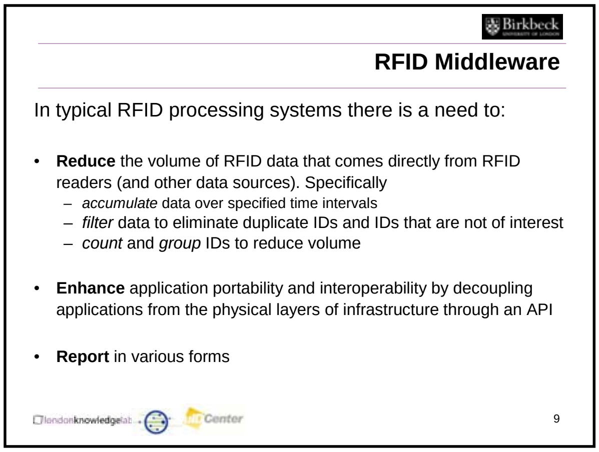


RFID System (detail)

Event Manager
Edge Domain
Premises Domain
Business Process Integration Domain
Device Protocols
MQTT
MQ WebSphere HTTP
Applications
WebSphere RFID Device Infrastructure (WRDI)
- Device I/F Agents
- Event Handler Agents
- Message Routing
- OSGi Runtime
WebSphere RFID Premises Server
- Event Processing
- Event Recording DB
- Edge Device Configuration
WebSphere Business Integration
- Custom RFID Business Processing Logic
- Integration to back-end business applications
Edge Controller(s) (e.g., ARCOM)
Premises Server
Business Integration Server
IBM WebSphere RFID Solution



Event Manager Internals

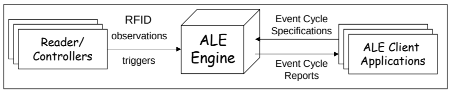


RFID Middleware

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

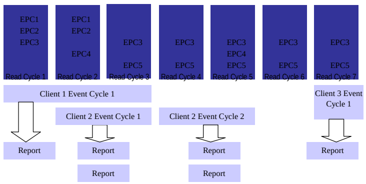
ALE Terminology

- *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*

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

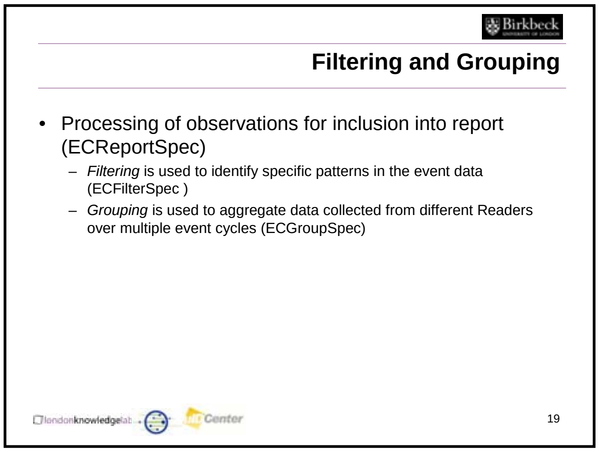
- Event Cycle Specification ECSpec
- `readers : List`
  - A logical reader i.e. any tag source (one or more readers with one or more antennas, also an EPC-enabled bar code reader)
  - For example a location (e.g. a dock door) where read events are captured for all physical readers attached to a location 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.

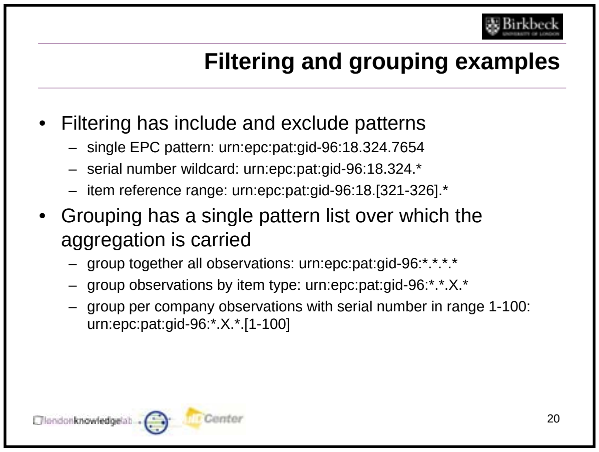
Event Cycle Boundary (cont.)

- 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.



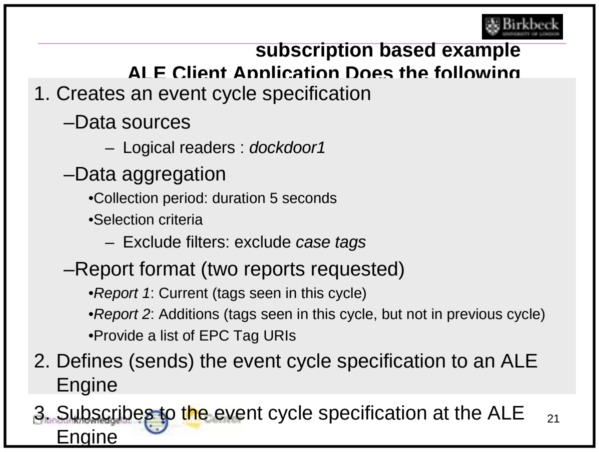
Filtering and Grouping

- 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)



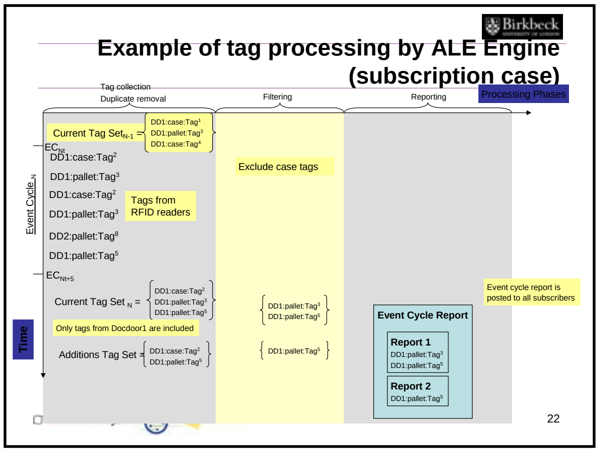
Filtering and grouping examples

- 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)

Processing Phases

Tag collection
Duplicate removal

Filtering

Reporting

Event Cycle$_N$

Time

Current Tag Set$_{N-1}$ = { DD1:case:Tag^1, DD1:pallet:Tag^3, DD1:case:Tag^4 }

EC$_{Nt}$

DD1:case:Tag^2

DD1:pallet:Tag^3

DD1:case:Tag^2

DD1:pallet:Tag^3

Tags from RFID readers

DD2:pallet:Tag^8

DD1:pallet:Tag^5

EC$_{Nt+5}$

Current Tag Set$_N$ = { DD1:case:Tag^2, DD1:pallet:Tag^3, DD1:pallet:Tag^5 }

Only tags from Docdoor1 are included

Additions Tag Set = { DD1:case:Tag^2, DD1:pallet:Tag^5 }

Exclude case tags

{ DD1:pallet:Tag^3, DD1:pallet:Tag^5 }

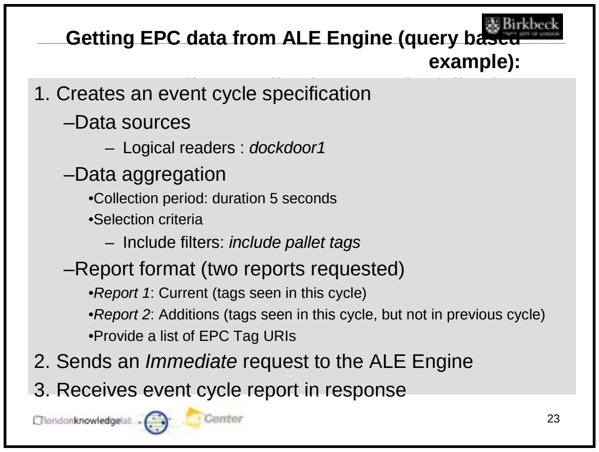
{ DD1:pallet:Tag^5 }

Event Cycle Report

Report 1
DD1:pallet:Tag^3
DD1:pallet:Tag^5

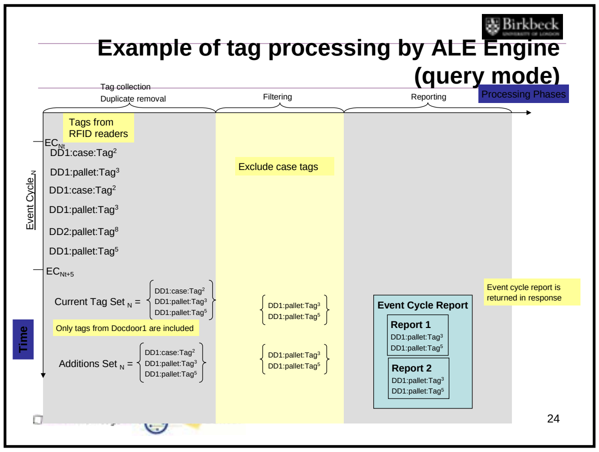
Report 2
DD1:pallet:Tag^5

Event cycle report is posted to all subscribers



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)

Processing Phases

Tag collection
Duplicate removal

Filtering

Reporting

Event Cycle$_N$

Time

Tags from RFID readers

EC$_{Nt}$

DD1:case:Tag^2

DD1:pallet:Tag^3

DD1:case:Tag^2

DD1:pallet:Tag^3

DD2:pallet:Tag^8

DD1:pallet:Tag^5

EC$_{Nt+5}$

Current Tag Set$_N$ = { DD1:case:Tag^2, DD1:pallet:Tag^3, DD1:pallet:Tag^5 }

Only tags from Docdoor1 are included

Additions Set$_N$ = { DD1:case:Tag^2, DD1:pallet:Tag^3, DD1:pallet:Tag^5 }

Exclude case tags

{ DD1:pallet:Tag^3, DD1:pallet:Tag^5 }

{ DD1:pallet:Tag^3, DD1:pallet:Tag^5 }

Event Cycle Report

Report 1
DD1:pallet:Tag^3
DD1:pallet:Tag^5

Report 2
DD1:pallet:Tag^3
DD1:pallet:Tag^5

Event cycle report is returned in response

Recap: EPCglobal NRFID architecture

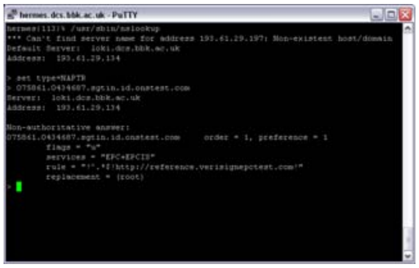
Discovery	Object Naming Service (ONS)	Discovery of authoritative object manufacturer information
	EPC Discovery Service	Track-and trace chain information discovery (pointers to)
Storage	EPC Information Service	Store and retrieve item and class level usage information
Authentication	EPC Trusted Services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

ONS Recap

Solaris 10 nslookup

Set DNS record type to NAPTR

ONS reply



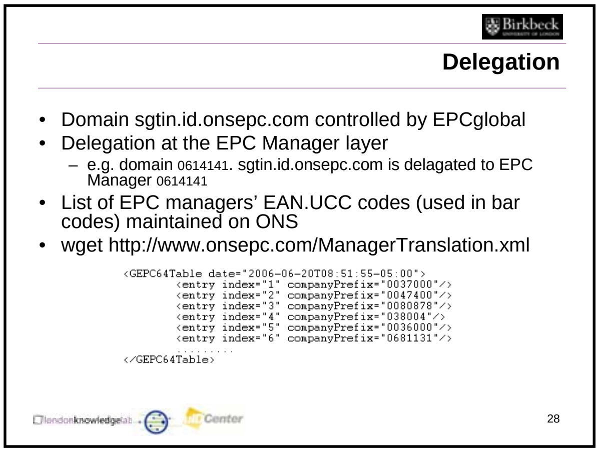
```
hermes(113)% /usr/sbin/nslookup
*** Can't find server name for address 193.61.29.197: Non-existent host/domain
Default Server:  loki.dcs.bbk.ac.uk
Address:  193.61.29.134

> 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

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>
```

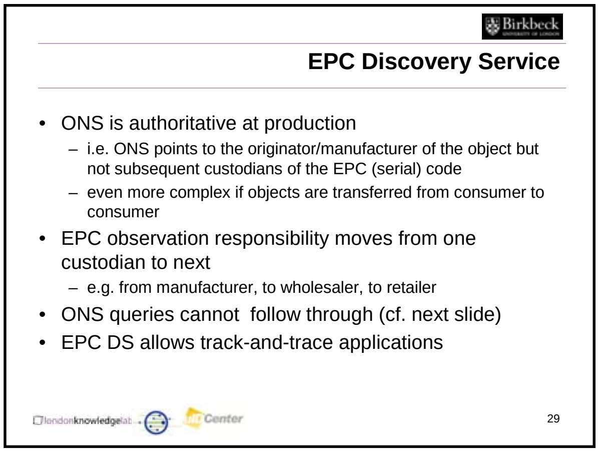
- EPC DS interfaces in flux
- Open source implementation of a possible solution with XML RPC
- Based on the eXist native XML engine and query XQuery processor available via **exist**.sourceforge.net



Delegation

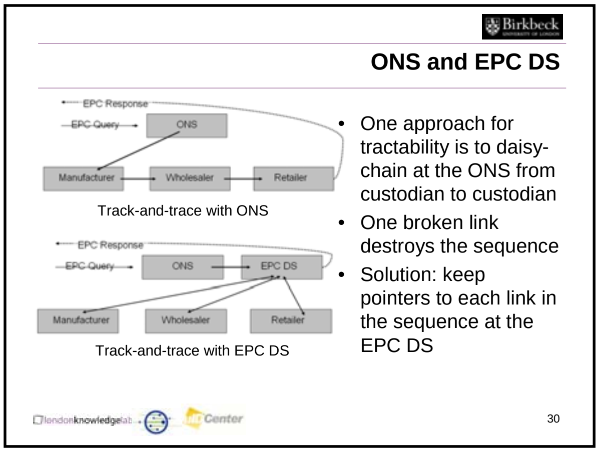
- Domain sgtin.id.onsepc.com controlled by EPCglobal
- Delegation at the EPC Manager layer
  - e.g. domain 0614141. sgtin.id.onsepc.com is delagated to EPC Manager 0614141
- List of EPC managers' EAN.UCC codes (used in bar codes) maintained on ONS
- wget http://www.onsepc.com/ManagerTranslation.xml

```
<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/manufacturer 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

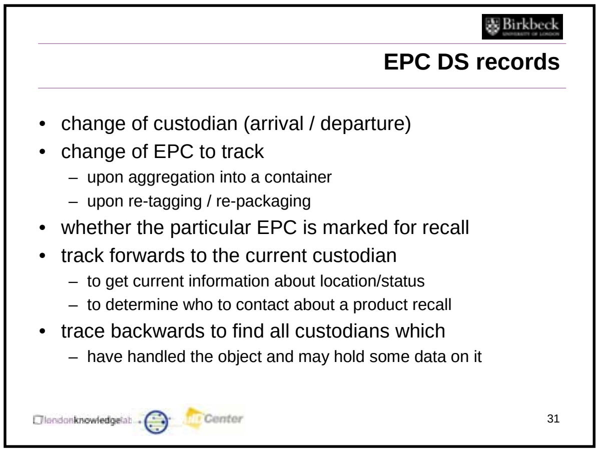


ONS and EPC DS

Track-and-trace with ONS

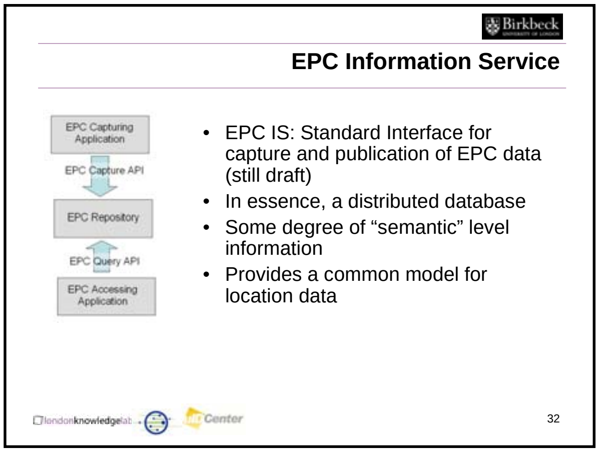
Track-and-trace with EPC DS

- 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

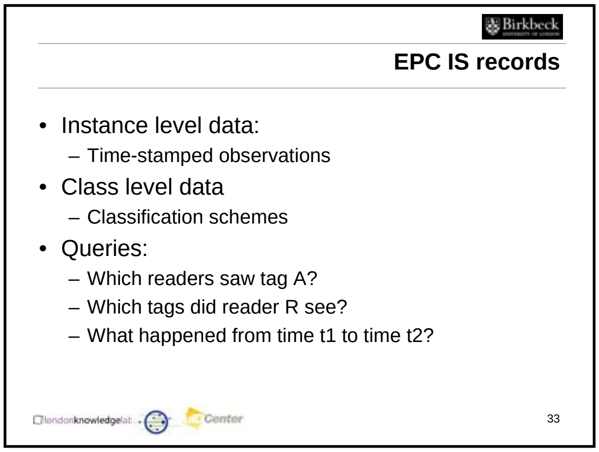
- change of custodian (arrival / departure)
- change of EPC to track
  - upon aggregation into a container
  - upon re-tagging / re-packaging
- whether the particular EPC is marked for recall
- track forwards to the current custodian
  - to get current information about location/status
  - to determine 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 Capturing Application
EPC Capture API
EPC Repository
EPC Query API
EPC Accessing Application

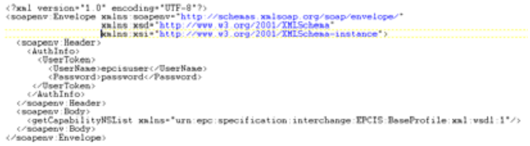
- EPC IS: Standard Interface for capture and publication of EPC data (still draft)
- In essence, a distributed database
- Some degree of “semantic” level information
- Provides a common model for location data



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 t1 to time t2?

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

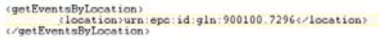
- Clients log observations in batch to the EPC-IS
- Same or different locations allowed
- Same or different observations allowed

```
<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

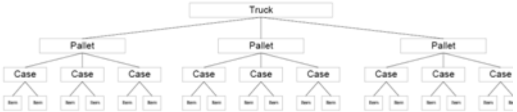
```
<logEvents>
  <logEvent>
    <location> urn:epc:id:gln:900100.7296</location>
    <observation>
      <DateTime>2005-12-17T09:30:47-05:00</DateTime>
      <Tag>
        <ID> urn:epc:id:sgtin:900100.456.989</ID>
      </Tag>
   </observation>
    </logEvent>
</logEvents>
```

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

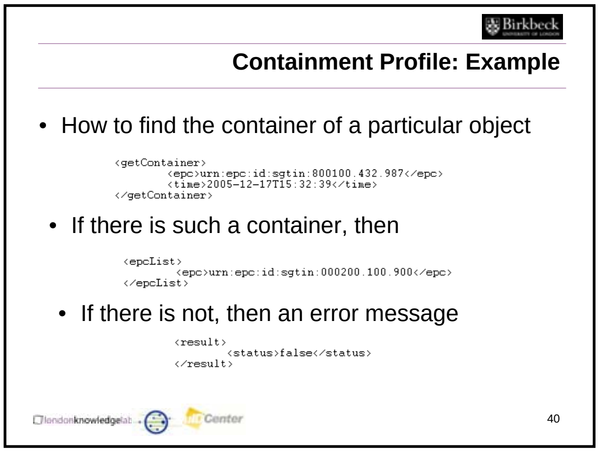
Containment Profile: Example

- 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

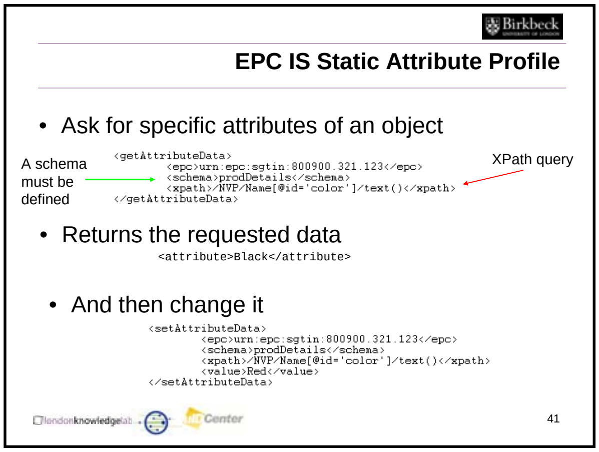
```
<getContainer>
        <epc>urn:epc:id:sgtin:800100.432.987</epc>
        <time>2005-12-17T15:32:39</time>
</getContainer>
```

- 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

XPath query

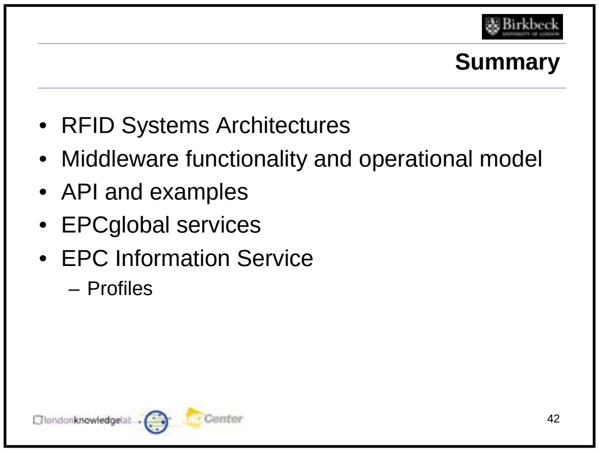
```
<getAttributeData>
        <epc>urn:epc:sgtin:800900.321.123</epc>
        <schema>prodDetails</schema>
        <xpath>/NVP/Name[@id='color']/text()</xpath>
</getAttributeData>
```

- 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