TENTH ANNUAL CONFERENCE AND EXHIBITION

APR. 3-5, 2012 | WALT DISNEY WORLD SWAN AND DOLPHIN RESORT, ORLANDO, FLA
RFID Data Capture Standards: LLRP and ALE

Kenneth R. Traub, PhD
Ken Traub Consulting LLC
3 April 2012
Outline for Today

• **11:30**: RFID Visibility Data for Business Applications
  – What’s the important data, and how do you use it?

• **12:15**: RFID Data Capture Software
  – How do you collect the important data?

• Lunch

• **1:45**: Putting It Together: Architecture, Product Selection and IT Governance
  – How do you build a complete system for the enterprise?

• **2:30**: RFID Tag-Data Standards

• **3:30 (now)**: RFID Data-Capture Standards: LLRP and ALE
About the Speaker

• Independent Consultant
• Specializing in EPC/RFID Standards adoption
  – Software architecture for enterprises and solution providers
  – Educational programs on standards tailored to clients’ needs
• Actively involved in GS1/EPC standards development
  – Member, GS1 Architecture Group
  – Editor, GS1 System Architecture
  – Editor, EPCIS and CBV specifications
  – Co-chair, Filtering & Collection (ALE) Working Group
  – Editor, EPC Tag Data Standard
  – Contributor to four other software specifications

• Consulting Instructor for Academia RFID
Visibility into the Physical World

• Business decisions are made here, in the company headquarters data center

• …but there’s an awful lot of important action here, in the real world.

⇒ Unique identification and RFID technology can bring awareness of the physical world.
Visibility Architecture

Single Most Important Design Decision: This Interface

Key “hinge” between different worlds
Data Capture Goal

• Data Capture goal: create visibility data for business applications
  – What, When, Where, Why
• Hide the details of “how” each process step occurs
• Design this data first, then create the right capture infrastructure

Visibility Event

What: urn:epc:id:sgtin:0400001.000001.2
When: 2007-10-02 10:00:00
Where: urn:epc:id:sgln:0400001.00300.0
Why: urn:epcglobal:cbv:bizstep:receiving
What Leads to Visibility Data?

- Asset Storage
  - Monitoring assets on a shelf, in a bin, etc
- Asset Flow
  - Entry/Exit through doors, choke points, etc
- Lifecycle Events
  - Manufacturing new products, product end-of-life, etc.
- Warehouse/Logistics Operations
  - Picking to order, packing, unpacking, etc
- Shipping/Receiving Operations
  - Verifying order against PO, ASN, etc
- In-Store Operations
  - Stocking, replenishment, point-of-sale, etc
Data Capture Architecture

What, When, Where, Why: completion of a business step
Hides “how” the step was carried out

Custom business logic that coordinates RFID, human data
entry, displays, queries to back-end systems, machines, to
carry out an operational process step.

What and When; and which logical readers
Hides reader make/model/quantity and other device details

Combine multiple readers, remove duplicates, filter unwanted tags,
accumulate over time, decode into meaningful representation,
report when trigger conditions met

Dozens of individual tag read events from specific antenna
ALE Goals

Custom business logic that coordinates RFID, human data entry, displays, queries to back-end systems, machines, to carry out an operational process step.

We want it to be easy to write!

Please give me:
- a report every 60 seconds
- with fully decoded EPC identifiers
- from the readers at loading dock #5
- only Acme products
- no item-level tags only what’s changed

Reduces volume of data from readers to applications
Elevates level of abstraction for application writers
Insulates applications from device details
Shares data among multiple applications
Extensible to vendor changes
Integrates easily using standard XML / Web Services technology

Individual tag reads,
several times / second for every tag that is in range
Things that might implement ALE

- Middleware:
  - Software system that interfaces to readers and other devices via network protocols
  - Exposes ALE to other software applications that wish to read and write tags
- Concentrator:
  - Similar to middleware, but embedded in a network appliance
- “Smart” Reader:
  - Embedded ALE implementation provides a high-level interface for applications that want to interact with that reader
- Printer:
  - Similar to smart reader

- Most of these are commercially available today (ALE 1.0)
ALE Current Status

- ALE 1.0 ratified September 2005
  - 22 products certified as of April 2012 (many more on market)
- ALE 1.1 ratified February 2008
  - 7 products certified as of April 2012
  - full support for UHF Class 1 Gen 2 / ISO 18000-6C features: memory banks, kill/lock, etc
  - support for ISO 15962 encoding (encompasses EPCglobal Tag Data Standards for user memory)
  - new API for writing tags and doing other operations
  - new API for defining named memory fields
  - new API for configuring logical to physical reader mappings
  - security features
ALE APIs

- **Reading API**
  - Reads tags, reports in variety of ways

- **Writing API**
  - Initialize, read, write, lock, kill

- **Tag Memory API**
  - Define symbolic names for memory fields, for use by Reading & Writing APIs

- **Logical Reader API**
  - Define symbolic names for reader/device resources, for use by Reading & Writing APIs

- **Access Control API**
  - Control access by clients to other API features

Primarily used by applications (data plane)

Primarily used for setup and administration (control plane)
Reading and Writing APIs – Principles

• ALE specifies an interface
  – ALE Client – application or other system component that wants to operate upon Tags
  – ALE Implementation – system component that implements the ALE APIs, and carries out client requests by interacting with readers or other devices (or it may be embedded in a reader itself)
  – The design of an ALE implementation is outside the scope of the spec

• ALE is declarative
  – ALE Client says what it wants done
  – ALE Implementation figures out how best to carry out that request
  – ALE Implementation has great freedom to push processing down to the reader or even the tag, to combine simultaneous requests, and otherwise optimize the use of resources

• ALE interface centers around “specs” and “reports”
  – Event Cycle Spec (ECSpec) = ALE Client request in Reading API
  – Event Cycle Report (ECReport) = ALE Implementation’s response
  – Command Cycle Spec / Report (CCSpec & CCReport) = corresponding things in Writing API
How it works (Reading API)

<table>
<thead>
<tr>
<th>ALE Client’s Declarative Specification (ECSpec)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What Locations</strong> (&quot;logical readers&quot;)</td>
</tr>
<tr>
<td>e.g. Dock Door 22 or Shelf 14</td>
</tr>
<tr>
<td><strong>Time Interval for Accumulating Reads</strong></td>
</tr>
<tr>
<td>Start: Continuous, Interval or Trigger</td>
</tr>
<tr>
<td>Stop: Duration, Stable Field, Trigger, or</td>
</tr>
<tr>
<td>When New Data Available</td>
</tr>
<tr>
<td><strong>What to Read:</strong></td>
</tr>
<tr>
<td>Which Tags</td>
</tr>
<tr>
<td>What Data Fields</td>
</tr>
<tr>
<td>How to Report</td>
</tr>
<tr>
<td>Report Set: Current, Additions, or Deletions</td>
</tr>
<tr>
<td>Filters: Include and Exclude Patterns</td>
</tr>
<tr>
<td>Grouping: Grouping Patterns</td>
</tr>
<tr>
<td>Output: List of tags and contents of specific fields,</td>
</tr>
<tr>
<td>Count of tags,</td>
</tr>
<tr>
<td>or Both</td>
</tr>
</tbody>
</table>

Client presents to ALE API in one of two modes
Request Modes

Subscribe Mode:
Asynchronous ("push") reports from a standing request (ECSpec)

Most often used for:
- Continuous operation; or
- Triggered by time, external events
Request Modes

Poll Mode:
Synchronous (on-demand, “pull”) report from a standing request

Most often used for operations triggered programmatically (e.g., GUI-driven)
Example #1

Smart Shelf: “Report once per minute about things added and removed from Shelf #123”
“Report once per minute about things added and removed from Shelf #123”

<ale:ECSpec …>
    <logicalReaders>
        <logicalReader>shelf123</logicalReader>
    </logicalReaders>
    <boundaries>
        <duration unit="MS">60000</duration>
    </boundaries>
    <reportSpecs>
        <reportSpec name="New Stuff">
            <reportSet set="ADDITIONS"/>
            <output includeEPC="true"/>
        </reportSpec>
        <reportSpec name="Stuff Taken Away">
            <reportSet set="DELETIONS"/>
            <output includeEPC="true"/>
        </reportSpec>
    </reportSpecs>
</ale:ECSpec>
Example #1 (cont’d)

Event Cycle Specification (ECSpec)

ALE Implementation

Event Cycle Report (ECReport)

Header

Timestamp = 2007-07-06T13:11:00Z
Time = 60032 ms

Output

Report “New Stuff”:
Tag #1:
  EPC = urn:epc:id:sgtin:0614141.112345.3
Tag #2:
  EPC = urn:epc:id:sgtin:0614141.112345.4

Report “Stuff Taken Away”:
Tag #1:
  EPC = urn:epc:id:sgtin:0614141.112345.5
ECReports in XML

<ale:ECReports ...
 <reports>
     <report name="New Stuff">
         <group>
             <groupList>
                 <member>
                     <epc>urn:epc:id:sgtin:0614141.112345.3</epc>
                 </member>
                 <member>
                     <epc>urn:epc:id:sgtin:0614141.112345.4</epc>
                 </member>
             </groupList>
         </group>
     </report>
     <report name="Stuff Taken Away">
         <group>
             <groupList>
                 <member>
                     <epc>urn:epc:id:sgtin:0614141.112345.5</epc>
                 </member>
             </groupList>
         </group>
     </report>
 </reports>
</ale:ECReports>
# ALE Benefits

<table>
<thead>
<tr>
<th>With ALE</th>
<th>Without ALE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe what data you want; ALE responds</td>
<td>Write code to command each reader explicitly</td>
</tr>
<tr>
<td>Easy to change make, model, and number of readers</td>
<td>Rework reader interface code</td>
</tr>
<tr>
<td>Combine data from multiple readers with no code changes</td>
<td>Application must combine data from multiple readers; defeats each reader’s filtering</td>
</tr>
<tr>
<td>Application “tuning” by changing EC Spec</td>
<td>Tuning may be buried deep in application code</td>
</tr>
<tr>
<td>Tag contents decoded (e.g., into EPC URIs)</td>
<td>Application must interpret raw hexadecimal content</td>
</tr>
<tr>
<td>Standards-based XML Web Services interface</td>
<td>Proprietary wire protocol for each reader, usually not XML</td>
</tr>
<tr>
<td>Standards based; easy to switch vendors</td>
<td>Possible lock-in to reader vendor SDK</td>
</tr>
</tbody>
</table>
ALE vs LLRP

Please give me:
- a report every 60 seconds
- with fully decoded EPC identifiers
- from the readers at loading dock #5
- only Acme products
- no item-level tags only what’s changed

Individual tag reads,
several times / second for every tag that is in range

Full control over reader operation
Why Care about LLRP?

• If you’re using middleware (ALE or otherwise):
  – Avoid dependence on whether a given reader’s proprietary interface is supported
  – Predictable reader interaction

• If you’re not, because:
  – You need very detailed control over how the tag and reader interact
  – Your overall software application must be extremely small and lightweight
## ALE vs LLRP

<table>
<thead>
<tr>
<th>ALE</th>
<th>LLRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface to one or more readers</td>
<td>Interface to a single reader</td>
</tr>
<tr>
<td>Many clients may share same readers</td>
<td>Only one client may control reader</td>
</tr>
<tr>
<td>Reader’s identity may be hidden behind a “logical reader name”</td>
<td>Client knows reader’s identity, make, model</td>
</tr>
<tr>
<td>Details of reader/tag interaction are hidden</td>
<td>Full control over reader/tag interaction details, if desired</td>
</tr>
<tr>
<td>Data encoded/ decoded according to Tag Data Standards, ISO 15962</td>
<td>Raw binary data input and output</td>
</tr>
<tr>
<td>XML/SOAP wire protocol</td>
<td>Binary wire protocol</td>
</tr>
<tr>
<td>Suitable starting point for application business logic (local workflow)</td>
<td>Suitable starting point for middleware controller of reader network</td>
</tr>
</tbody>
</table>
LLRP Fundamentals

- One **LLRP Client** talks to One **Reader**
- Reader has 1 or more **Antennas**
- Each Antenna can interact with tags via 1 or more **Protocols**

- LLRP is declarative
  - LLRP Client says what it wants done
  - LLRP implementation carries this out autonomously, reports back on-demand or asynchronously

- LLRP interface centers around “specs” and “reports”
  - Reader Operation Spec (ROSpec) = how to acquire tags
  - Access Spec = what to do with acquired tags beyond reading EPC
  - Report = output from LLRP to client saying what happened with tags
  - Event = output from LLRP to client reporting on LLRP lifecycle
  - Basic application uses only ROSpec and Report
LLRP Operation Timeline

- **Antenna Inventory**
  - Antenna = 1
  - Protocol = Gen2

- **RFSurvey**
  - Channels 5 – 10

- **Antenna Inventory**
  - Antenna = 2
  - Protocol = Gen2

- **Antenna Inv.**
  - Antenna = 1
  - Protocol = Gen1

- **Tag ID (EPC) is read**
- **Singulate**
- **Tag**
- **Tag**
- **Tag**
- **Tag**
- **Tag**
- **Tag**
- **Tag**
- **Tag**
- **Tag**
- **Tag**
- **Tag**

- **Write, Kill, Lock, user memory read, etc**

---

Ken Traub Consulting LLC
Developing with LLRP

• LLRP is a binary wire protocol
• Free development kits are available:
  – www.llrp.org

• Confirm LLRP support with your reader vendor
Summary

• Standards simplify RFID-based data capture:
  – Avoid vendor lock-in
  – Easier development
  – Community of resources

• ALE: best starting point for application business logic

• LLRP: low-level interface when you need full control
Thank You