RFID JOURNAL LIVE!

TRACK AND MANAGE **EVERYTHING**

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

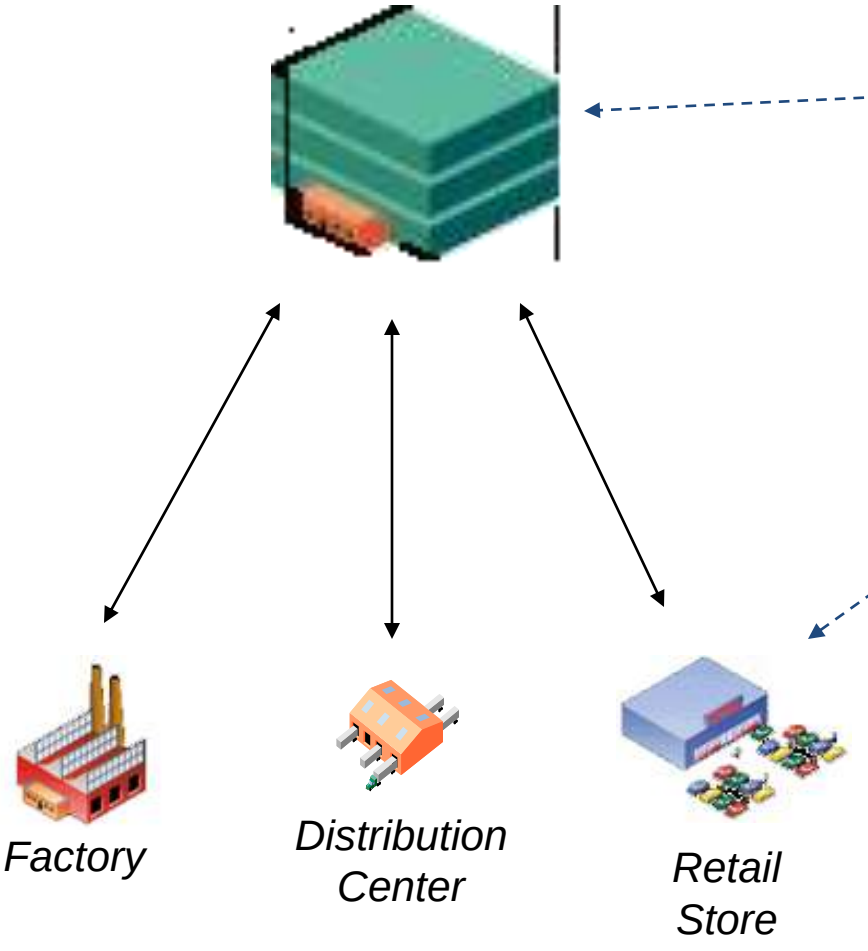
- 11:30: RFID Visibility Data for Business Applications
  - What's the important data, and how do you use it?
- 12:15: RFID Data Capture Software
  - How do you collect the important data?
- Lunch
- 1:45: Putting It Together: Architecture, Product Selection and IT Governance
  - How do you build a complete 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



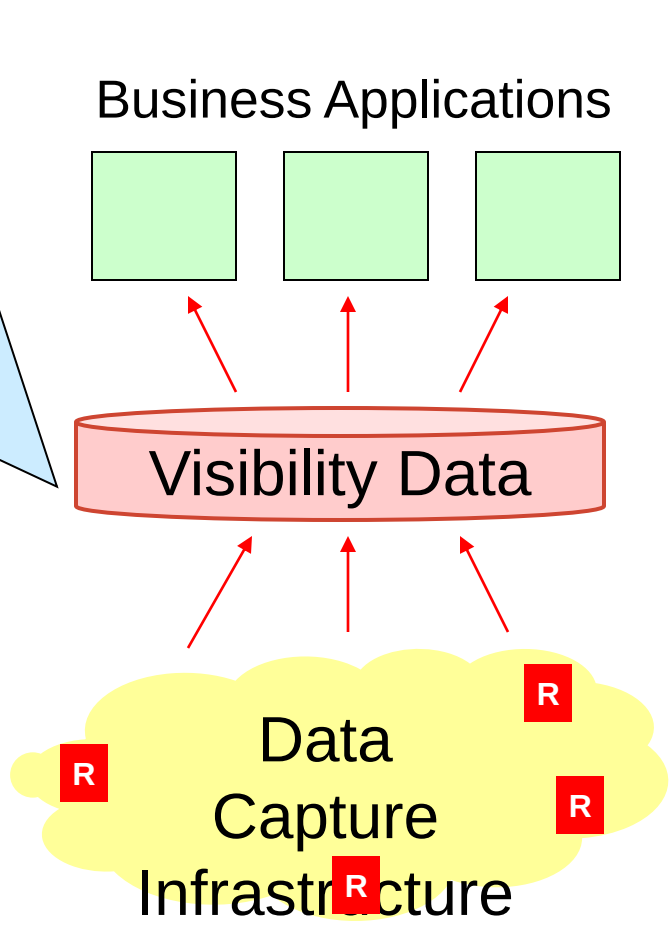
- Business decisions are made here, in the company headquarters data center
- …but there's an awful lot of important action here, in the real world.

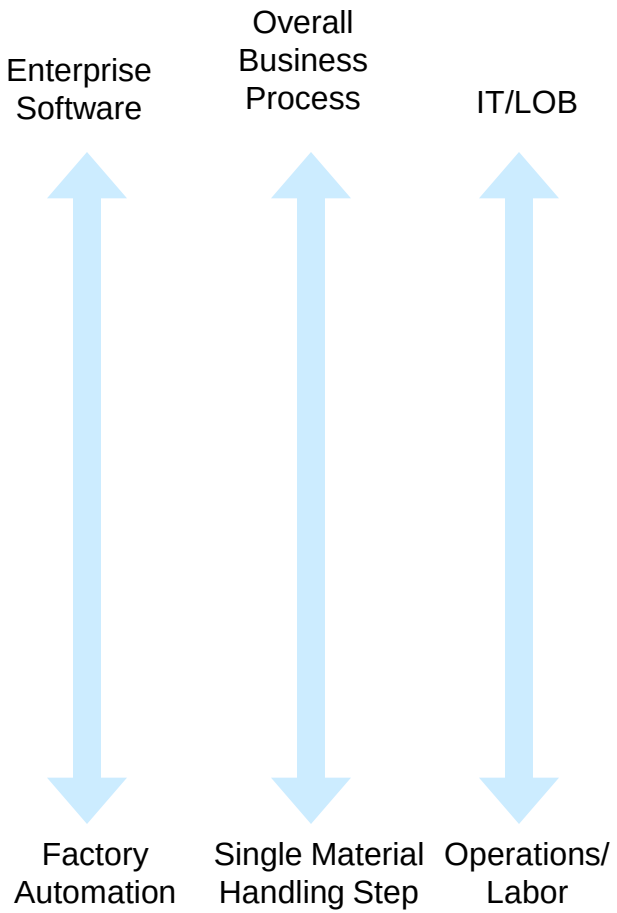
➔ Unique identification and RFID technology can bring awareness of the physical world

Visibility Architecture

Single Most Important Design Decision: **This Interface**

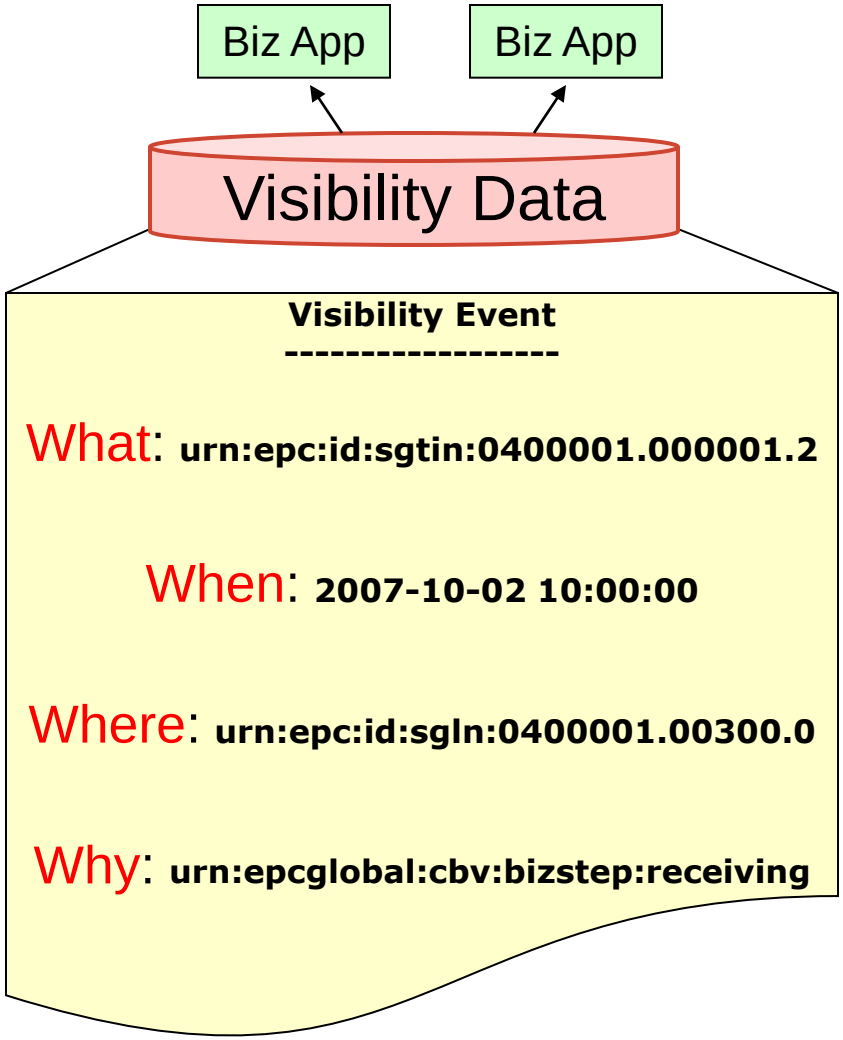
Key “hinge” between different worlds





Data Capture Goal

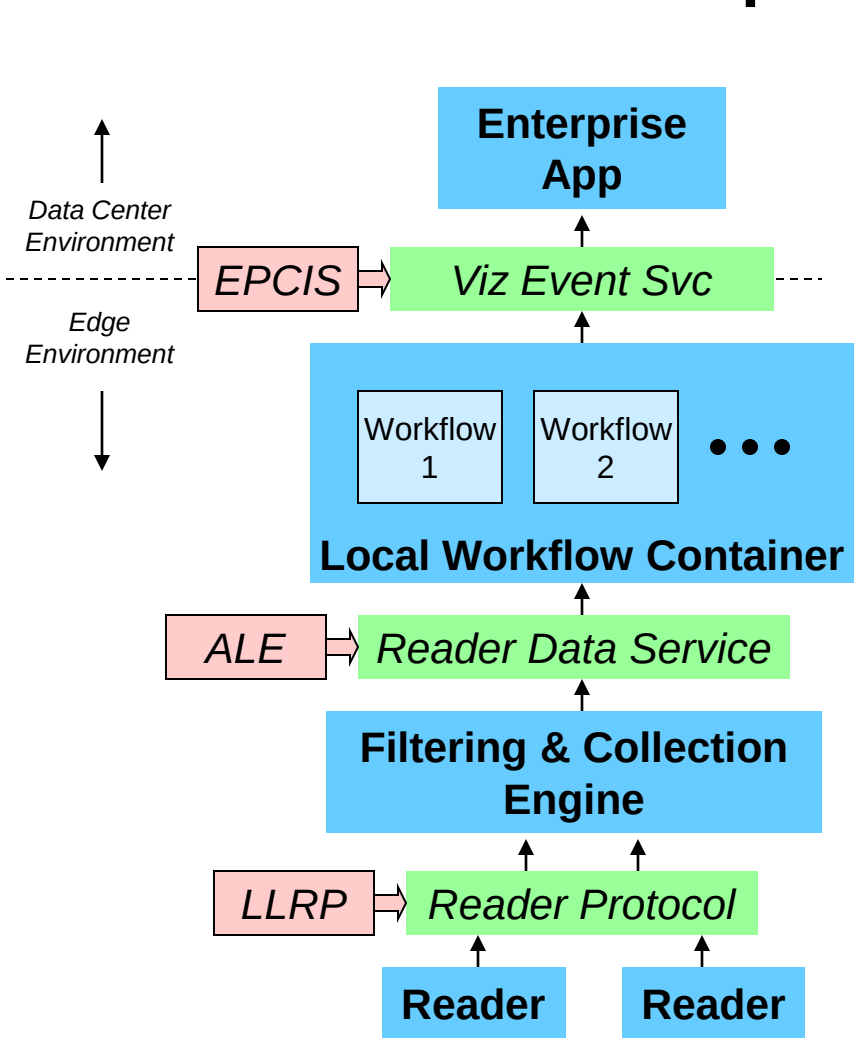
- Data Capture goal: create visibility data for business applications
  - What, When, Where, Why
- Hide the details of "how" each process step occurs
- Design this data first, then create the right capture infrastructure



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

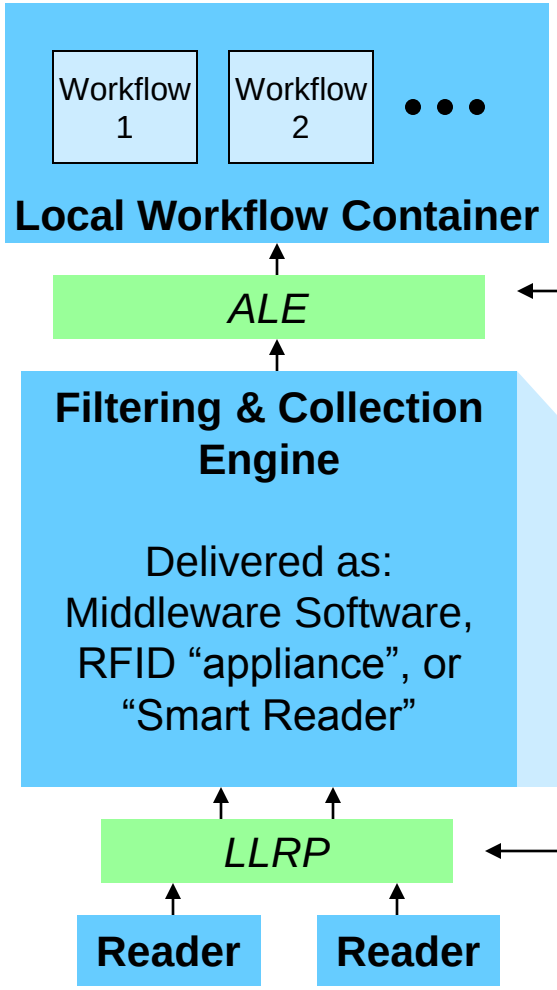
Custom business logic that coordinates RFID, human data entry, displays, queries to back-end systems, machines, to carry out an operational process step.

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

Primarily used by applications (data plane)

- 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 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)

ALE Client's Declarative Specification (ECSpec)

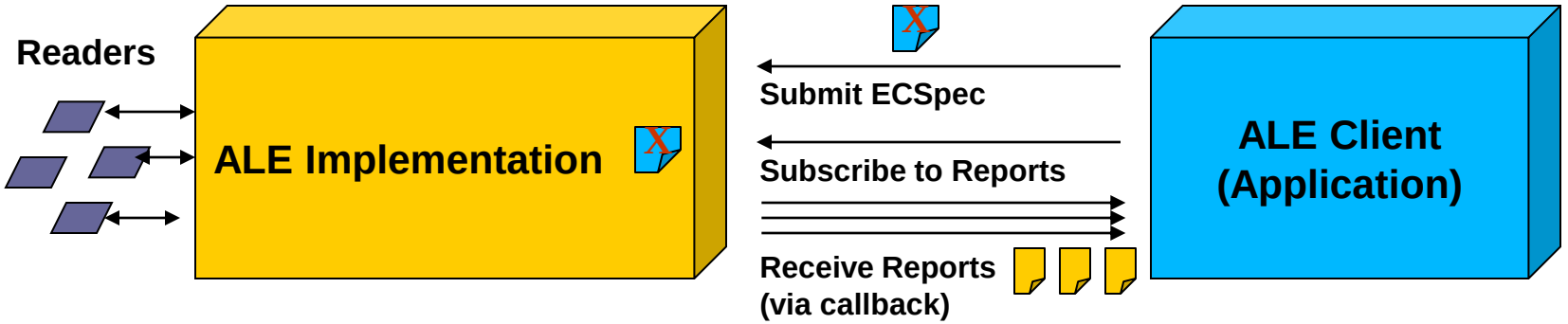
What Locations ("logical readers")	e.g. Dock Door 22 or Shelf 14
Time Interval for Accumulating Reads	Start: Continuous, Interval or Trigger Stop: Duration, Stable Field, Trigger, or When New Data Available
What to Read: Which Tags What Data Fields How to Report	Report Set: Current, Additions, or Deletions Filters: Include and Exclude Patterns Grouping: Grouping Patterns Output: List of tags and contents of specific fields, Count of tags, or Both

Client presents to ALE API in one of two modes

Request Modes

Subscribe Mode:

Asynchronous (“push”) reports from a standing request (ECSpec)



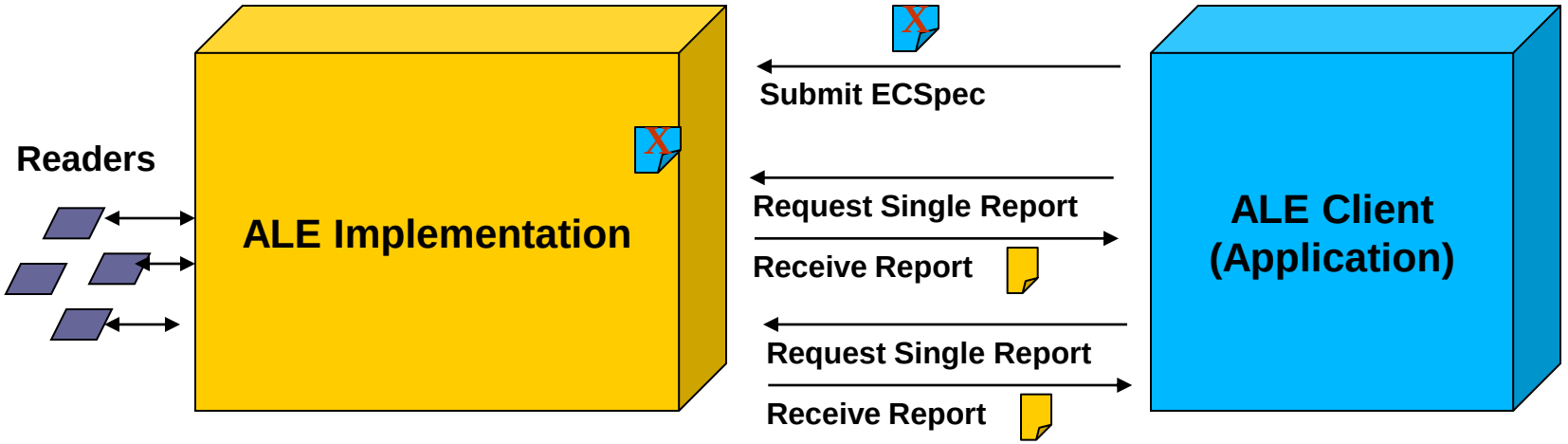
Most often used for:

-Continuous operation; or

-Triggered by time, external events

Request Modes

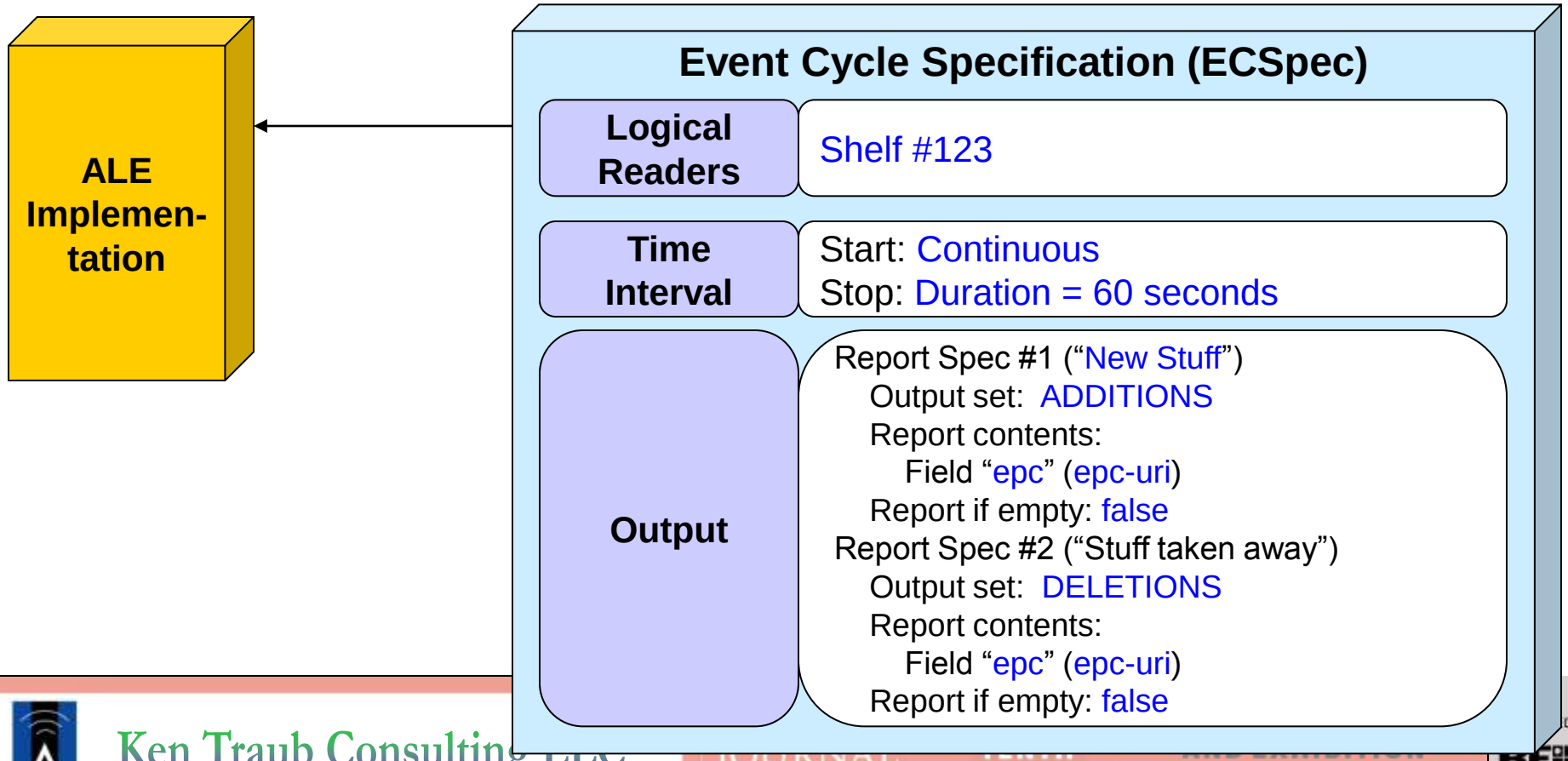
Poll Mode:
Synchronous (on-demand, “pull”) report from a standing request



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”

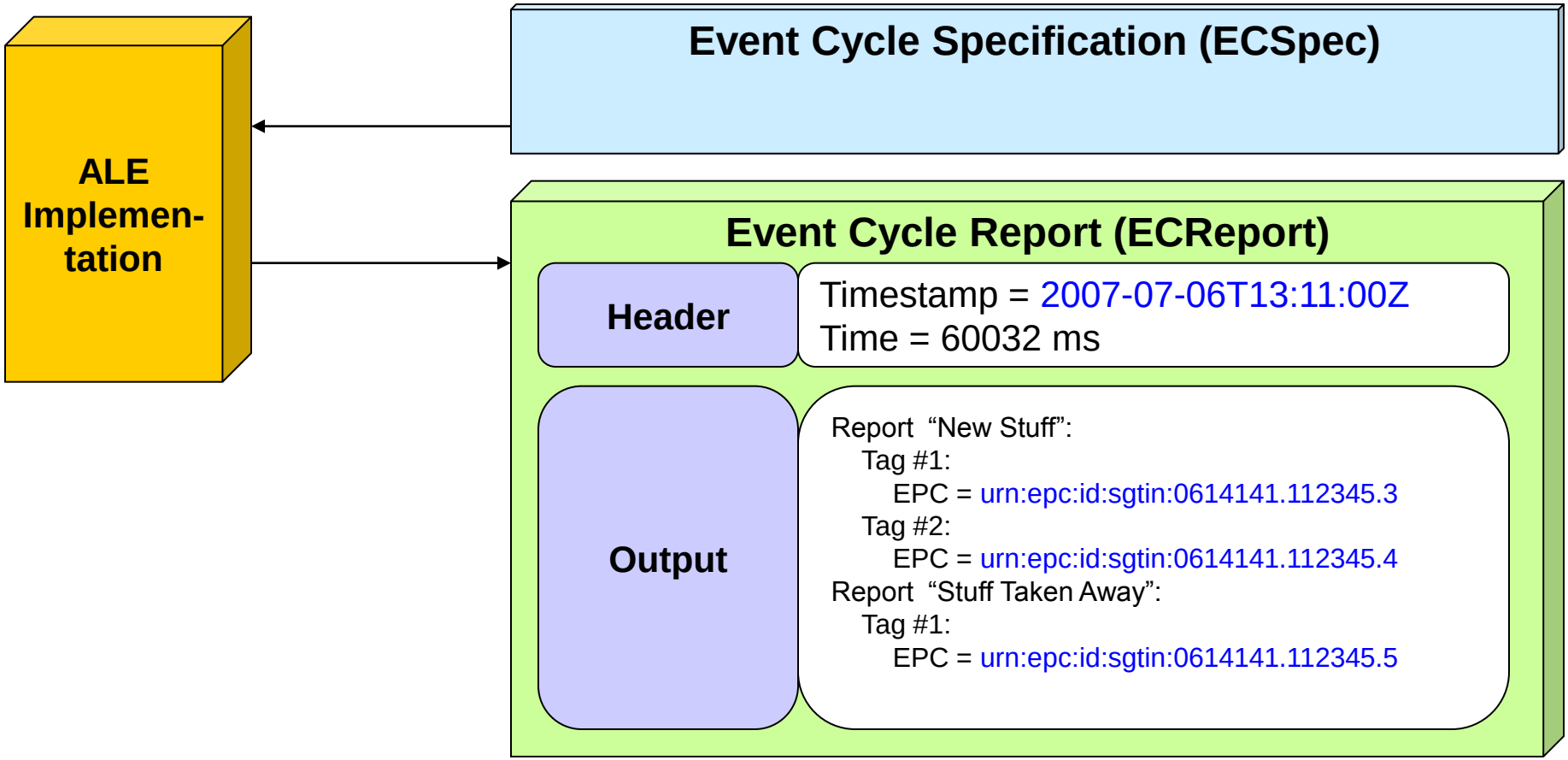


ECSpec in XML

"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)



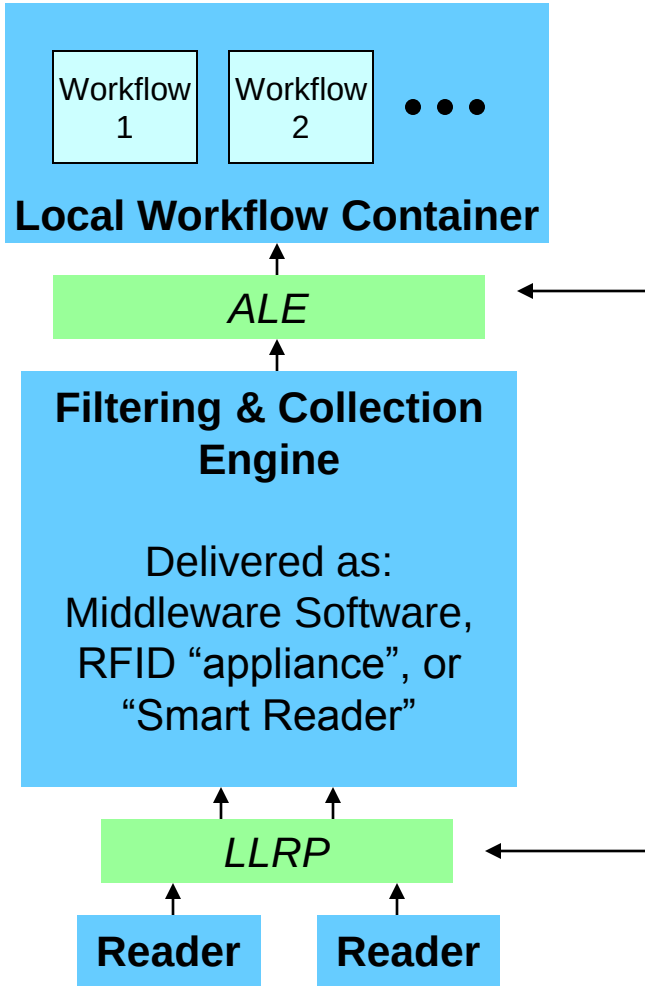
ECReports in XML

```
<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>
</ale:ECReports>
```

ALE Benefits

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

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

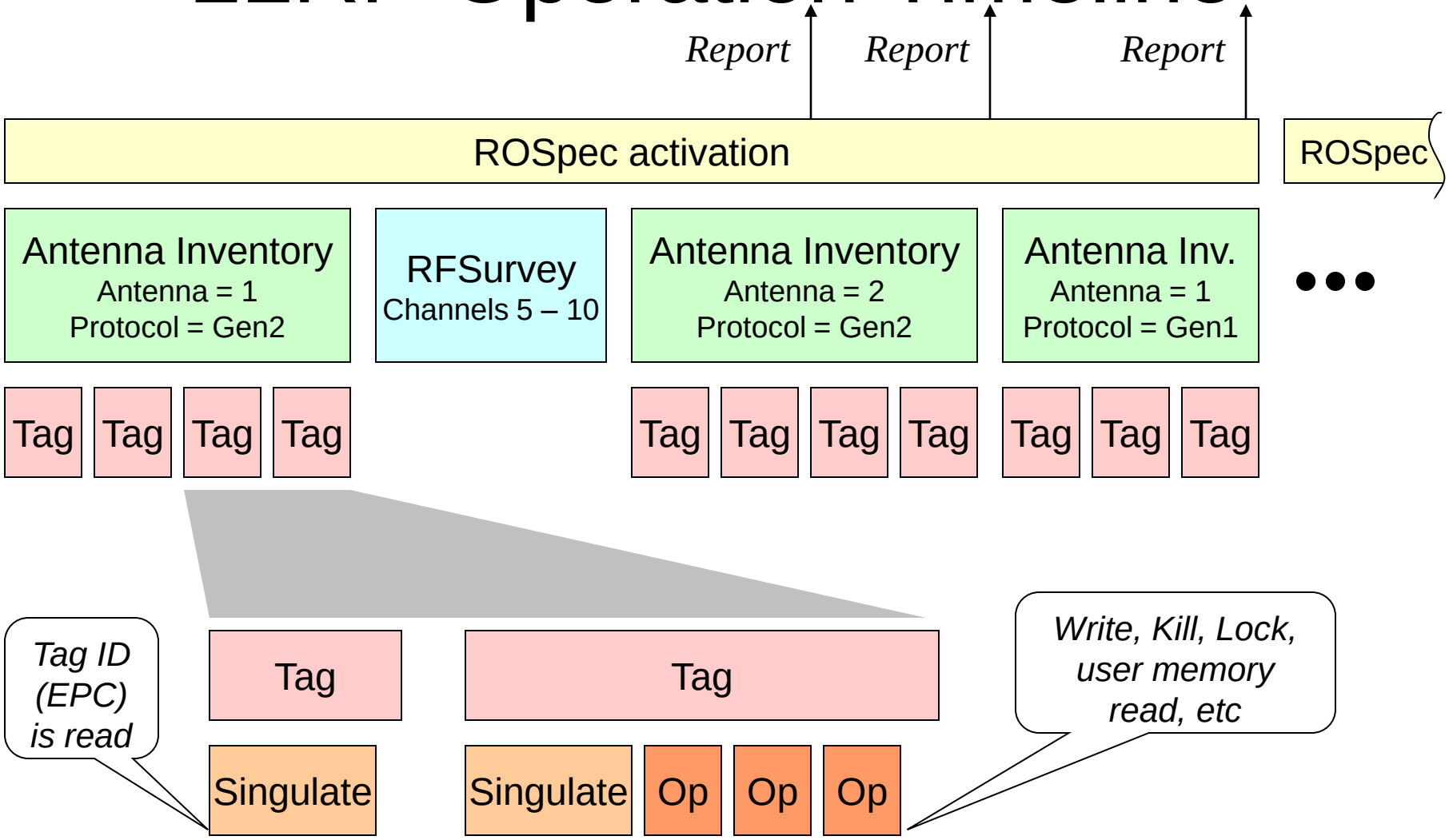
ALE	LLRP
Interface to one or more readers	Interface to a single reader
Many clients may share same readers	Only one client may control reader
Reader's identity may be hidden behind a "logical reader name"	Client knows reader's identity, make, model
Details of reader/tag interaction are hidden	Full control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

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



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