Application Level Events

Mobicom 2006 tutorial
Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

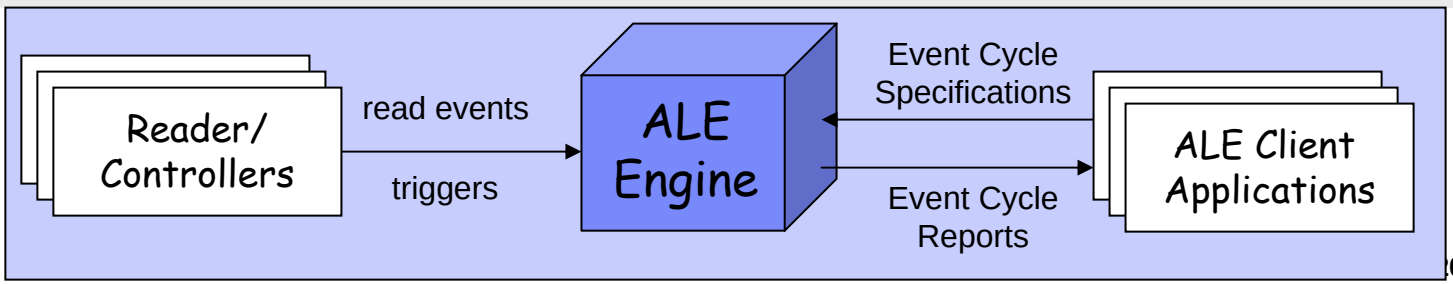
Overview

- ALE functionality and operational model
- ALE core concepts
  - readers, cycles, events, reports
- ALE core objects
  - ECSpec and ECBoundarySpec
- Application programming interface
- Examples (subscription and query)
- Future versions

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

ALE: Why? What?

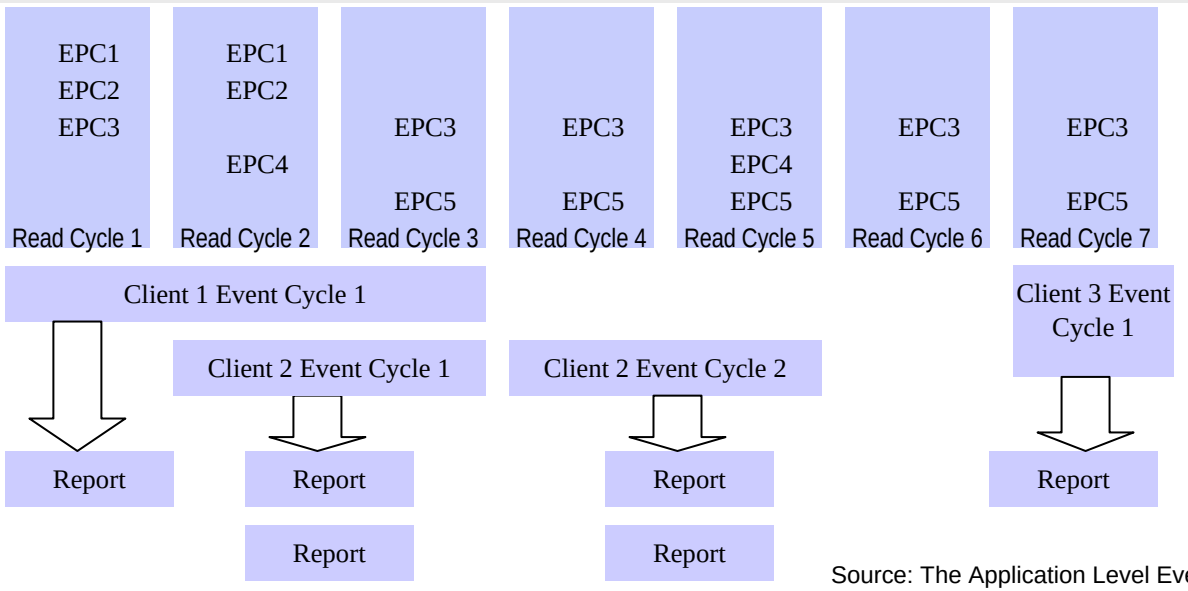
- In typical EPC processing systems there is a need for:
  - Reducing the volume of EPC data that comes directly from EPC data sources such as RFID readers
  - Enhancing application portability and interoperability by decoupling applications from the physical layers of infrastructure through an abstract interface
- EPCglobal Application Level Events (ALE) specification
  - **Provides a means for clients to specify, in a high-level, declarative way, what EPC data they are interested in, without dictating an implementation**
    - Abstract application programming interface: to get EPC data from ALE Engine
    - SOAP bindings maps abstract API to Web service implementation
    - **Does not specify how ALE interfaces with EPC data sources or trigger sources**
    - Formal processing model
- ALE Engine processes EPC data coming from readers against clients' event cycle specifications as follows:
  - *Receives* EPCs from one or more data sources such as RFID readers
  - *Accumulates* data over intervals of time
    - *filters* to eliminate duplicate EPCs and EPCs that are not of interest
    - *counts* and *groups* EPCs to reduce the volume of data
  - *Reports* in various forms



Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

Some ALE Terminology

- Reader: source of raw EPC data
  - Examples: 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
- Event cycle: smallest unit of interaction between client and ALE interface; it may consist of one or more read cycles
- Report: data about event cycle communicated from ALE implementation to a client
- Event cycle boundaries may:
  - Extend for a specified interval of real time; e.g., accumulate reads into five-second intervals.
  - Occur periodically; e.g., report only every 30 minutes, regardless of the read cycle.
  - 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.
  - Be delimited when no new EPCs are detected by any Reader specified for that event cycle for a specified interval of time.



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

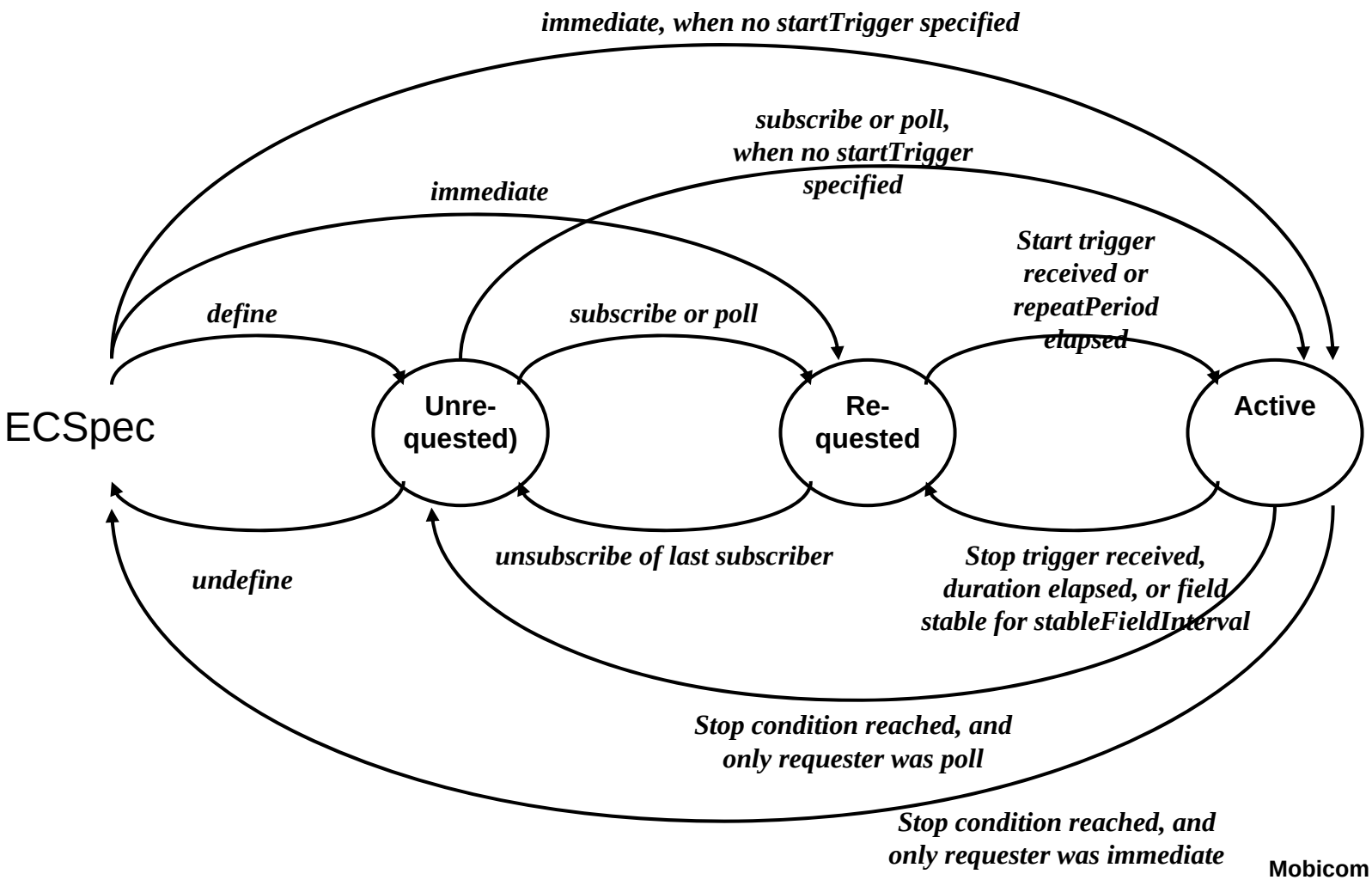
Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

ECSPEC: Event Cycle Specification

- **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

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

ECSpec State Diagram



Mobicom 2006 tutorial
Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

ECBoundarySpec

- 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.
- 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.

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

ECBoundarySpec

- **startTrigger : ECTrigger**
  - A URL to be posted when you wish to start an ECSpec
  - The start trigger in the spec would be specified as:
    http://host:port/StartTrigger?triggerId=xxx
    where xxx = your trigger id for example: “specname_readername”
  - When you are ready to start the ECSPEC your application or browser would issue:
    “http://host:port/AleHttpAdapters/StartTrigger?triggerId=xxx”
    where host and port is the name of the host WAS system that has the engine installed.
- **repeatPeriod : ECTime**
  - Is a long non-negative time value (in ms)
  - Once an ECSPEC is in requested state, it immediately becomes active and remains active for this time;
- **stopTrigger : ECTrigger**
  - URL to be posted when you wish to stop an ECSpec
  - Same definition as startTrigger except using “StopTrigger”
- **duration : ECTime**
  - Millisecond long time value
  - At the end of this duration, end the event cycle
- **stableSetInterval : ECTime**

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

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 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].*
  - full wildcard: urn:epc:pat:gid-96:*.*.*
- 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]

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

ALE Engine Core APIs 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

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

ALE Engine Core APIs 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

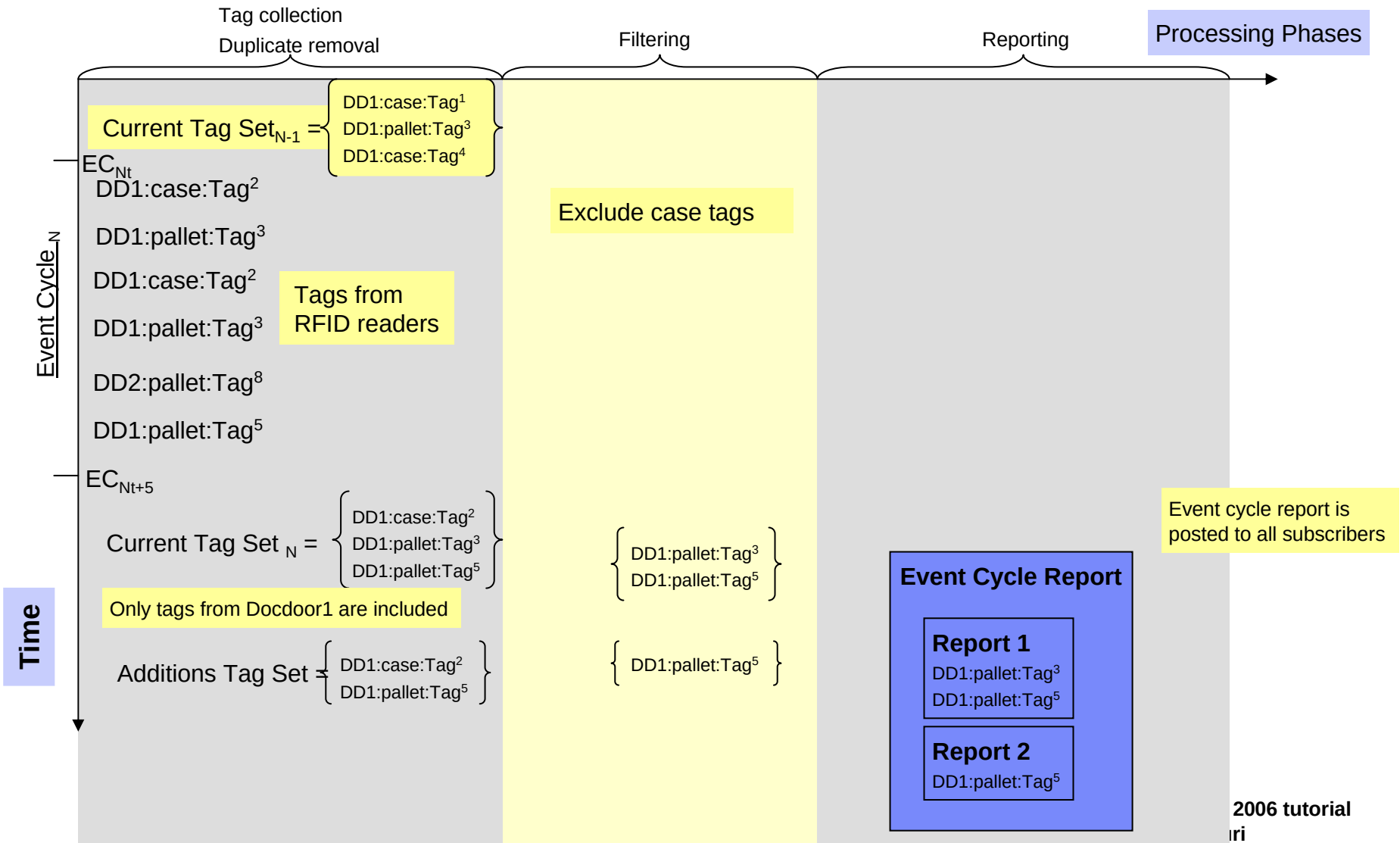
Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

Getting EPC data from ALE Engine (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
4. Receives event cycle reports and extracts EPC data
5. Unsubscribe to event cycle specification when reports are no longer needed
6. Undefine event cycle specification when it is no longer needed

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

Example of tag processing by ALE Engine (subscription case)



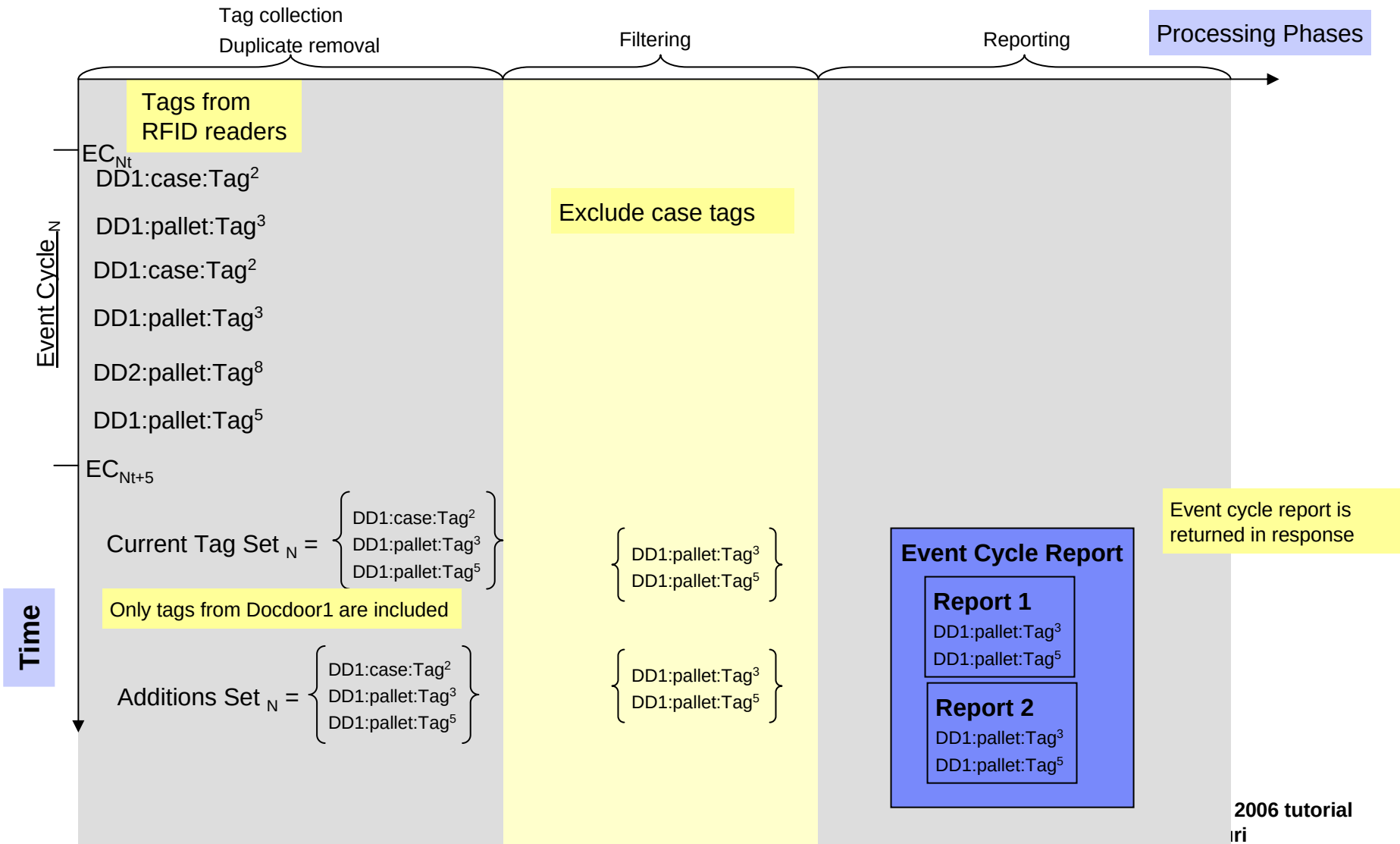
2006 tutorial
ri
sastry@us.ibm.com
IBM Research, TJ Watson

Getting EPC data from ALE Engine (query 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
       - 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

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

Example of tag processing by ALE Engine (query mode)



2006 tutorial
ri
sastry@us.ibm.com
IBM Research, TJ Watson

ALE 1.1

- Extensions to support Gen 2 tags
  - Filtering, grouping, and reporting extended to include memory banks
- Multiple start/stop triggers
- New stop condition on data arrival
- Writing API

- ALE implementation for IBM Websphere available via Alphaworks

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson

Summary

- ALE functionality and operational model
- ALE core concepts
  - readers, cycles, events, reports
- ALE core objects
  - ECSpec and ECBoundarySpec
- Application programming interface
- Examples (synchronous and asynchronous)

Sastry Duri
sastry@us.ibm.com
IBM Research, TJ Watson