Singularity

Architecture

Empowering People with Information that *Moves*

Tom Rose (tom.rose@i-konect.com)

Sunday, April 23, 2006

Version 1.3

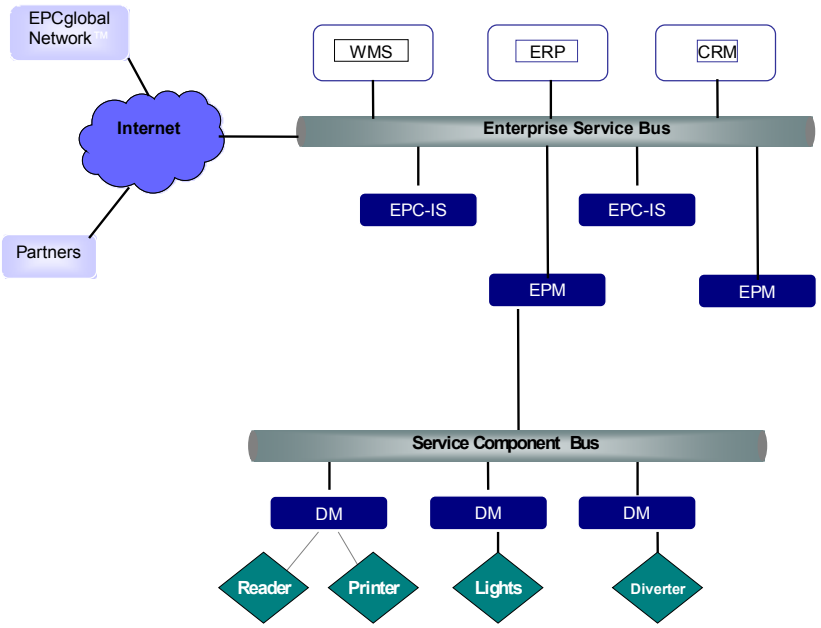
Revision History

Name	Date	Reason For Changes	Version
Tom Rose	2005-03-31	Initial Release	1.0
Tom Rose	2005-04-12	Remove Trademark, changed to Middleware	1.1
Tom Rose	2005-06-15	Added Details on Customization for EPC-IS	1.2
Tom Rose	2006-04-20	Revised EPM	1.3

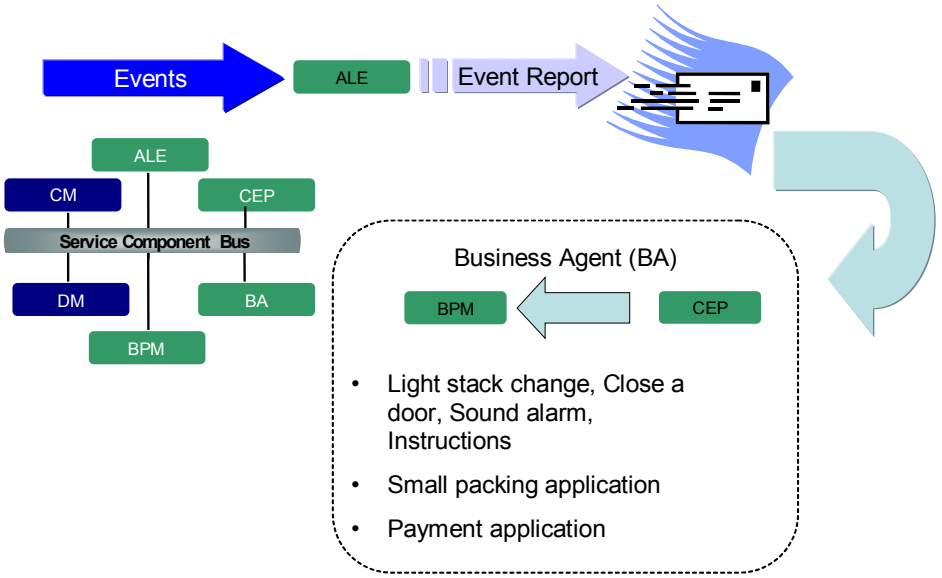
Introduction

The purpose of this document is to define the Software Architecture for Singularity, a complete open source RFID Supply Chain Solution. Includes RFID Middleware as well as Electronic Product Code™ Information Service (EPC-IS).

Singularity Architecture



The architecture consists of two significant components, Middleware and EPC Information Service (EPC-IS). The Middleware contains the RFID device management as well as the Event Process Management services. The RFID Device Managers are distributed agents that provide provisioning, health monitoring, and the first level of event filtering. The Event Process Management services provide the next level of processing of the RFID events, placing the events into business context, and publishing Application Level Events (ALE).



EPM is logical grouping of services available on the Service Component Bus (SCB), providing Complex Event Processing (CEP), Business Process Management (BPM), and Application Level Event (ALE) services. Custom Business Agents will be created that combine the use the services into useful business services at the edge. Also on the SCB are the Configuration (CM) and Device Management (DM) services, to manage the physical reader, printer, and other hardware devices.

The EPC-IS is a subscriber to the ALEs, and is responsible for the management of the ALE event information. EPC-IS is an Enterprise Service existing in a federation of other EPC-IS(s) in the enterprise that can communicate directly with each other, or ultimately reside as a member of an Enterprise Service Bus (ESB). EPC-IS provides EPC event information to the enterprise, as well as to business partners via EPCglobal Network™, through the EPC-IS web services interface.

Each EPC-IS will be able to participate in a distributed query across all multiple Enterprise EPC-IS, as well as including other external EPC-IS residing on the EPCglobal Network™. Any request to the EPC-IS can specify additional scope of the request to be dynamically runtime defined; such as local, regional, enterprise, or global. One of the well-know requirements this will support is Track and Trace. The scope will define what EPC-IS services will be queried for information.

Technology Enablers

The Middleware will utilize Jini™ for the for a distributed fault tolerant agent architecture. For non-Java™ based Middleware devices, proxy agents can be developed to bring the device into the Jini™ enabled agent network.

The EPC-IS will use J2EE™ application services in a clustered configuration for fail-over, and load balancing. However, there may be multiple clusters of EPC-IS in a realistic enterprise deployment.

The selection of Java technology as base for Singularity provides the best possible position for introduction into any company infrastructure. Moreover, this architecture does not preclude a Microsoft® .NET implementation, and Singularity may expand the offering to Microsoft® .NET implementation as the solution evolves.

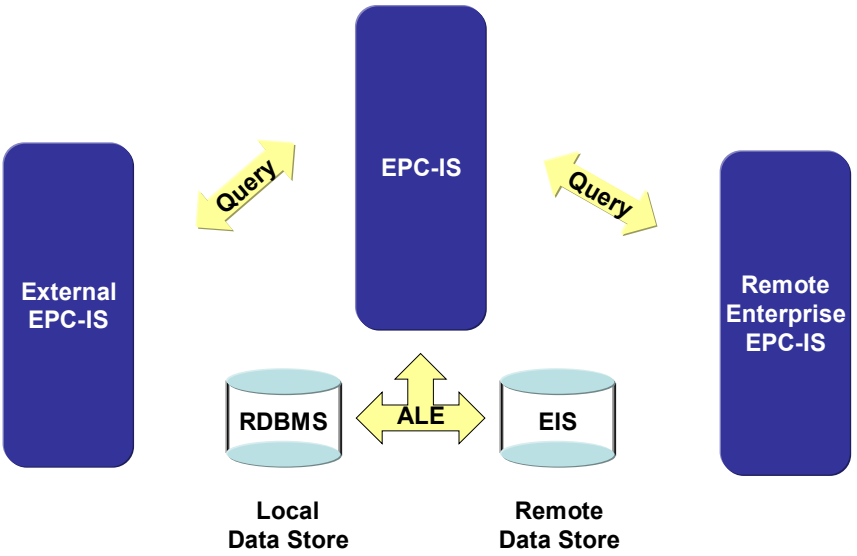
Business Agents at the ESB level will propagate high-level business events (Business Level Events) to the enterprise utilizing an Enterprise Service Bus, other enterprise integration services (Application Integration Services), or point-to-point integration utilizing, for example, Java Connector Architecture. EPC-IS has a pluggable framework to facilitate consuming ALE compliant Middleware events, as well as other custom events from non-EPC related components. These other event generation components may not be directly related to EPC events, however, will be other critical operational information that are important to the interpretation of the Application Level Events that the EPC-IS will receive. Such event information may be machine monitoring events, enterprise business events, or other environmental sensor events. The additional information may be required for accurate handling of EPC information requests for the enterprise or external systems.

The Middleware has a similar design of a pluggable framework, allowing non-EPC related information to flow into the EPM services to facilitate reader management, and publishing of the appropriate ALE to the EPC-IS. Device Managers are pluggable components that may be manage RFID readers, other sensor related devices, etc. They are meant to reside on the host computing platform the device is attached to, and in some cases the computing platform is the device itself.

EPC Information Service Architecture

The EPC Information Service (EPC-IS) manages the EPC related information. It is the entry point to query EPC information from the enterprise or from the EPCglobal Network™.

EPC event data will be sent to an EPC-IS from the Middleware and it will store it locally or may forward to another Enterprise Information System(EIS) such as the enterprise ERP system. Regardless queries/updates serviced by this EPC-IS will not be aware of the location of the information store. In addition, queries that are sent to the EPC-IS will resolve locally (local repository or remote EIS), another EPC-IS in the enterprise, or external EPC-IS on the EPCglobal Network™



It's important to think of EPC-IS as a distributed service, as opposed to a standalone repository of EPC event information. The distinction is the EPC-IS will manage the information, as opposed to just being a local data store of the information for query later. The Singularity EPC-IS approach is a federated service model.

Also note, integration of EPC-IS with EIS services is expected to be accomplished with an ESB. However, if an ESB is not available each EPC-IS will utilize the Java Connector Architecture (JCA) to accomplish the integration with other EIS services.

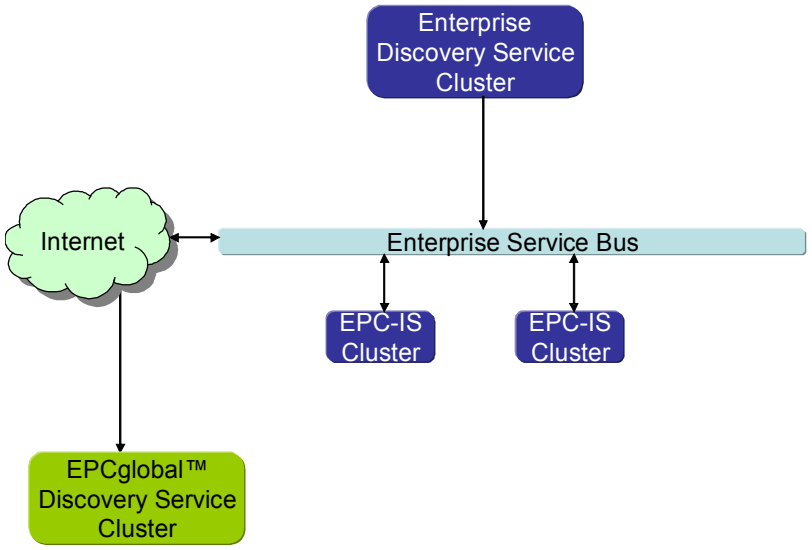
Federated Query

Any EPC related query could be directed at any EPC-IS, and it would answer the query directly or know what other EPC-IS services to query. It would know the scope of EPC-IS services to query in the enterprise through configuration information, or via an Enterprise Discovery Service (EDS).

An enterprise may want to centralize observation information and implement an EDS to facilitate the distributed queries. In this model, the EDS would have enterprise observations logged, and subsequently a subset of those observations would be sent up to the EPCglobal Network™ from the EDS, as opposed to every EPC-IS forwarding observations to the EPCglobal Network™.

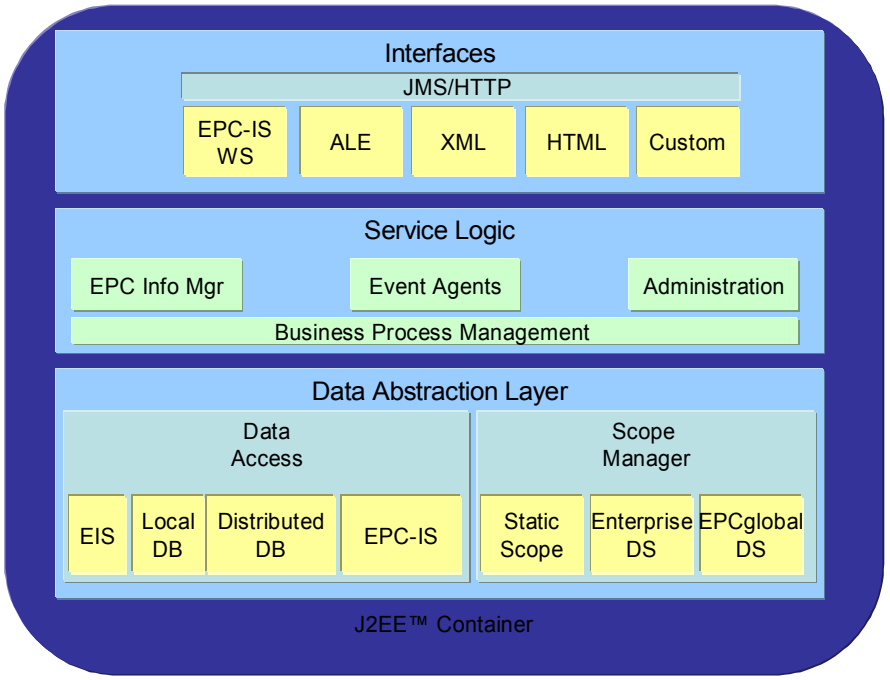
The EPC-IS can have a query scope defined as local, static list of EPC-IS, or dynamically list of EPC-IS services. The dynamic list is created by using EDS and EPCglobal™ discovery service. EPC-IS query scope can be configure as dynamic, static, or both. By embracing this hybrid model of static or dynamic defined information services a corporation my also utilize its investment of distributed database infrastructure, pushing the federated behavior of EPC-IS down into the data model.

If an EDS is used, all enterprise EPC-IS would log relevant observations to the EDS, and EDS would in turn log relevant observations to EPCglobal™ Discovery services. Think of it as a hierarchy of discovery services, similar to a hierarchy of ONS, or DNS services.



Note: EPC-IS instances must be able to participate in a cluster for load balancing as well as hot failover for high-availability. There may be many clusters in the enterprise deployment. Queries are directed at a cluster and not a particular EPC-IS instance.

EPC-IS Component Design



Interfaces

The interfaces provides EPCglobal™ EPC-IS Web Services, Event Web Services, XML, HTML, or other custom application level protocols all over JMS or HTTP. Other non-EPC events can be sent over the other protocols, although the WS interface should be used whenever possible to maximize interoperability.

Service Logic

EPC Information Manager

This tier provides service modules to support the public interfaces. The EPC-IS EPCglobal™ interface is serviced by the EPC Information Manager (InfoMgr). Placing the actual logic to support EPC-IS EPCglobal™ interfaces profiles, the same functionally can be exposed through multiple interfaces if required.

Also provides interfaces to support EPC and product associations, manifest, and other business artifacts that may be associated with the EPC. The business artifacts may reside locally or in other enterprise systems, and the interface provides for use of those artifacts in business process workflows, or to answer EPC related queries from EPCglobal Network, or interaction with enterprise systems.

Event Agents

The Application Level Event (ALE) interface is serviced by Event Agents. Middleware will be sending ALE events; therefore there must be ALE event agents. This will allow

any Middleware provider compliant with the ALE specification to be utilized with EPC-IS. Rather than limit EPC-IS to only ALE events, other non-EPC related event information may be sent, and other agents can be created to service these events. Once an event is defined, it must be associated with one or more event agents that will process the event.

Other types of Event Agents will monitor the event stream and local event repository to infer complex business events, and execute the appropriate business process. A common interface for event agents will facilitate the administration throughout the enterprise, as well as allow other custom agents to be created as needed with additional specific interfaces.

Also Base Event Agent that all Event Agents will inherit from implements an event pipeline of filters. Event Agents can be chained together to provide additional functionality that the initial Event Agent is not aware of. Such as Security or Logging of the event.

Administration

For Administration of the EPC-IS there will be a Web application utilizing the HTML interface and subsequently the Administration service component. Other enterprise administration tools may want to integrate with the EPC-IS, therefore a Web Service or custom XML interface can be developed to expose the Administration service component functionality to those tools.

A Rich Component Platform for desktop administration application utilizing a plug-in framework for each administration component. As new functionality and customizations are made to the system, new custom administration plug-ins can be created as well to manage the customization.

Business Process Management

The Business Process Management (BPM) component can be utilized by all Service Logic components to define their logic; events, administrative tasks, and EPC Information requests may have complex long running workflows defined. For instance and event agent is monitoring the event stream for an alert condition of advanced shipping notice does not match shipment received. That alert would then trigger a workflow that ultimately requires information in the ERP to change state, perhaps e-mails, or SMS, or pager notifications to be sent.

Technology Enablers

For Phase 1, EPC-IS will utilize a J2EE™ 1.4 container to provide security and transaction management, as well as high-availability configurations. J2EE™ will provide the most interoperable platform to accommodate introduction of EPC-IS into the enterprise infrastructure.

Data Abstraction Layer

The EPC-IS will manage local EPC related information either in dedicated repository, EIS, or both. The location and implementation of the data store is abstracted by this layer. Queries to the EPC-IS will require accessing locally managed EPC data, EPC

data managed on another EPC-IS in the enterprise, or externally on the EPCglobal Network™. This layer will manage the scope of the query, as well as any transformation of the data into a conical message format for the EPC-IS components.

The DAL will provide interfaces into other EPC-IS, discovery, database, or EIS systems throughout the enterprise or externally. When an Enterprise Service Bus is available, the complexity of integrating to the EIS is substantially reduced. However, in the lieu of an ESB, the Java Connector Architecture will be utilized to plug-in an open source, commercially available, or internal corporate developed Resource Adapter.

Customization

Out-of-the-box EPC-IS will support basic business artifacts to support business process, and event agents, however, each installation may require specific business artifacts be made available to Singularity. A customization toolkit provides the ability to create new agents, and business artifacts used in the agents and subsequently utilized by business processes that interact with other enterprise systems. This will allow Singularity to conform to the business without creating a one-off unsupported solution. Giving the best combination of a standard product, with in-house customizations to maximizes its effectiveness on business processes when introducing sensor event information into the enterprise.

All of the standard business artifacts are created via the customization process as well. Overview of the customization process is:

1. Define Business Artifact interface or subclass an interface currently available (i.e. JavaBean).

2. Create a Data Access Object (DAO) that implements the Business Artifact Interface. The DAO will utilize the Data Access Framework for information stored in Singularity. Utilize JCA or ESB to access remote enterprise information systems.

3. If stored locally, utilize the customization build utilities and the custom Business Artifact will be compiled and installed. Also the Data Definition Language (DDL) will be generated for target RDBMS (MySQL, Oracle, Sybase, etc), and the appropriate tables created.

   Create an Agent to populate the Business Artifact, and then utilize the customization build utilities to generate remote interfaces for access by HTTP or JMS. Use ESB to facilitate integration with enterprise systems.

4. Once deployed the Business Artifact is available for use in rules or business process (workflow).

Security

The Event pipeline will have security Event Agents that implement various security procedures for the events to authenticate origination of the message, or ensure the message has not been tampered with in transit. Singularity will provide PKI and Shared key event agents; however, other custom security agents can be created.

Object security is implemented via permission based model. All customizations will create objects that must implement getters and setters for information that will be on all objects such as owner, last-updated-by, last-time-updated, etc. Permissions can be created for a class, or set of classes. Permissions have three elements, class, action, and principle (or group of principles) allowed to perform the action. An action also has constraints that can be applied based on the attributes of the class. For example, a read permission for an advanced shipping notice may be constrained to only the owner, recipient, and system. When a read action is executed at runtime the security manager checks the permissions, and throws a security exception if not authorized. The Java security framework provides the mechanisms to implement an object permission based security model that also allows for customized objects. If a customized object does not implement the appropriate interfaces, security checks cannot execute and the system will not allow any action to be performed on the object. This will keep customizations from trying to circumvent system security.

If desired the system can be configured to require customizations to digitally sign the code base, to keep unauthorized customizations from entering the system.

Monitoring

Health monitoring of Singularity will be done via JMX an SMTP interfaces, allowing third party enterprise monitoring tools (i.e. CA Unicenter, HP Openview, etc.) to monitor the health of the Singularity components. The monitoring tools can also provide feed back to Singularity, if required, via a custom agent to incorporate the monitoring event into the event stream of Singularity.

Moreover, it is not envisioned that Singularity would provide monitoring capabilities, however, there is nothing preventing a monitoring agent to be created, along with a custom administration component for that agent if other monitoring packages are not available.