FirstOpen

Open Source for the
Sensor Enabled World

Singularity

Middleware Design

i-Konect

Empowering People with Information that *Moves*

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

Sunday, April 23, 2006

Version 1.0

INTRODUCTION 3
SINGULARITY ARCHITECTURE 3
SINGULARITY EDGE SERVICES DESIGN 4
OVERVIEW 4
Interrogator (Device Driver) 5
MESSAGE SPACE 6
EVENT PROCESS MANAGER 6
Configuration Management 7
Application Level Events (ALE) Service 7
Complex Event Processing 7
Business Process Execution 7
Alerts 8
GRAPHICAL MIDDLEWARE VIEWER 8
SECURITY 8
PUBLIC KEY INFRASTRUCTURE (PKI) 9
MESSAGE AUTHENTICATION CODE (MAC) 10
ADMINISTRATION 10

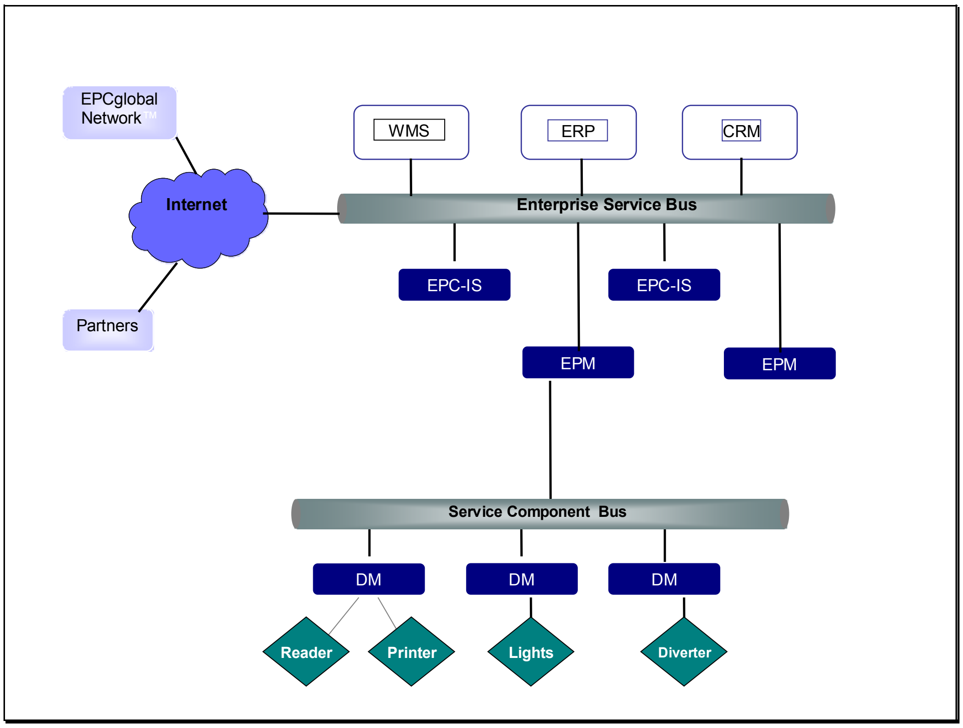
Revision History

Name	Date	Reason For Changes	Version
Tom Rose	2005-05-04	Initial Draft	0.1
Tom Rose	2005-05-09	Updated Device Manager Configuration	0.1.1
Tom Rose	2005-05-28	Updated Event Process Manager	0.1.2
Tom Rose	2006-04-20	Revised EPM (Service Component Bus)	1.0

Introduction

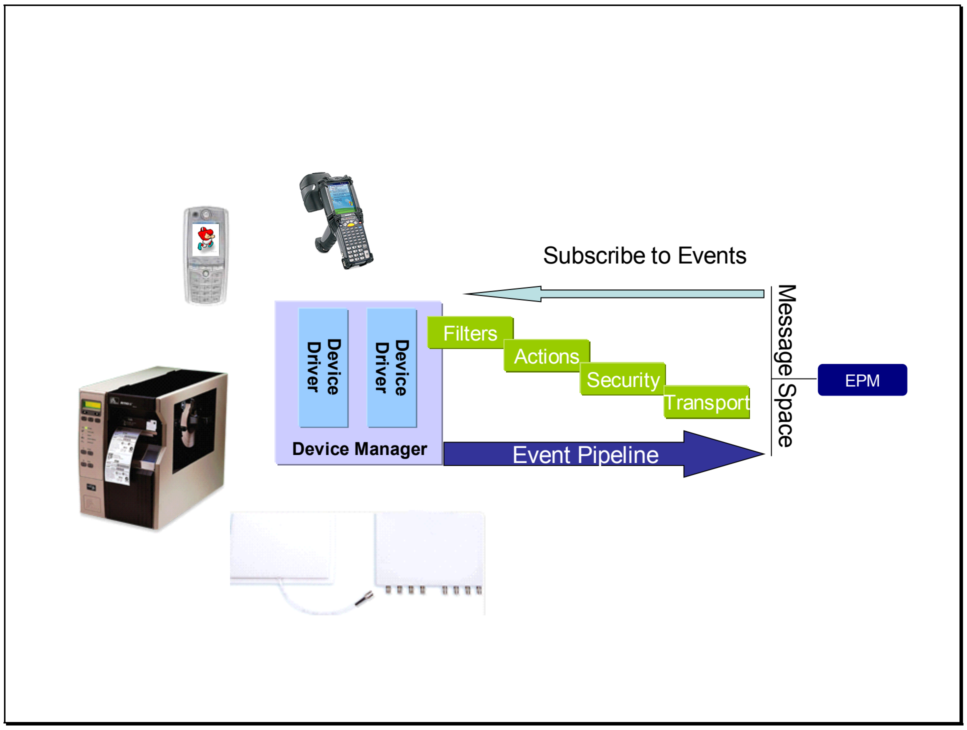
The purpose of this document is to define the Software Design for the Middleware components of Singularity.

Singularity Architecture



Singularity Architecture is has two major components: Information Management Services (EPC-IS), and Middleware. EPC-IS facilitates the integration from of RFID event information to EPCglobal™ as well as other enterprise services (see Singularity Architecture document for further details of EPC-IS). Middleware, the focus of this document, manages devices, filters and propagates events to consumers. Consists of Event Process Management(EPM), and Device Management (DM) services.

Singularity Edge Services Design



Overview

The Middleware Design contains the RFID Device Management as well as the Process Management components. The RFID Device Managers are distributed agents that provide provisioning, health monitoring, and the first level of event filtering. Device managers contain interrogators (Device Adapters), for each reader or sensor device the agent must interact. The Process Management component provides the next level of filtering of the RFID events, placing the events into business context, and then publishing Application Level Events (ALE).

Device Manager

Provides and interface to configure the device remotely, manages one or more interrogators/adapters to the readers or other sensors, and manages ALE event cycles that it may be assigned. There is a one-to-many relationship between Device Managers and Interrogators; each Interrogator will be assigned one Physical Device Profile. Each Physical Device Profile may have multiple sensors (i.e. multiple antennas). When a Device Manager initializes it will determine what Physical Device Profiles it is responsible for and their configuration information, then also securely download and initialize the appropriate Interrogators, and then register with the Event Message Space.

The Device Manager is listening for Sensor Events from the interrogators, those sensor events will equate to their assigned Reader Events. Sensor Events from multiple interrogators may have to be combined to create the Reader Events. Those

Reader Events will equate to the assigned Logical Reader Events that must be sent by the Device Manager to the ALE Service. Logical Readers can be made up of any number of other Readers, managed by one or more instances of a Device Manager. The Device Manager is not aware what other “Readers” are participating in a Logical Read Event. Also a Reader is confined to one instance of the Device Manager.

(Note: can EPC for the Device Profile, and a unique id appended to identify the actual physical device (i.e. 4 Antennas on a single reader device each antenna is SensorID(SGTIN+antennaid)).

Note: as per ALE the Reader Event stream may be coming from multiple Physical Device Profiles. This would require that if a Reader (Sensor Event Streams) were defined to come from multiple Physical Device Profiles then those components would have to be serviced by the same Device Manager Instance.

Device Manager services one or more Device Profiles; each Device has one or more Sensors. The Device Manager creates an Interrogator (Device Driver) for each Device Profile, and instructs the Interrogator on what Sensors are tied to a particular Reader (Sensor Event Stream). The Interrogator sends the Sensor Event streams to the Device Manager that constructs the Reader Event form one or more Sensor Event streams, as the Sensor Event Streams may be coming from different Device Profiles. The Device Manager is also aware of what ECSpec it is participating in, and associates the Reader Event to Logical Reader Event and sends it to the ALE Service.

The following are unique physical identities that will exist that the Device Manager must be aware:

- Device Manager ID – Uniquely identifies a device manager instance, this uniqueness should have the domain space of the enterprise.
- Device Profile ID – Uniquely identifies a Physical Device Profile, this uniqueness should have the domain space of the enterprise.

- Reader ID – Uniquely identifies each source stream of sensor events, and is called a Reader, and there may be multiple Readers within a Physical Device Profile (i.e. multiple antennas, or one-wire sensor connections)

- Sensor ID – Uniquely identifies physical device (sensor). Each physical device (e.g. antenna or one-wire sensor) has a unique identification, and there may be multiple sensors in a physical component.

Configuration of a Device Manager will consists of the device(s) it manages and the interrogators that it will spin up. A Device Profile will be created for each physical device, and then associated with its appropriate device manager. Each Device Profile will consist of the physical device attributes, and how to configure an interrogator. Physical device attributes are, but not limited to, named sensors (i.e. antennas, and one wire sensors).

Interrogator (Device Driver)

The Interrogator is an instance of an adapter/driver for the specific reader or other device. There is one Interrogator for each Physical Device Profile, and a specific type of agent for each reader model. Although one Interrogator may service multiple

Physical Device Profiles, each **physical reader** may only have one interrogator. Each of the Interrogator's assigned devices may have multiple sensors, such as multiple antennas, or multiple one-wire devices. A SensorID will be given for each sensor. As standards for reader interfaces are established these interfaces will consolidate, however, there will always be vendor specific configurations that will be accommodated.

The Device Manager will initialize the integrator with configuration information for the assigned Physical Device Profiles. This will include the physical interface configuration (internet address:port, serial port), and sensor configuration. The agent will send the event stream from each sensor to the Device Manager, to construct the Reader event.

- ❖ Sensor ID – Uniquely identifies physical device (sensor). Each physical device (e.g. antenna or one-wire sensor) has a unique identification, and there may be multiple sensors in a physical component.

Message Space

The Message Space is a common fault-tolerant event message broker. It will support receiving events asynchronously from the Device Managers, as well as ALE and other configuration requests from the Event Process Manager. The Event Message Space provides a store and forward mechanism as well. Ensuring important events are not lost in the case of intermitted network or other component failures.

Event Process Manager

The Event Process Manager (EPM) consists of ALE Service, Device Management, Complex Event Processing, and Business Rule Execution. Access to the EPM can be achieved through HTTP, JMS, Web Service Interface, or other interfaces. These are service components available on the service component bus. The service component registers/subscribes to the events that it is interested in, and then is notified when the events are available in the Message Space(currently via ALE, however, lower subscription directly to the device managers can be accomplished as well). Business rules at this level are meant to initiate device interaction such as indicator lights, audio signals, door activation, etc. Business processes that require integration with other Enterprise Information Systems are expected to exist as Business Agents at the Enterprise Service Bus level.

Upon receipt of these events, for example, a business agent can apply complex event processing (filters), and then business rules can be executed. In the case where an external client (such as EPC-IS) has registered to receive Application Level Events, those filtered events are sent out to the ALE client specified location.

Configuration Management

Physical devices will be deployed, and their physical information such as MAC address, IP, public keys, shared keys, logical reader assignments, etc. is managed in this component. The information may be stored local or in the case of security keys access external directory services or other repositories where the information managed for the enterprise.

Define the following, and their relationships:

Sensor (Antenna, one-wire events)
Physical Device Profile (May have one or more sensors)

Reader (one more sensor events event streams from one or more Physical Device Profiles)

Logical Reader

Application Level Events (ALE) Service

This service is an implementation of the EPCglobal™ ALE Specification. It will provide a Web Services Client Interface as well as a JMS interface for the ALE API. At a high level ALE specification allows logical readers, and logical read cycles to be created. One or many readers, as well as one more many read events from those readers can constitute a logical event. The ALE also specifies where those logical events must be sent. The ALE service essentially orchestrates participation of Device Managers in a logical reader, and subsequent logical events. The ALE service then propagates those events to the appropriate recipients.

Complex Event Processing

A rule engine used to filter events into meaning business events that can then be utilized by business rules. ALE service will feed the Complex Event Rules, that will subsequently generate useful events. These events can then be used to initiate business rules.

Business Process Execution

Since the Middleware is expected to be deployed in a variety of business environments specific functionality is not hard-wired into the Middleware. Business Agents can be defined to listen for events (ALE or lower-level), and initiate various business processes through the use of a rule engine, BPM, or both. Although the ALE will provide a rich event cycle definition interface, interpretation of those events and subsequent actions that must be taken, and will be specific to the deployment environment. Business process may require a door to be shut, or alarm initiated, notifications sent, etc.

When the business process is at a higher level, and requires integration with enterprise systems such as Warehouse Management Systems, it's expected that these business process will be defined as Business Agents at the Enterprise Service Bus level. However, a Business Rule Engine is available at the Middleware level to allow better customization and integration with other lower-level devices specific to each deployment.

Alerts

Create named alerts, and subsequently become topics that can be support subscriptions in as in a publish and subscribe messaging model. The Alert provides for a named alert, description, message topic, as well as message payload persistence.

The alerts are defined, and then instances of the alerts can also be persisted. Alert instances have related Alert Type, message payload (optional data sent with the alert, may be null).

Graphical Middleware Viewer

The Graphical Middleware Viewer (GMV) will utilize Eclipse Rich Client Platform as a base framework for GUI development. This will allow all features to be created as plug-ins utilizing the OSGi model http://www.osgi.org . This model has tremendous benefit for adapting a rich client with many useful features, and successfully allowing them to co-exist without prior knowledge other plug-ins. For further information about Eclipse RCP goto http://www.eclipse.org/rcp/.

The GMV will provide a graphical interface into administer and configure all Middleware components. It will also be able to receive events from the Event Message Space, to allow a graphical representation of real-time events.

Security

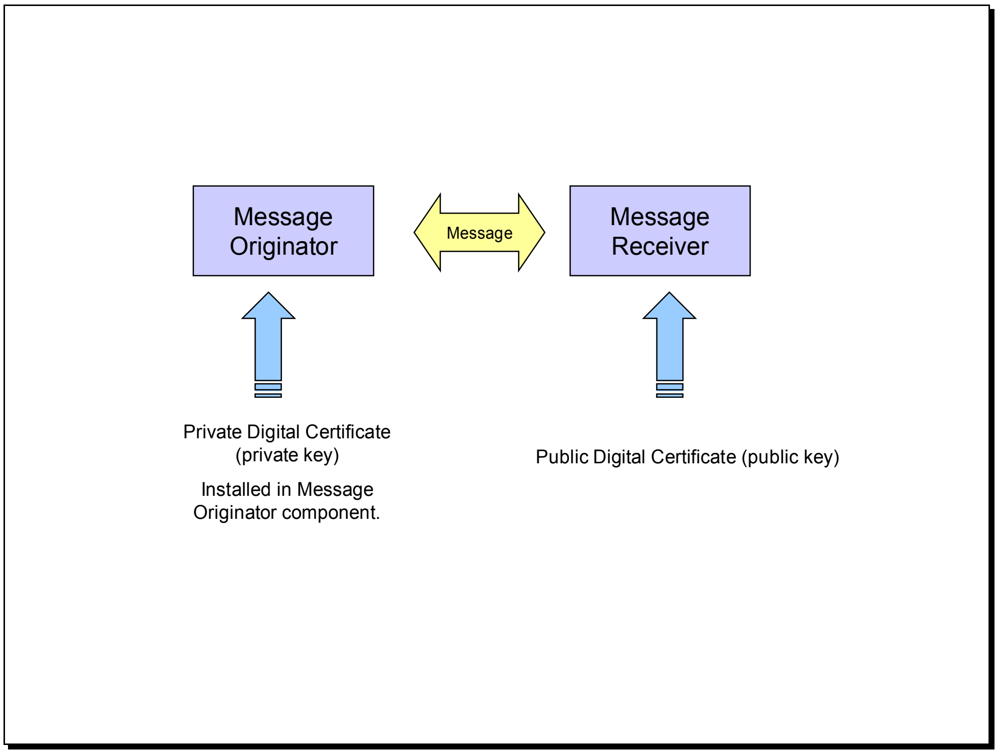
All access to components will be secured utilizing The Java Authentication and Authorization Service (**JAAS).** JAAS will allow the component security to easily integrate with most enterprise authentication environments. If required custom JAAS components can always be created and plugged into the Singularity services.

Java Cryptography Architecture will be utilized for encryption schemes such as MAC or PKI. The following will provide additional details about what is available http://java.sun.com/j2se/1.5.0/docs/guide/security/ ; however, each deployment has different security requirements. The Singularity components allow communications to utilize the appropriate security protocols to meet each deployments particular need. The base functionality of each component is unchanged by configuring none, or different types of encryption/digital signature schemes.

Below describe two methods for digitally signed messages, and are only here as an example of what is possible.

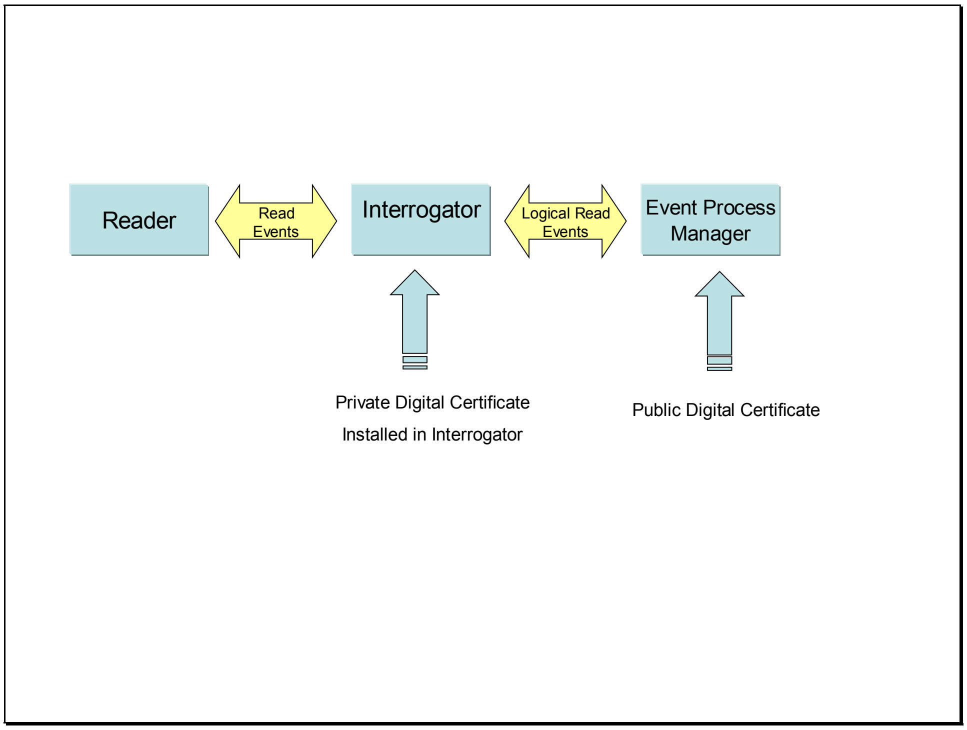
Public Key Infrastructure (PKI)

PKI consists of a key pair, one private key and held only by the owner. The other key is public, and freely distributed.


Message Originator
Message
Message Receiver
Private Digital Certificate (private key)
Installed in Message Originator component.
Public Digital Certificate (public key)

Regardless of the actual components involved in Singularity a message has an originator and a receiver. In the case of the Interrogator being the originator, each interrogator has their private key installed. This will enable the interrogator to digitally sign a message. Whatever device that receives the signed message it must have the pubic key to verify the signature created by the private key. This allows any device to verify the validity of the message, since only the public key is required, and also uniquely identify and verify the originator.

Messages can be passed on unchanged to other devices. If changed, there is a new originator and a new private key is used that is associated with the new message originator to sign the new message. If unchanged, the additional receivers just need the public key of the originator/signer of the message, just like the original receiver.
Keys can be issued by trusted certificate authorities like Verisign, or internal corporate certificate authorities.


Reader
Read Events
Interrogator
Logical Read Events
Event Process Manager
Private Digital Certificate Installed in Interrogator
Public Digital Certificate

For each Interrogator serviced their public key is installed in this component, or is made available through a remote query mechanism. This will enable the component to verify the message's digital signature and validity. Whatever device that receives the Logical events (not just the middleware) it must have the pubic key to verify the signature created by the private key on the originating device.

Message Authentication Code (MAC)

A MAC mechanism that is based on cryptographic hash functions is referred to as HMAC. HMAC can be used with any cryptographic hash function, e.g., MD5 or SHA-1, in combination with a secret shared key. HMAC is specified in RFC 2104.

The process for digitally signed message using this mechanism is similar to PKI. The difference being there is only one key, it's private and is shared by both originator and receiver. However, the same flow will follow.

Administration

Install DeviceManagers, once DeviceManagers are started for the first time their DeviceManager ServiceIDs are created, and then are available for configuration in the Administration Browser.