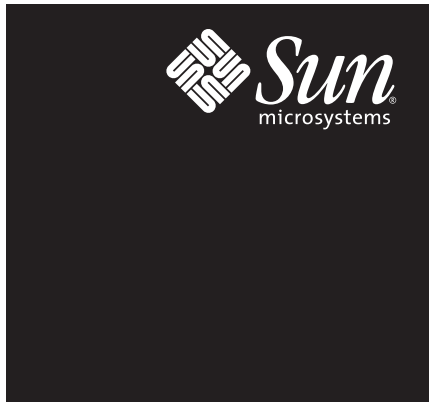
The Sun™ EPC Network Architecture



A Technical White Paper
February 2004

Table of Contents

Chapter 1

Introduction

The Electronic Product Code (EPC) and the EPC Network are intended to help businesses improve asset visibility and help ensure product safety and integrity across the supply chain. Companies not only need to know where their assets are, but also need to share that information with their trading partners to deliver seamless, efficient, and secure business transactions. The EPC Network enables trading partners to track and trace items automatically throughout the supply chain. This provides businesses with an unprecedented real-time view of their assets and inventories anywhere, thereby enabling significant gains to operational efficiencies and brand protection efforts. The EPC Network supplies benefits beyond operational efficiencies by enabling safe and secure supply chains with applications that address counterfeiting, tampering, terrorism, and regulatory compliance, among others.

This document describes the architecture for enabling the EPC Network. The architecture is specifically designed to address large-scale implementations in enterprises that need to integrate real-time data flowing in from existing business processes and back-end enterprise systems using Automatic Identification (Auto-ID) tag technologies such as Radio Frequency Identification (RFID) tags. This architecture is part of the Sun™ EPC initiative, which encompasses software, hardware, services, and best-of-breed partnerships to help create comprehensive solutions for the enterprise.

The Sun EPC Network architecture is built with Sun Java™ Enterprise System software, which delivers a set of shared technology components and component products for enhanced integration and simplified maintenance. The Java Enterprise System provides an integrated set of industry-leading network services that virtually all businesses need today, at a single, annual license fee for software, support, maintenance, consulting, and education services.

Chapter 2

EPC Network Architecture

The architecture for enabling an EPC Network is shown in Figure 2-1.

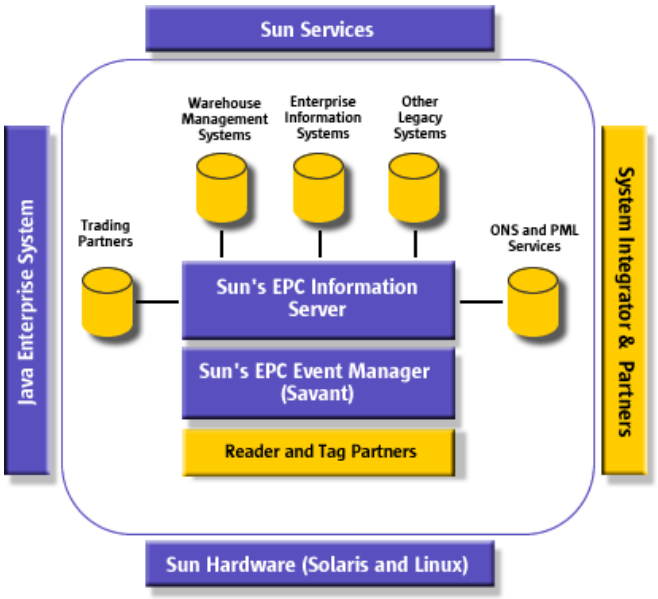


Figure 2-1: Sun EPC Network Architecture

At the bottom of the stack are tag readers or sensors that are responsible for reading tagged items, which may be on a shelf or may be moving through a portal such as a door or cross dock. Each reader continuously reads many tagged items and sends that data to the EPC Event Manager (Savant) for processing. For each reader, typical throughput is approximately 200 reads per second.

The next layer in the architecture stack is Sun's Savant middleware, the Sun EPC Event Manager, which is designed to process streams of tag or sensor data (event data) coming from one or more reader devices. Sun's implementation of the Savant middleware has the capability to filter and aggregate data prior to sending it to a requesting application. For example, a tagged object that is sitting in front of a reader without moving generates many redundant reads. The Sun EPC Event Manager's filters can be programmed to throw out any data that shows the tagged object in the same place, and trigger an action or event only when there is a change in state for the object. For example, an action is triggered when the object moves or a new object comes into the reader's view. The Sun EPC Event Manager can be programmed with other types of filters to enforce specific business rules, as well. Once the filtering has occurred, relevant data may be persisted for use by other layers in the Java Enterprise System stack.

To localize reader traffic, an enterprise may have numerous instances of the Sun EPC Event Manager at each geographically remote site, such as a store, distribution center, or warehouse. A typical store or warehouse is likely to have many readers. Given the amount of network traffic from readers, it is important to localize data by enabling the Sun EPC Event Manager servers to filter the tag data at each site, instead of sending it over the Internet. In addition, it is good practice to isolate the readers from the Internet for security reasons.

The third layer in the Sun architecture stack is the Sun EPC Information Server. Sun advocates that integration technologies be used to connect the Sun EPC Event Manager layer to enterprise information systems (EIS) such as legacy, enterprise resource planning (ERP), warehouse management systems (WMS), supply chain management (SCM), and customer relationship management (CRM) systems, as well as other applications that might want to use tag information. These include components and technologies that comprise the Java Enterprise System, such as Web, application, communication, and security services, and Java technologies such as the Java Message Service (JMS) and the Java 2 Platform, Enterprise Edition (J2EE™) Connector Architecture (CA) to enterprise information systems. Data translation as well as business process management may be needed to enable EIS systems to optimally leverage the real-time information collected and forwarded by the Sun EPC Event Manager. Depending on the specific requirements, either session beans or servlets can be written to run on the Sun Java System Application Server platform.

The topmost layer in the architecture stack is comprised of EIS systems such as ERP, WMS, legacy systems, and proprietary enterprise systems. These must accept and integrate data and events received about tagged objects. As part of the standard EPC Network architecture, it is planned that there will be EPC Information Servers (IS) that hold and disseminate product information in the Physical Markup Language (PML) format. The rest of the paper discusses each of the stack's layers in more detail.

Sun EPC Event Manager – The Savant

The Sun EPC Event Manager is a key component of the EPC software stack. It is based on version 1.0 of the Savant standards, which were developed as part of the Auto-ID Center Software Action Group (SAG). The Sun EPC Event Manager adheres to the basic 1.0 specifications, and provides additional features and functionality that are specifically designed to address large-scale, enterprise implementations.

Let us first look at the functionality that the standard Savant was envisioned to provide, and then at how the Sun implementation (the Sun EPC Event Manager) adds value.

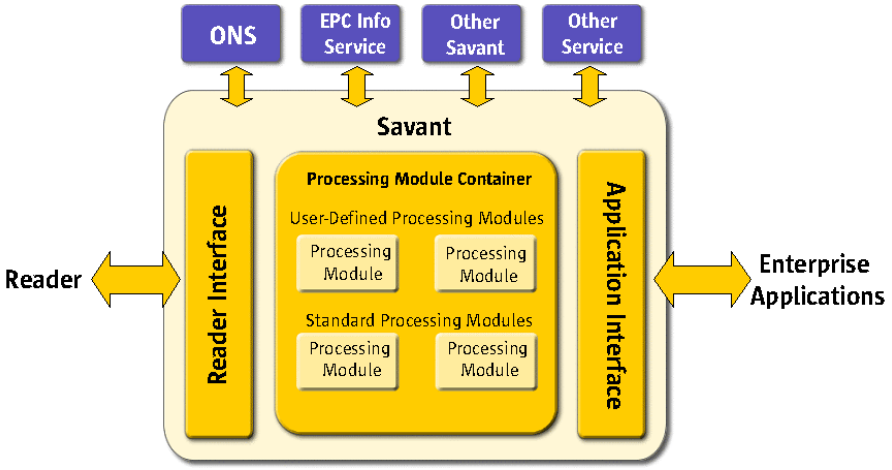


Figure 2-2: Basic EPC Savant Architecture

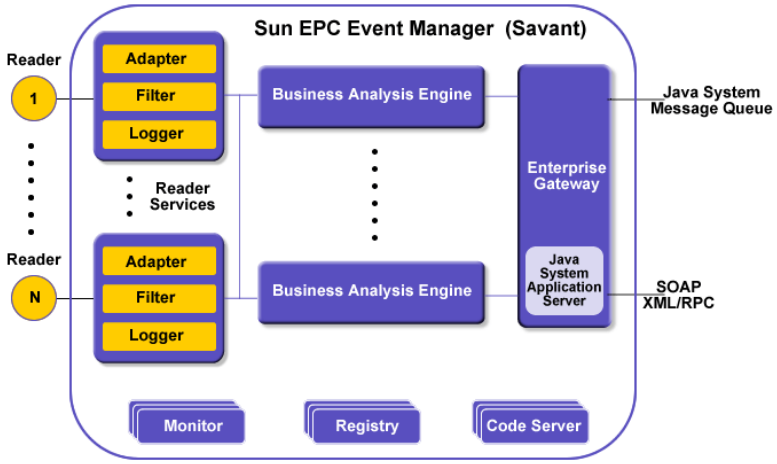
The Savant is primarily responsible for processing data from RFID tags with a unique EPC that describes a tagged object's manufacturer, product type, and serial number. The Savant provides the following benefits:

- It provides an interface that enables RFID readers and other network devices or sensors to be connected to the EPC Network.
- It helps integrate RFID event data with the EIS by defining a set of interfaces that facilitate sending and receiving real-time data to and from these systems.
- It provides a general-purpose, event routing system.

Essentially, the Savant is a data collector and router that performs operations such as data capture, data monitoring, and data transmission. For each reading, the Savant gathers a minimum amount of information such as the tag's EPC, the EPC of the reader that scanned the tag, and the timestamp. Specific requirements for EPC processing vary from application to application, so the Savant is defined in terms of processing modules or services, each of which provides a specific set of features and may be combined to address specific application requirements. Since the emphasis in the version 1.0 specification of the Savant framework is on extensibility rather than specific processing, it defines only the most basic processing modules, and lays out a framework within which user-defined processing modules can function. This modular architecture facilitates innovation without committing — at this stage — to a monolithic specification that attempts to satisfy universal requirements.

The Sun EPC Event Manager was designed to allow for flexible deployment capabilities without sacrificing availability, scalability (both horizontal and vertical) or manageability. One characteristic that makes the Sun EPC Event Manager unique is its distributed architecture.

Figure 2-3: Sun EPC Event Manager Architecture



A fundamental tenet of distributed systems is that they must be able to accommodate changes that may occur on a network. Compute resources on a network, such as a server or other networked device, may fail or die. Sometimes a new compute resource may be introduced to the network. As a result of the above actions, applications executing on a particular compute resource may perform poorly or fail completely. This is very similar to the situation experienced in large-scale enterprise Auto-ID and EPC network deployments. For example, a warehouse reader may go down unexpectedly, or a server may be knocked over by a forklift. Since distributed systems are designed to inherently accommodate the vagaries of compute resources connected by a network, the Sun EPC Event Manager is also designed around this type of architecture.

The Sun EPC Event Manager implements what is known as a federated service architecture, which essentially provides distributed, self-organizing, and network-centric capabilities. These building blocks enable a dynamic, distributed architecture capable of adapting to unforeseen changes on the network. This architecture also makes the Sun EPC Event Manager highly scalable. For example, individual services or components — the Reader Service or Registry — may be distributed to run across multiple computing resources on a network.

Additionally, this architecture is resilient. If a reader or other computing resource is physically disabled or damaged on the shop floor, the Event Manager continues to work by dynamically provisioning and relocating software services from the disabled computing resource to another compute resource on the network.

Some key components of the Sun EPC Event Manager are:

- Device Adapter: This layer enables devices from many different manufacturers, such as RFID or bar code readers, to communicate and interact with the Event Manager.
- Filters: These assist in deciphering useful data from the noise constantly generated by tagged objects. Filters may also contain small pieces of process or business logic. Standard filters are provided for event smoothing, event batching, event changes (tags in and out), and event pass/blocking.
- Loggers: These are somewhat similar to device adapters, except that loggers are used to notify external systems of RFID and non-RFID event data. The Sun EPC Event Manager provides standard loggers that log information either to the file system, a JMS queue, or through XML, http, and SOAP messages.
- Enterprise Gateway: This component is used as the common interface for enterprise applications requesting data from the Sun EPC Event Manager.

Sun EPC Information Server

The Sun EPC Information Server provides access to significant business events generated by the Sun EPC Event Manager. It also serves as an integration layer that offers several options for integrating the Sun EPC Event Manager with existing EIS or custom enterprise applications. Direct connections from the Sun EPC Event Manager layer to enterprise applications can result in the creation of information silos within a company. Using the Sun EPC Information Server between the Sun EPC Event Manager and back-end applications allows for maximum flexibility when business requirements or enterprise applications change. Using software and APIs that are part of the Java Enterprise System enables developers to quickly and flexibly integrate EPC data with enterprise applications.

Data from the Sun EPC Event Manager feeds into the Sun EPC Information Server, where it is stored and made available in a consistent manner to any application that needs it. This approach increases overall system reliability and flexibility, while reducing maintenance and support costs. It also provides a suitable location for correlating EPC events to business logic. Another benefit is that the Sun EPC Information Server can be used to implement additional functionality, such as data reformatting and data warehousing.

The Java System Application Server provides three options for point-to-point integration with EIS systems, meaning it ties a Savant server to an EIS system. These options are:

- J2EE CA
- Asynchronous reliable messaging through the Java Message Service
- Native support for Web services

The J2EE CA, which is part of the J2EE platform (version 1.3 onwards), defines a standard way to tightly couple an EIS system with either a Web application or Web service. With the appropriate connector installed, an application can employ the functionality of the EIS without having to deal with the complexity of integrating remote access, transactions, and security. The functionality of the EIS appears as just another service provided by the application server.

In the case of the J2EE CA, the EIS or custom system is tightly coupled to the application. This means it can make a call to the EIS, much like it would make a call to a database, via JDBC™ technology. To the Web application or Web service using the J2EE CA, the EIS looks like a local resource, a familiar paradigm that eases the developer's task. Additionally, the Java System Application Server can take care of underlying system issues such as pooling, security, and transaction support.

The disadvantage of tightly coupled integration is that many times it is not appropriate, because unlike a database, the EIS may take a relatively long period of time to make updates. For example, integration with a supplier's ordering system may require human intervention before receiving a response to an order request, which could mean holding the connection open for days. In cases where connections are held open for a long period of time, a loosely coupled approach where asynchronous messages are sent and queued to the EIS is more appropriate.

For loosely coupled integration, the Java System Message Queue, which is included in the Java System Application Server, provides the standard JMS asynchronous reliable messaging mechanism for integrating Web applications or Web services with an enterprise's Message-Oriented Middleware (MOM) environment. This facility allows Java Enterprise System applications to exchange messages with the EIS.

The third option is provided by the platform's native support for Web services. By their nature, Web services easily cross machine and software boundaries, and so are well-suited for solving integration problems. Because the EIS and Web service running on the Java System Application Server view each other as Web services, they can interact using standards such as the Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI).

These approaches provide flexible options to integrate the EIS to Savant servers with point-to-point integration for a narrow or single purpose. The developer could write workflow logic into an Enterprise JavaBeans™ (EJB™) stateful session bean or servlet to tie multiple EIS systems together using the integration options previously mentioned. This is an excellent approach to consider if the business process is straightforward and fairly static.

Tightly Coupled Integration

As previously mentioned, the J2EE CA defines a standard architecture for connecting the J2EE platform to heterogeneous EIS and custom applications. It addresses the key issues and requirements of EIS integration by defining a set of scalable, secure, and transactional mechanisms that enable the integration of EIS with service containers and enterprise applications.

With the J2EE CA, a service container and connector (and underlying EIS) can collaborate to keep all system-level mechanisms — remote access, transactions, security, and connection pooling — transparent to the application. The J2EE CA provides the most direct mechanism for integrating the Sun EPC Information Server with an EIS. This mechanism results in a close coupling between the two, providing both high performance and high reliability.

Loosely Coupled Integration

The Java System Message Queue provides asynchronous reliable messaging through the coordination of the following main components:

- Administered Objects
- Client Runtime
- Messaging Service

Administered Objects

Administered Objects encapsulate provider-specific implementation and configuration information in objects that are used by client applications. Such objects are created and configured by an administrator, stored in a name service, accessed by client applications through standard Java Naming and Directory Interface™ (JNDI) lookup code, then used in a provider-independent manner.

The Java System Message Queue provides two types of administered objects: ConnectionFactory and Destination. While both encapsulate provider-specific information, they have very different uses within a client application. ConnectionFactory objects are used to create connections to the Java Message Service, while Destination objects identify physical destinations. Administered Objects make it easy to control and manage a Message Service because the behavior of connections can be controlled by requiring client applications to access preconfigured ConnectionFactory objects through a JNDI API lookup.

The proliferation of physical destinations can be controlled by requiring client applications to access only those Destination objects that correspond to existing physical destinations. This arrangement provides control over Java Message Service configuration details. At the same time, it allows client applications to be provider independent. They do not need to know about provider-specific syntax, object naming, or configuration properties.

Client Runtime

As the second main component of the messaging system, Client Runtime provides client applications with an interface to the Java Message Service by supplying them with the JMS programming objects. It supports all operations necessary to enable clients to send messages to destinations and receive messages from them.

The Java Message Service

The Java Message Service provides the core functionality of the asynchronous reliable messaging system. It is made up of the following main components:

- *Broker* – A broker provides delivery services for the messaging system. Message delivery relies upon a number of supporting components that handle connection services, message routing and delivery, persistence, security, and logging. The Java Message Service can employ a single- or multi-broker configuration.
- *Physical Destination* – Delivery of a message is a two-phase process — delivery from producing client to a physical destination maintained by a broker, followed by delivery from the destination to one or more consuming clients. Physical destinations represent locations in a broker's physical memory and persistent storage.

PML and EPC Information Service

This section briefly describes the still-evolving Physical Markup Language (PML) and the EPC Information Service standards.

PML development is part of the Auto-ID Center effort to develop standardized interfaces and protocols for communication with and within the Auto-ID infrastructure. The Auto-ID PML Specification 1.0 defines syntax and semantics.

PML provides a standardized, XML-based format for the exchange of EPC data and consists of two parts:

1. PML Core: Describes raw data received from tags and sensors
2. PML Extensions: May be used to associate arbitrary information with physical objects

In the end, PML provides a common, broadly hierarchical method for describing physical objects. For instance, a particular vendor's beverage might be described as a Carbonated Beverage, which falls under the subcategory Soft Drink, under the broader category of Food. Not all classifications are so simple, so to ensure that PML has broad acceptance, the Auto-ID Center relies on work already completed by other standards bodies. PML does not attempt to replace existing vocabularies for business transactions or other XML application libraries, but does complement these with definitions about EPC Network system-related data.

Types of PML Data

In addition to product information that does not change (such as material composition), PML includes data that changes constantly (dynamic data) and data that changes over time (temporal data). Dynamic data in a PML file might include the temperature of a shipment of fruit. Temporal data, such as an object's location, changes throughout an object's life. Making all of this information available in the PML enables companies to employ information in new and innovative ways. A company could, for instance, set triggers so the price of a product falls as its expiration date approaches. Third-party logistics providers could offer service-level contracts indicating that goods are stored at a certain temperature.

EPC Information Service

The EPC Information Service essentially makes EPC Network-related data available in PML format to requesting parties with appropriate authorization. Data available through the EPC Information Service may include:

1. Data collected from the Sun EPC Event Manager through readers or sensors
2. Specific data about the tagged object such as date of manufacture, weight, expiration date, and so on
3. Product catalog information

In responding to requests, the EPC Information Service obtains information from a variety of data sources that already exist within an enterprise before translating that data into PML format. So an EPC Information Service may be a simple Web service front-end to an EIS.

Java Enterprise System

Bringing the best in Java Web and application services together, the Java Enterprise System delivers enormous value to IT organizations by helping to reduce both cost and complexity. It provides all the core enterprise network services needed to build a secure, scalable, and highly available Java technology-based application or service. Enterprise network services are the capabilities that sit between the traditional operating system — such as the Solaris™ OS or Linux OS — and business applications. Enterprise network services are engineered and deployed to meet business requirements for a scalable, interoperable, available, and secure IT software infrastructure. This enables IT organizations to focus on business logic development when deploying business solutions.

The Java Enterprise System delivers a set of shared technology components and component products for enhanced integration and simplified maintenance. The Java Enterprise System is offered at $100 per employee, per year. This single low price eliminates the complexity of managing multiple pricing structures for multiple software products, support contracts, and service agreements, making it easier for customers to acquire, deploy, and run the software they need. This single annual license fee covers software system, support, maintenance, consulting, and education services.

The Java Enterprise System and services help customers spend more time focusing on their business requirements rather than integrating and supporting a myriad of point products. Customers gain better control and increased business agility while benefiting from reduced IT costs and complexity through the Java Enterprise System's simplicity, predictability, and affordability.

The core set of the enterprise network services that the Java Enterprise System delivers are:

- Web and Application: Based on J2EE technology, these services maximize application reuse and developer collaboration, enabling IT organizations to develop, deploy, and manage applications for a broad range of servers, clients, and devices.
- Network Identity Services: Used to improve security and protect key corporate information assets, these services help ensure that appropriate access control policies are enforced across all communities, applications, and services on a global basis.
- Portal: Provide anytime, anywhere access capabilities to user communities, delivering personalization, aggregation, security, integration, mobile access, and search. Portal services enable mobile employees, telecommuters, knowledge workers, business partners, suppliers, and customers to securely access their personalized corporate portal from anywhere outside the corporate network through the Internet or extranet.
- Communications and Collaboration: Specific capabilities include messaging, real-time collaboration, calendaring, and scheduling in the context of the user's business environment.
- Availability: Enables the predictability and resilience that businesses expect from their application infrastructure. Availability services also provide the patented “Always-On” technology for application and Web services, delivering extremely high-quality service and massive scalability.
- Security Services: Delivers the peace of mind business today demands. Security services provide consistent single sign-on to online resources. They protect content using the latest security standards and resilient authentication and access control options.

Components

The following components comprise the 2004Q2 release of the Java Enterprise System. Additional components will be added in future releases.

- Java System Directory Server 5.2
- Java System Identity Server 6.2
- Java System Directory Proxy Server 5.2
- Java System Application Server Platform Edition 7
- Java System Application Server Standard Edition 7
- Java System Message Queue Platform Edition 3.5
- Java System Message Queue Enterprise Edition 3.5
- Java System Web Server 6.1
- Java System Portal Server 6.3
- Java System Portal Server Mobile Access 6.3
- Java System Portal Server Secure Remote Access 6.3
- Java System Messaging Server 6.1
- Java System Calendar Server 6.1
- Java System Instant Messaging 6.2
- Cluster 3.1
- Sun Cluster Agents for System components: Web, Application, Directory, Messaging, and Calendar Servers and Message Queue

Software Service and Support

Sun's consulting, proactive support, and in-depth education services help customers architect, implement, and manage their Java Enterprise System environment. Sun Services offers comprehensive architecture, implementation, and management services and methodologies that enables customers to take full advantage of the Java Enterprise System. Sun Services provides customers with technical support, software maintenance, installation services, custom consulting, and comprehensive education to ensure a smooth transition and integration of the Java Enterprise System into their enterprise.

Summary

The Sun EPC Network architecture is specifically designed to address large-scale, enterprise Auto-ID EPC deployments. The architecture can scale from a single small site with a few readers, to many geographically dispersed sites with multiple readers at each site. The Sun EPC Event Manager is uniquely designed for enterprises with reliability, scalability, and manageability in mind, providing an integration layer that further allows scaling of EPC Network infrastructures. It provides many options for incorporating tag data and events with existing business processes and EIS systems.

The Java Enterprise System is a radical new approach that changes forever the way businesses acquire, develop, and manage software. Using the Sun EPC Network Architecture in the Java Enterprise System infrastructure can deliver many business benefits. Only Sun has the experience and the end-to-end portfolio to deliver such a unique and industry-revolutionizing strategy. With the Java System, EPC information is integrated into essential business applications are faster, easier, and at a lower cost than ever before — so you can focus on innovation, competition, and bottom-line results.

Chapter 3

Best Practices

Sun encourages the following best practices and steps when pursuing an Auto-ID EPC deployment:

1. **Empower a multidisciplinary team**: A dedicated team with executive sponsorship is important to the success of any Auto-ID EPC deployment or pilot. Many companies underestimate the effort required to undertake an Auto-ID EPC project. Because the project can often be complex, it requires the cooperation of executives and senior-level managers from IT, Engineering, and Operations departments.
2. **Define a simple and measurable business issue**: What are the business goals that the enterprise wants to achieve with its deployment? Given the emerging nature of Auto-ID EPC, it is best to find a contained compelling application for Auto-ID within the four walls of the company that offers the most value to the organization. Addressing labor-intensive tasks such as asset tracking, shrinkage-reduction, and inventory threshold (out of stock) applications are popular pilot projects.
3. **Scope out data and business processes**: An accurate understanding of existing data and business processes at the outset of any Auto-ID EPC project is crucial to success. Without knowing how a business process works today, companies will find it difficult to implement and track changes.
4. **Select partners**: Trying to manage everything by yourself can be daunting, so it is essential to leverage your partner s' experience. Why build expertise in areas that are not likely to be part of your core competency, when you could be focusing on those aspects of the project that are more critical to your company's bottom line?

5. **Map network architecture and enterprise integration**: Most companies start with readers and tags as part of their Auto-ID EPC pilot. However, this causes them to get bogged down with physics experiments. Therefore, it is important to think early on about the overall network architecture. Assuming that the tag reads are figured out, think about how data will flow from the reader and integrate with the EIS systems. This may actually help you better understand overall system requirements and even what type of readers and tags infrastructure is required.
6. **Determine the right readers and tags**: Once the data requirements and overall network architecture are mapped, specify the readers and tags requirement. Key factors include material (metal or liquid), size of items to be tagged (cases or pallets), application (global asset tracking, local shrinkage problem), environment (indoor or outdoor, temperature and moisture), regulatory (what frequency is legal in which part of the world), and human or psychological factors (how does this affect stakeholders).
7. **Prototype**: Start small and see how all the pieces come together in a test environment. Compare data from existing processes. Test your assumptions and make sure you are seeing the hoped-for improvement.
8. **Implement**: Once you have necessary data from the prototyping efforts, go ahead and implement the new system.

Chapter 4

Conclusion

The EPC Network holds the promise to significantly improve supply chain productivity by providing unprecedented visibility — a real-time view of assets and inventories wherever they are located. This visibility can enable companies to track and trace goods from the original manufacturer through distribution points and on to the retailer. This information can be integrated into business-critical applications, or shared with supply partners, and can provide dramatic gains to operational efficiencies and brand protection efforts.

While creating a cost-effective way to generate and collect product data is a significant challenge, so is integrating it into the EIS and ERP systems and repositories. Modern businesses rely on multiple back-end applications to run their business, and many of these will need access to the data feeds from Savants distributed across company operations. Rather than creating new, standalone modules, Sun's architecture for EPC Networks integrates EPC data with existing applications — those used to run your business today.

As an early sponsor and leader of the EPC Network technology, and Sun has participated in some of the first trials with leading consumer goods manufacturers and retailers. With over three years of experience, Sun technology and expertise — along with best-of-breed partners — are helping companies with a range of solutions. Whether you need a proof of concept or a production deployment, Sun can help your company, too.

Chapter 5

More Information

To learn more about the Sun's Auto-ID technology and products, visit the Web sites below. Organizations can also contact Sun or its partners to learn how we can help build competitive advantage with solutions that match business goals and processes to technology solution.

sun.com/autoid	Sun Auto-ID
sun.com/erpscnetwork	iForce ERP/Supply Chain Network
sun.com/iforce	Sun's iForce℠ Initiative
java.sun.com	Java Technology and Standards
sun.com/software/javasystem	Sun Java System
www.epcglobalinc.org	EPCGlobal
www.autoidlabs.org	Auto-ID Labs
sun.com/javaenterprisesystem	Java Enterprise System

Please
Recycle

Adobe PostScript

Learn More

Get the inside story on the trends and technologies shaping the future of computing by signing up for the Sun Inner Circle program. You'll receive a monthly newsletter packed with information on the latest innovations, plus access to a wealth of resources. Register today to join the Sun Inner Circle Program at sun.com/joinic.

To receive additional information on Sun software, products, programs, and solutions, visit sun.com/software.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 800 786-7638 or +1 512 434-1577 Web sun.com

Sun Worldwide Sales Offices: Africa (North, West and Central) +33-13-067-4680, Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India–Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +822-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand–Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China–Beijing +86-10-6803-5588; Chengdu +86-28-619-9333; Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Singapore +65-6438-1888, Slovak Republic +421-2-4342-94-85, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland–German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44-1-276-20444, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800 02/04 R1.0