# Sun's Auto-ID Architecture

White Paper
June 2003

# Table of Contents

Chapter 1

# Introduction

Auto-ID technology can greatly enhance an enterprise's ability to manage its supply chain by providing a real-time view of how items are moving throughout the supply chain. And the result can be unprecedented gains in operational efficiency for the enterprise. Typical applications that customers are pursuing with Auto-ID include minimizing out-of-stock situations, minimizing shrinkage of inventory through theft or expiration, and improving inventory management in general. In all of these cases, significant positive impact can be made to a firm's bottom line by increasing revenue or decreasing costs.

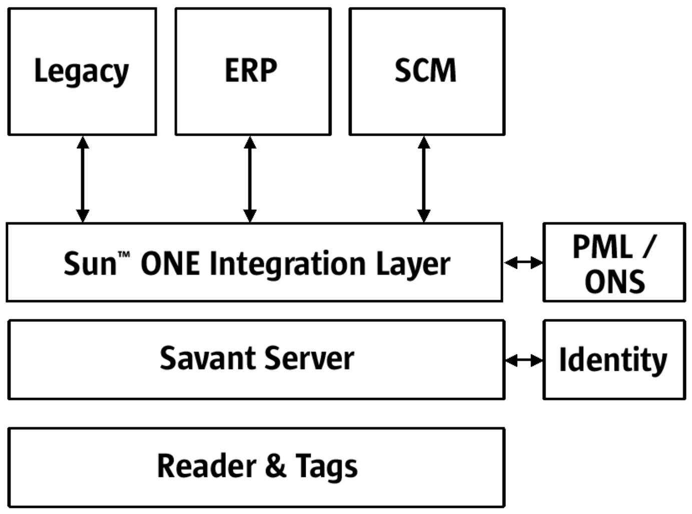
This document describes Sun's Auto-ID architecture, which is currently being pursued by some early adopters in both the retail and manufacturing industries, with Sun as the integrator.

Chapter 2

# Sun's Auto-ID Architecture

Sun's Auto-ID architecture is illustrated inFigure 2-1.

**Figure 2-1:** Sun's Auto-ID architecture



At the bottom layer is the reader that will be responsible for reading the tagged items that may be on a shelf or may be moving through a portal such as a door or dock. Normally, the reader will be reading many items continuously and sending that data to the Savant server for processing. Typical throughput would be 100 reads per second.

The next layer in the architecture is the Savant server. The Savant server's role is to quickly process all the tag data that is coming in from one or more readers. It buffers the reads and then, through filtering, keeps pertinent reads. For example, in the case of a shelf reader, most of the reads are from items that have not moved on or off the shelf. Since the item would have been stored in a database previously, the redundant reads need to be filtered out. Other types of filters can be used to enforce business rules as well. Once the filtering has occurred, the data that is kept needs to be persisted for use by other layers in the architecture.

Typically, an enterprise with a large supply chain will have numerous Savant servers located at each site within their supply chain to localize the reader traffic. A typical store or warehouse will have numerous readers in the various shelves. Given the amount of network traffic from the readers, it is important to localize the data by having the Savant servers filter the tag data at each site instead of sending tag data over the Internet. In addition, it is good practice to isolate the readers from the Internet for security reasons.

The third layer in the architecture is the Sun™ Open Net Environment (Sun ONE) Integration layer. Sun advocates that integration technologies be used to connect the Savant layer to the enterprise information systems (EIS), such as legacy, enterprise resource planning (ERP), warehouse management, supply chain management (SCM), and customer relationship management (CRM) systems, as well as other applications that might want to use tag information. These technologies include Java™ technologies such as the Java Message Service (JMS) and Java 2 Platform, Enterprise Edition (J2EE™) Connector Architecture to connect the Savant server to the EIS systems. Data translation as well as business process management is still necessary to have the EIS systems optimally leverage the real-time information on the supply chain that is contained in the Savant. Depending on the specific requirements, either session beans or servlets can be written running on the Sun ONE Application Server or Sun ONE Integration Server and can be used in more complex integration scenarios.

EIS systems are at the top layer of the architecture. They can benefit from using the tag data to better manage a firm's supply chain. In the future, there will also be product information stored on Physical Markup Language (PML) servers. The remainder of the paper discusses each of these layers in more detail.

Chapter 3

# Savant Server

This section describes the Savant server using the preliminary Savant Server specification from the Auto-ID Center as a reference. The Savant server is responsible for processing data from Radio Frequency Identification (RFID) tags that have unique electronic product codes (EPCs) describing the manufacturer, product number, and serial number. The Savant server provides the following benefits:

- Enables third-party vendors of RFID systems and other auxiliary systems providers to connect to the Savant by providing device adapters
- Helps integrate RFID data with existing ERP or other systems by defining a set of generalized logger and adapter interfaces that can connect to these systems to send or receive real-time data
- Provides a general-purpose, event-routing system

The Savant is a data router that performs operations such as data capturing, data monitoring, and data transmission. For each reading, the Savant gathers, at minimum, information such as the EPC of the tag read, the EPC of the reader that scanned the tag, and the timestamp of the reading. The Savant consists of three major modules:

1. Event Management System (EMS)
2. Real-time In-memory Event Database (RIED)
3. Task Management System (TMS)

The Savant's Event Management System (EMS) allows the Savant to react to various real-time events such as EPC reads, sensor measurements, and bar code scans. As events are gathered in the network, they can be persisted in Savant's Real-time In-memory Event Database (RIED). The RIED subsystem supports SQL-based access data held in a relational schema. Such a data store will be updated on every event that enters the Event Management System. To provide a consistent view of the data while queries are performed, without making the updates inefficient, RIED supports queries on snapshots of its state.

Batch operations can be configured on the information gathered by the EMS using the Task Management System (TMS). Operations such as periodic lookup of PML data or data exchange with ERP systems can be configured as tasks in the TMS.

**Figure 3-1:** Savant server architecture

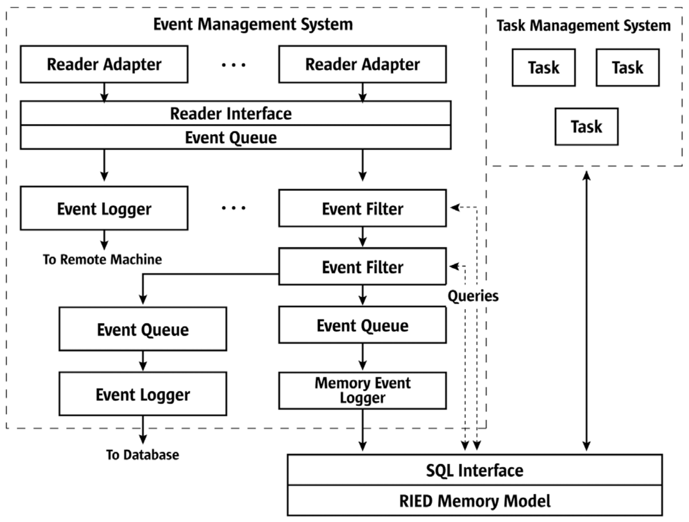


Figure 3-1 depicts a sample configuration of a Savant server that uses the EMS, RIED, and TMS modules. As shown, events captured from reader adapters are processed by event queues, event filters, and event loggers. These events are then logged in the RIED memory model. Meanwhile, tasks managed by the TMS monitor the EPC data.

## Event Management System

The Event Management System (EMS) collects tag reads and other events. The EMS is responsible for:

- Allowing adapters to be written for various types of readers
- Collecting EPC data from readers in a standard format
- Allowing filters to be written to smooth or clean EPC data
- Allowing various loggers to be written, such as database loggers to log EPC data into the database, and HyperText Transport Protocol (HTTP)/JMS/Simple Object Access Protocol (SOAP) network loggers to broadcast EPC data to remote servers
- Buffering events to enable loggers, filters, and adapters to operate without blocking each other

## Real-time In-memory Event Database

Real-time In-memory Event Database (RIED) is a specification for a real-time in-memory relational database system. Edge Savants maintain and organize events sent by readers. The Event Management System provides a framework to filter and log events. The loggers can log events in a database. However, databases cannot handle more than a few hundred transactions per second. RIED provides the same interface as a database, but offers much better performance.

RIED maintains one or more snapshots of the database. The latest snapshot is the updatable image of the database. Other snapshots are read-only images of the database. The maintenance of old snapshots may be useful in applications such as inventory counting. These applications can run on read-only snapshots without affecting the performance of the event updates.

The Java DataBase Connectivity (JDBC™) interface presents a JDBC view to the RIED schema. This industry-standard interface allows a remote machine to access the database using standard SQL queries and locate the database using standard URLs.

The reference implementation of RIED supports SQL operations such as SELECT, UPDATE, INSERT, and DELETE. The RIED reference implementation supports a subset of data manipulation operations defined in SQL92.

## Task Management System

The Task Management System (TMS) is a generalized scheduling manager for executing custom tasks in the Savant. Its main purpose is to provide a simple, yet powerful system for:

- Scheduling recurring tasks based on a schedule
- Scheduling permanent tasks that are ensured to run continuously and are restarted on failure or completion
- Running tasks on demand
- Enabling dynamic registration and deregistration of tasks

The Savant software performs data management and data monitoring using customizable tasks. Loosely, a task can be considered as the equivalent to a process in a multitasking system. The TMS manages tasks, just as the operating system (OS) manages processes.

The Task Management System provides many features that garden-variety thread managers and multiprocessing operating systems do not offer, such as:

1. An external interface to schedule tasks
2. A platform-independent (Java) virtual machine with uniform libraries loaded on-demand from redundant class servers
3. A robust scheduler maintaining persistent information about tasks and capable of restarting tasks following a failure

The TMS simplifies the maintenance of distributed Savants. The enterprise can maintain Savants by merely keeping the tasks on a set of class servers up to date, and appropriately scheduling tasks on the Savants. The tasks written for the TMS can access all the facilities of the Savant. The tasks can perform various operations for the enterprise, such as:

1. Data gathering: Send or receive product information to and from another Savant
2. PML lookup: Look up Object Name Service (ONS)/PML servers to gather static and dynamic product instance information
3. Remote task scheduling: Schedule and remove tasks on other Savants
4. Remote upload: Send product information to remote supply-chain management servers

Chapter 4

# Sun ONE Integration Layer

The Sun ONE Integration layer provides several options for integrating Savant servers with existing EIS or custom applications through either the Sun ONE Application Server, for point-to-point integration, or the Sun ONE Integration Server EAI Edition, for tying multiple systems into a complex workflow.

## Integration With the Sun ONE Application Server

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

- The J2EE Connector Architecture
- Asynchronous reliable messaging through Java Messaging Service
- The Sun ONE architecture's native support for Web services

J2EE 1.3 technology, through the J2EE Connector Architecture (J2EE CA), defines a standard way to tightly couple an EIS system with either a Web application or Web service. With the appropriate connector installed, a Sun ONE application is able to use the functionality of the EIS without having to deal with the complexity of integrating EIS remote access, transactions, and security. The functionality of the EIS appears to the developer as a new service provided by the application server.

In the case of J2EE CA, the EIS or custom system is tightly coupled to the application, which means it can make a call to the EIS much like it would make a call to a database via the JDBC API. To the Web application or Web service using J2EE CA, it makes the EIS look like a local resource. The advantage is that it makes it easier for the developer because it is a familiar paradigm, and as mentioned before, the Sun ONE Application Server can take care of issues like pooling, security, and transactions.

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

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

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

These approaches provides flexible options to integrate EIS systems to Savant servers with point-to-point integration for a narrow or single purpose. The developer would write workflow logic into a stateful session Enterprise JavaBeans™ (EJB™) component or servlet and tie multiple EIS systems together using the integration options previously listed. If the business process is straightforward and considered to be fairly static, that is an excellent approach.

**Tightly Coupled Integration**

As mentioned, J2EE CA defines a standard architecture for connecting the J2EE platform to heterogeneous enterprise information systems 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 systems with service containers and enterprise applications.

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

The Sun ONE Connector Builder is a set of tools, components, and libraries that enables developers to build J2EE Connector Architecture connectors for EIS, custom, and legacy applications. The resource adapters built using the Sun ONE Connector Builder provide J2EE applications with an easy-to-use way to access these external systems using the J2EE CA standard. The generated resource adapters are accessible from Web applications using the Common Client Interface (CCI) API. Optionally, resource adapters generated using the Sun ONE Connector Builder can also be accessed using SOAP services, thus providing a Web services solution to application integration.

**Loosely Coupled Integration**

The Sun ONE Message Queue provides asynchronous reliable messaging through the coordination of the following main components:

- Administered Objects
- Client Runtime
- Message 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) API lookup code, then used in a provider-independent manner.

The Sun ONE 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 Message Service, while Destination objects, which represent physical destinations, are used to 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 Destination objects that correspond to existing physical destinations.

This arrangement, therefore, provides control over 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 and object naming or provider-specific configuration properties.

**Client Runtime**

As the second main component of the Sun ONE Message Queue, the Client Runtime provides client appli-cations with an interface to the Message Service by supplying them with all the JMS programming objects. It supports all operations necessary to enable clients to send messages to destinations and to receive messages from them.

**The Message Service**

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

- *One or More Brokers:* 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. A Message Service can employ a single- or multi-broker configuration.
- *Physical Destinations:* 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/or persistent storage.

## Integration With the Sun ONE Integration Server

The approaches listed for the Sun ONE Application Server can solve numerous integration challenges. However, at some point the business process is either so complex (involving numerous EIS systems with a complex workflow tying them together) or is so dynamic (for example, new suppliers are expected later) that a different approach is needed. Enterprise application integration (EAI) is where EIS systems are viewed as independent entities that are loosely coupled together into a common communication infrastructure. This is the approach taken with the Sun ONE Integration Server EAI Edition. Each EIS system is accessed through an adapter that communicates to the EIS system's API. A process engine is then used to coordinate how and when the EIS systems interact. A business analyst can control the workflow through a graphical process specification tool, making it easier to change the process or add new EIS systems as well.

The basic unit of a process model is the activity. An activity represents a unit of work in a process. The process logic involves, principally, answering the following questions in the order shown below.

1. What is the general sequencing (or routing) of activities in the process?
2. What conditions must be met before an activity can be performed?
3. Which user or users can perform each activity?
4. What activities should be performed before completion of each activity?

The personnel that are most uniquely positioned to understand how the enterprise accomplishes its business are most often domain experts, sometimes called business analysts. They have expertise in how the enterprise accomplishes its goals but may not be software developers. Thus, a graphical process specification tool is essential to the effective management of information assets in conjunction with business abstractions. Such a tool can specify how the system is to manage, coordinate, and drive to conclusion all the information and personnel aspects of a system.

The Sun ONE Integration Server is a suite of business integration tools for integrating and coordinating heterogeneous applications. The tools and software components provided with the Sun ONE Integration Server enable integration of newly developed applications, legacy applications, and off-the-shelf packages (including Web browsers) into business processes that are managed and controlled by a process engine.

The Sun ONE Integration Server includes the following features:

- *Point-to-point integration between applications.* Integration is accomplished using eXtensible Markup Language (XML) messaging transported over HTTP or by using implementations of the Java Message Service (JMS). Adapters are available to integrate many package applications that do not have a native XML interface. The Sun ONE Integration Server adapter toolkit enables integration of any custom application.
- *Business process management using the process engine*. Application proxies interface with the process engine to provide end-to-end control and automation of all the processes in an enterprise's business. Business process integration can combine automated systems with human interactivity.
- *Message brokering and XSL data transformation using the backbone application proxies.* XML messages are transformed according to eXtensible Stylesheet Language (XSL) rules registered with the Sun ONE Integration Server backbone. Messages are then routed through the process engine and delivered to partner applications. XSL provides a standards-based way to share data across applications and domains.
- *Interactive tools for process development and XML/XSLT authoring.* The process development workshops allow customization of the workflow of activities in a business process, including defining user profiles and user validations, and sharing data between activities in a process. The XML/ eXtensible Stylesheet Language: Transformation (XSLT) workshops include an interactive debugger.
- *Robust runtime environment with failover capabilities.* Interactive process management tools enable monitoring and management of process engine performance. The process engine can be configured for failover and contains tools for the backup and restoration of process data.
- *Support for multiple styles of messaging.* The Sun ONE Integration Server supports XML messaging over HTTP and Secure Sockets Layer (SSL), as well as messaging using implementations of JMS. Application proxies can also access application services made available with SOAP. Different styles of messaging can be combined in the deployment of a Sun ONE Integration Server system.

## Sun ONE Integration Architecture

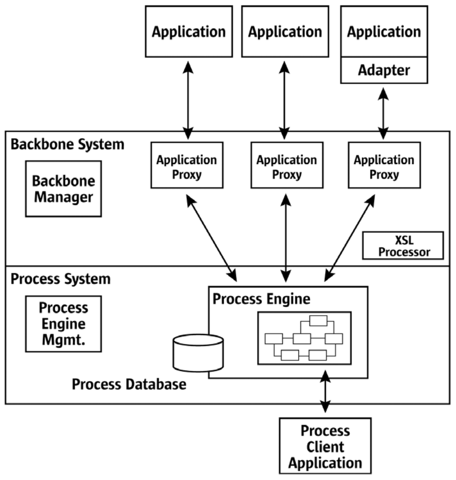


**Figure 4-1:** Sun ONE Integration Server architecture

The Sun ONE Integration Server contains two systems — a backbone system and a process system, shown in Figure 4-1.

The backbone system contains application proxies that use XML-based messaging to communicate with one or more partner applications. Application proxies can also access applications that export services using SOAP. The process system contains a process engine that manages and controls a business process, coordinating activities in the process with application proxies.

**Sun ONE Integration Server Backbone**

The Sun ONE Integration Server backbone system enables XML messaging between enterprise applications and a Sun ONE Integration Server process engine. Customers primarily use a backbone system to:

- Provide an XML interface that allows applications to exchange and translate XML messages (this allows different types of applications to communicate with each other or with the process engine)
- Enable applications to participate in a managed business process

*Application Proxies.* The heart of a Sun ONE Integration Server backbone is a set of *application proxies* that perform message brokering and data transformation on behalf of applications. For business process support, proxies interact with a process engine on behalf of applications that participate in a common business process. The main purpose of these interactions is to communicate the initiation and completion of work activities according to a process defined for the process engine.

Application proxies are configurable, and can act as a client, server, or both with respect to an application. A proxy can also represent more than one application, if appropriate. Because proxies are configured (not coded), their function and behavior can be quickly specified and modified. Configuration information includes management of incoming and outgoing XML messages and data as well as maintaining sessions with its partner applications.

*Adapters.* Adapters provide an interface to packages or custom applications that lack a native XML interface into a backbone. The Sun ONE Integration Server provides an adapter toolkit that allows creation of adapters according to specific needs. An adapter serves the following functions:

- Transforms XML documents from the proxy into requests using the application's native API
- Interprets the responses to these requests and generates appropriate application documents to be sent to the proxy
- Responds to messages or application callback requests, generates appropriate documents, and forwards these to the proxy

*XML Messages.* XML messages between a proxy and its partner applications can be transported over HTTP or by using implementations of the Java Message Service. Secure transmission over HTTP is available using SSL. Additionally, application services that are made available through SOAP can be accessed by application proxies.

*XML/XSLT Workshops.* The Sun ONE Integration Server provides XML/XSLT workshops that facilitate the development, testing, debugging, storage, and management of XML documents and XSL stylesheets that are used for message transformation between applications.

*XSL Stylesheets.* XSL stylesheets contain XSL rules that are assigned to a proxy for decoding and encoding XML documents exchanged in messages with partner applications. By writing XSLT rules and adding them to the stylesheet base, the function and behavior of a proxy can be defined.

*XSL Processor.* The XSL processor transforms XML messages, according to rules in the XSL stylesheet base, into documents that can be used by a proxy to coordinate a business process with the process engine or to communicate with applications.

*Backbone Management Utility.* The Sun ONE Integration Server contains the Fusion Script utility for configuring, starting, and managing a backbone system and its proxies.

**Sun ONE Integration Server Process System**

A process system supports the execution and management of business processes, coordinating the activities in the process with external applications. The heart of this system is the process engine, which coordinates the work of different resources and applications that participate in the processes. Customers use the process system to:

- Develop process logic using the process development workshops
- Manage processes using the process engine console and other tools
- Build process client applications that make direct API calls to the process engine using the process client APIs

*Process Engine.* The process engine executes and manages process definitions created with the process development workshops. The process engine interacts with application proxies in a backbone system to coordinate the activities of a process with external applications. The process engine also interacts directly with process client applications (applications created using a process client API). A process engine maintains an engine database containing state information on processes that it executes. The engine can be configured for failover and recovery, using information from the engine database as well as logging and history information to restore the state of the engine.

*Process Development Workshops.* The process development workshops enable the interactive definition and editing of business processes, which then can be registered with a process engine. Tasks that can be performed include:

- Defining work activities for the process
- Specifying process flow by routing work between activities
- Creating and setting timers to automatically trigger activities and events
- Specifying user profiles and validation rules
- Incorporating assignment rules for activities defined in the process logic
- Defining process attributes that control the state of the process across the system

*Process Management Tools.* The Sun ONE Integration Server provides a suite of tools for distributed system management of a process system, including configuring, starting, and shutting down process engines. Highlights of system management include:

- Registering and executing processes
- Monitoring and managing process execution
- Maintaining application sessions with the process engine
- Handling failover and recovery
- Maintaining history logs of process execution

Chapter 5

# PML Service

This chapter describes the PML service, which is a standard still under definition by the Auto-ID Center, using the preliminary PML specification from the Auto-ID Center as a reference. The Electronic Product Code identifies individual products, but all the useful information about the product will eventually be written in a new language called the Physical Markup Language (PML), which is based on XML.

PML will provide a common method for describing physical objects. It will be broadly hierarchical. So, for instance, a particular vendor's cola might be described as a carbonated beverage, which would fall under the subcategory soft drink, which would fall under the broader category food. Not all classifications are so simple, so to ensure that PML has broad acceptance, the Auto-ID Center will rely on work already done by other standards bodies.

## Types of PML data

In addition to product information that doesn't change (such as material composition), PML will include 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 changes throughout an object's life, such as an object's location. By making all of this information available in a PML file, companies will be able to use 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 will be stored at a certain temperature.

## PML Server

PML files will be stored on a PML server, a dedicated computer that is configured to provide files to other computers requesting them. PML servers will be maintained by manufacturers, and will store files for all of the items a manufacturer makes.

Chapter 6

# Best Practices

Sun's Auto-ID architecture allows enterprises to pursue demanding Auto-ID projects. The architecture can scale from one small site with a few readers to many sites with multiple readers through the Savant and reader layers. Through the integration layer, many options exist to tie in EIS systems located anywhere in the world.

In addition, Sun encourages following best practices when pursuing an Auto-ID deployment. First, it is important to define the business goals that the enterprise wants to achieve with Auto-ID. Given how new Auto-ID is, it is best to find very compelling applications for Auto-ID that offer tremendous value to the organization. Current deployments show that out-of-stock and theft control application are popular, but certainly many other applications exist. Most customers are doing limited deployments with certain stores and/or certain product lines, and will move ahead with other deployments later.

Second, look at the integration requirements needed to achieve the business goals for the project. What applications and data are needed, who are the players involved in the business process, and how does EPC play a role in the process are all factors that need to be taken into account. From there, integration requirements and what technologies to use within the Sun ONE Integration layer can be determined.

Finally, look at the readers and tags. Though most people tend to play with the readers and tags first, Sun believes the integration issues need to be sorted out. At some point though, readers and tags need to be evaluated, and any Radio Frequency (RF) issues must be resolved between them.

**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.