Telecom Network Management With Enterprise JavaBeans™ Technology

A Technical White Paper

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

May 2001

Send comments about this document to: docfeedback@sun.com

Please
Recycle

Adobe PostScript

Contents

Telecom Network Management With Enterprise JavaBeans™ Technology

This white paper explains how builders of the next generation networks will benefit from moving to a standard application server model for their management frameworks. It emphasizes the benefits of an application server model provided by a combination of the Java™ 2 Platform, Enterprise Edition (J2EE™) and the Enterprise JavaBeans™ (EJB™) component model. In particular, this white paper explains how such an application server model can ease the construction of element management and network management systems. This white paper assumes that the reader is familiar with telecom network management technologies.

Need for a New Approach to Telecom Network Management

As a result of deregulation, network service providers that were once monopolies are now subject to intense competition. Deregulation has been accompanied by developments in mainstream computing (notably the advent of Internet technologies) that have led to an explosive increase in the amount of data traffic carried by telecom networks.

This trend has continued as demand has grown for seamless communications anywhere at any time between a variety of devices. In addition to sophisticated telephony services with attractive pricing, telephone subscribers now expect other services such as voice mail, Internet access, multi-media, email, wireless access, and IP telephony.

To retain their competitive edge, network service providers need to be able to supply new and innovative services cheaply and quickly. They also need to develop high-bandwidth backbone infrastructures and manage the growth in mobile and Internet traffic while reducing operational and management costs.

Current telecom network management systems are unable to meet these needs. Incumbents have huge legacy systems, which have been built piecemeal over many years. New entrants are faced with system integration costs that massively exceed the costs of the components.

Telecom network management has the characteristics of a mature software discipline, such as reference models, information models, communications protocols, design patterns, and reference architectures. However, implementations are large, expensive, unwieldy, and unresponsive.

This situation has arisen because the development of telecom network management systems presents a unique set of challenges that are difficult to address adequately in a legacy development environment. Many of these challenges can be addressed by a move to an industrial-strength component-based development environment, such as that provided by J2EE and the EJB component model. The development environment provided by J2EE technology and the EJB component model helps simplify the development and deployment of large complicated enterprise applications.

Telecom Network Management Challenges

The many telecom network management systems in existence today are designed to meet a common set of challenges. Some of these challenges arise from the competitiveness of the telecom market. Others arise from the technical complexity of the telecom network management domain. This complexity comes from the need to manage performance, alarms, and states in networks consisting of many millions of network elements of many different types. To meet the challenges of today's telecom market, telecom network management systems must be highly reliable, offer high performance, and be easy to operate.

Short Time to Market

The telecom market is competitive and rapidly changing. To maintain their competitive edge, network service providers need to be able to meet the demands of their subscribers for new and better services. However, because there are many players in this market, the time to market for new services is very short.

When the time to market is short, technologies must evolve quickly. New products must be developed efficiently and must be highly reliable. There is no longer time to design, develop, test, and deploy a complete telecom network management solution from scratch. Instead, new products must be developed by assembling existing components that are known to be reliable and offer the required functionality.

For example, components that provide basic management functions (such as alarm and topology management) are reused in combination with components that provide extended capabilities (for example, user profiling or logging services). Reusing components in this way requires network management frameworks that are versatile enough to support many different configurations.

Wide Distribution of Services

Network elements managed by telecom network management systems are typically spread over a wide area. Network service providers maintain systems that may span countries and even continents. The trend to reducing the cost of ownership of such systems requires more centralized management, as local management becomes infeasible. Centralized management requires distributed management systems and reliable communications between those systems and managed elements.

A typical approach to more centralized management is to form hierarchies of regional, national, and global management. Enterprise networks are also typically configured into local campuses connected by an enterprise-wide backbone network. Management systems often follows the same style of distribution.

Increasingly, the need for more centralized management is being met by employing technologies originally developed for the Web. The separate elements of an application are often distributed and made manageable via the Web by using web servers. Using the Web for distributing applications encourages the use of the Java platform, particularly for:

- Graphical user interfaces
- Upgradeable behavior interfaces in network elements
- Added-value extensions to management frameworks

Transaction Management

Managed resources in a telecom network are constantly changing. Whenever a network resource changes, that change must be propagated throughout the network management system. For example, updates to the configuration of a device must be propagated to the managed object that represents the device in the application's database, the in-memory representation of the managed object, and any representation of the device presented to an operator.

Consequently, telecom network management applications require sound transactions. Transactions in a telecom network application must be atomic, consistent, isolated, and durable (ACID) without impairing the performance of the application.

Concurrent Processing

There is no single thread of control in a telecom network management application. Telecom networks are built from a large number of devices, each of which may exhibit many autonomous threads of control. While managers are issuing requests, managed objects are issuing responses and may be emitting alarms and event notifications. With all this activity taking place simultaneously on a network, concurrent processing is a requirement for telecom network management applications.

As a result of this high level of activity, huge amounts of data need to be processed quickly. Applications require high performance to keep response times within acceptable limits. In particular, high performance is required for databases and their connectors. High performance for databases is usually achieved through efficient cache mechanisms.

To meet the need for concurrent processing, applications are often distributed to spread the load over several machines. Where the technology allows it, multithreaded programming techniques are used to represent the many threads of control in a telecom network management application.

Management frameworks based on multiprocess software architectures also help meet the need for concurrent processing. In such a framework, each software process is responsible for a single function. A multiprocess software architecture enables a management framework to benefit from a multiprocessor hardware architecture. A multiprocess software architecture also enables a single instance of a management framework to be deployed on multiple machines that communicate over a local area network (LAN) or wide area network (WAN).

The challenges of concurrent processing include setting load limits, distributing a single operation on several processors, and collecting results.

Scale

Today's telecom networks vary in size from metropolitan area networks operated by local and regional carriers, to WANs and satellite networks operated by major international carriers. Reusable code for telecom network management systems must scale to meet the demands of any situation it is used in. For example, an alarm server initially designed to manage ten pieces of equipment may be required to handle an alarm burst issued by ten thousand pieces of equipment. The architecture of such an alarm server must allow for the alarm server to be scaled up to meet such a requirement.

Managing huge quantities of data, such as performance data and configuration data, requires the synchronization and replication of many data repositories in a single network management system. This requirement is moving the focus of telecom network management away from management protocols to information modeling and representation in scalable databases.

Need for Generalized Solutions

Some problems in telecom network management, for example event correlation in mixed networks, require generalized solutions.

In a mixed network, a manager receives messages from agents that use different protocols, for example, traps from agents that use the Simple Network Management Protocol (SNMP) and event notifications from agents that use the Common Management Information Protocol (CMIP). The problem of event correlation requires a generalized solution because some event correlations need to be made between event sources that use different protocols. Any corrective action that results from a generalized correlation can then be mapped down to individual protocol domains.

Standards Compliance

Network service providers are averse to being locked into single-vendor solutions. They rely on standard interfaces to be able to mix and match equipment from different vendors. Because there are many standards bodies in the telecom domain, network service must decide which standards they should comply with.

Many of the standards published by these standards bodies overlap or are in competition with each other. With no single universal standard for telecom network management, network service providers and equipment vendors are faced with a choice of which of the available standards they should comply with. In many situations, standards compliance means interoperation between equipment in the same network that complies with different standards.

Reliability and Availability

Consumers of telecom services are accustomed to a high level of reliability and availability. Providing this level of reliability and availability places stringent requirements on the quality of code in a telecom network management system. In addition, proactive measures must be built into the software to cope with hardware and software failures.

To achieve “carrier-grade” availability, current telecommunications standards require that a platform must achieve “five nines” availability, that is be available for more than 99.999% of the time. This requirement translates to five minutes of downtime a year.

Security

Because of their strategic importance, telecom networks require secure operation. The physical separation of the call-control and management networks has helped make telecom networks more secure. However, opening up networks to customers, third-party providers, and other network service providers requires other security measures.

Security measures such as authentication, nonrepudiation, and encryption are increasingly becoming important aspects of telecom network management. These security measures must balance the need to protect the network from unauthorized operations with the need to offer access to numerous users or devices.

Telecom network management requires sophisticated access control policies. These policies determine:

- The managed objects that an operator is granted access to
- The operations that the operator is allowed to perform on those managed objects

Access control mechanisms provided by operating systems, for example, file access permissions based on UNIX users and groups, are not fine-grained enough for telecom network management. Consequently, a separately designed subsystem is required for the creation, modification, and enforcement of access control policies that have multiple parameters.

Maintenance by Network Service Providers

Telecom networks contain many different types of network elements from different manufacturers, each with its own characteristics. Because each network service provider has its own specific requirements, these elements are configured by network service providers, not by element manufacturers. To meet the continually changing demands on a network, network service providers also need to maintain their networks by, for example, balancing the load on equipment to maintain performance, or adding and removing equipment.

To enable a single administrator to maintain such a diverse array of equipment, operator interfaces of telecom network management systems must present different network elements in a logical, consistent, and intuitive way. Telecom network management systems must also enable administrators to reconfigure network elements dynamically without compromising the security of the network.

Event Forwarding, Filtering, and Logging

The states and operating conditions of managed resources in a telecom network are constantly changing. The components of the management software that represent these resources inform their peers and other relevant software components of these changes via event notifications. Making sure that notifications are delivered to all interested parties without impairing the performance of the system is a major challenge in telecom network management. Delivering information from devices based on protocols that do not support event notifications is an important aspect of that challenge.

Quick Operator Response

Quick response from operators to critical events is vital to the reliable operation of a network. To ensure that operators can respond quickly enough, there must be no bottlenecks between event sources and the operator. Eliminating bottlenecks requires:

- Efficient communication between components, usually through a software bus
- Rapid transaction processing to prevent slow database access from blocking critical events
- Effective filtering to ensure that only relevant event notifications are delivered to managers
- Clear presentation of event notifications to the operator, based on alarm severity and related to the operator’s profile

Event Logging

For diagnostic and auditing purposes, events need to be logged. Designers of telecom network management systems need to provide facilities that enable network managers to determine which events to log, where to log them, and whether they should be stored in persistent or volatile storage.

Event Filtering and Correlation

In a large and complicated network, the failure of a single piece of equipment may cause all network elements that depend on that piece of equipment to malfunction. Each malfunction may generate an event notification, leading to an event storm of several thousand notifications, all of which are the result of the same underlying problem.

Such an event storm could overwhelm the telecom network management system that is managing the network. Effective event correlation can help protect a telecom network management system against event storms by filtering out notifications related to the occurrence that would otherwise trigger an event storm.

Directories

Because of their size and complexity, some networks are organized into hierarchical management domains, thereby dividing the network into smaller subnetworks. Hierarchical management domains add another layer for a telecom network management system to manage. Management within individual subnetworks may be different from management between different subnetworks.

Directories are often used to store information on hierarchical management domains of a telecom network. Directories are also useful for storing name resolution and managed object location information, particularly in networks that are organized into hierarchical management domains.

Mobility

Mobile communications are the most rapidly expanding area of the telecom market. Mobile communications encompass two aspects of mobility:

- Personal mobility
- Terminal mobility

Personal Mobility

Personal mobility enables a user to move from terminal to terminal but to retain a consistent set of communications capabilities through a virtual home environment. A user obtains a virtual home environment by subscribing with a service provider.

Personal mobility breaks the formerly permanent binding between user and terminal. It allows a user the flexibility to register and reregister a preferred set of services to different terminals, often with optional time-of-day constraints. This flexibility greatly affects the logic for processing incoming calls. The logic is affected because the binding between service and terminal that the call processing software must make depends on factors that vary with:

- The time of day
- The date
- The location of the calling party
- The identity of the calling party

A call processing application must check each of these factors at runtime before it completes a call.

Terminal Mobility

Terminal mobility enables a user to move his or her terminal from one network-access point to another without loss of connectivity or service.

Terminal mobility breaks the permanent binding between terminal and point-of-presence on the network. It also provides a user with roaming capabilities without loss of connectivity. Terminal mobility puts new demands on the telecom networks for paging, location management, and handover.

Paging is a mechanism that allows the telecom infrastructure to resolve the exact location of a terminal, when required, without having to keep track of every roaming terminal in real time at all times. Paging is used for completing incoming calls to a mobile terminal.

Location management is a mechanism that performs real-time updates on terminal location based on predefined location areas.

Handover is a mechanism that maintains connectivity as a terminal moves from one network access point to another.

Handover during transmission of data presents a set of complex challenges to ensure that no data packets are lost. During data transmission, the transmission of data packets must be tracked to ensure that the loss of a data packet is detected and that the lost data packet is retransmitted. Retransmission of lost data packets is not required for voice communications.

Multiple Administrative Domains

A telephone call may cross multiple administrative domains, for example, two enterprise networks, two local exchange carriers, an interexchange carrier, and possibly an international carrier. Each administrative domain has its own policies, charging rates, and interpretation of quality of service. Enterprise networks may use a number of common carriers to connect local networks.

To build up service portfolios, competing carriers cross-lease transport facilities and equipment while retaining maintenance responsibility. Thus, a single network failure may affect many administrative domains with a variety of procedures and support systems. That raises the need to share relevant information to enable coordinated efforts between all administrative domains to detect, isolate and correct faults.

The challenge, perhaps not always met, is to provide a seamless end-to-end service for subscribers. Meeting such a challenge will involve standard interfaces, standard information and operational models, and appropriate bilateral agreements at network management and service levels.

Current Approaches to Telecom Network Management

Current attempts to meet the challenges of telecom network management are based on approaches such as:

- Telecommunications Management Network (TMN)
- Transaction Language 1 (TL1)
- Simple Network Management Protocol (SNMP)
- Common Object Request Broker Architecture (CORBA)

Telecommunications Management Network (TMN)

TMN is based on standards developed jointly by the International Organization for Standardization (ISO) and the International Telecommunications Union - Telecommunications Standardization Section (ITU-T).

The standards on which TMN is based define:

- A management protocol—CMIP
- Information modeling tools—Guidelines for the Definition of Managed Objects (GDMO) and Abstract Syntax Notation One (ASN.1)
- Distributed services—for example, event management, time synchronization, and testing

Information models expressed in GDMO have been defined to manage specific aspects of the TMN, for example:

- *ITU-T Recommendation G.774 - Synchronous Digital Hierarchy (SDH) Management Information Model for the Network Element View*
- *ITU-T Recommendation Q.821 - Stages 2 and 3 Description for the Q3 Interface Alarm Surveillance*
- *ATM Forum M4 Interface Requirements and Logical MIB*

However, there has been a slow acceptance of the use of TMN as an implementation technology because:

- CMIP, and the underlying Open Systems Interconnection (OSI) stack, are seen as heavyweight compared with, for example, SNMP running over Transmission Control Protocol/Internet Protocol (TCP/IP). The rapid acceptance of SNMP as the enterprise network management system of choice has further eroded support for CMIP.
- Although the information models of CMIP, GDMO, and ASN.1 are object-oriented, they do not map easily into popular object-oriented programming languages and environments. As a result, vendors have been slow to develop implementation kits. Until recently, implementors had few tools and application programming interfaces (APIs) to help them.
- Because ASN.1 is designed for efficient encoding of data for transmission over a network, it is inaccessible to mainstream application programmers. It bears little relationship to the data definition syntaxes in popular programming languages such as the Java programming language, C++, and Smalltalk. The existence of two incompatible versions of the ASN.1 standard has also alienated implementors.

Despite these development difficulties, TMN remains one of the most effective tools for managing equipment and networks because:

- The scoping and filtering features built into CMIP elegantly deliver features that can only otherwise be delivered by heavyweight on-line transaction processing arrangements.
- The event notification service, with standard headers and programmable filters, is extremely powerful.
- Internet Engineering Task Force (IETF) specifications that map CMIP to TCP/IP make OSI management available in the Internet world.

Transaction Language 1 (TL1)

TL1 is a character-oriented, command-response model based on Man-Machine Language (MML), ITU-T's human-machine interface protocol.

The protocol model offers two types of service: the carriage of commands to a network element (NE) and the carriage of responses (either solicited by a command or unsolicited) from an NE. Coherent information modeling tools are not specified for TL1. Rather, data structures are of an arbitrary nature defined by each application message set. These data structures tend to reflect a simple data store paradigm.

TL1-based systems have been popular with North American public network service providers because of the influence of Bellcore, the company that defined TL1. But because of the limited protocol model and the ad hoc nature of the message sets, it has been restricted to the interface between the element management and network element layers.

Simple Network Management Protocol (SNMP)

SNMP-based management systems have been very successful in the area of enterprise network management, especially that of corporate LANs and intranets. One reason for the success of SNMP-based management systems is that SNMP is a member of the TCP/IP family of standards. The carriage protocol, SNMP, is a request-reply protocol that reflects a simple fetch-store paradigm.

The manager is restricted to set and get operations on data items in the agent's management information base (MIB). More complex operations on an agent are requested by side effects of the set operation.

A restricted event reporting mechanism is included that allows a standardized set of six traps to be reported. Other events must be discovered by the managing node by periodic polling. The syntax of the protocol data units (PDUs) and data items in the MIB are defined as a subset of ASN.1.

Scaling and security concerns have restricted the use of SNMP to the interface between the element management and network element layers. Although SNMP v2 and SNMP v3 incorporate enhancements to address these problems, they have enjoyed only limited market acceptance.

Common Object Request Broker Architecture (CORBA)

In contrast to the protocol-centric, or bottom-up, perspective of a distributed-management protocol, CORBA has been developed from application-centric, or top-down, perspective with a focus on portability of applications. This perspective has led to the development of:

- Interface Definition Language (IDL)—an accessible C-like data-definition language
- Standardized mappings of IDL to popular programming languages
- Standardized APIs

This application-centric view and the agreement on the Internet Inter-ORB Protocol (IIOP) over TCP/IP as the common interoperability protocol have made CORBA the nonproprietary middleware of choice. The popularity of CORBA has led to numerous inexpensive implementations of object request brokers (ORBs) and development environments for nearly all platforms. Additionally, CORBA services and interfaces for database access, trading, transaction management access, and security have been defined and are rapidly being delivered by ORB vendors.

From a technical standpoint, CORBA is preferable to TMN as a distributed processing technology for the upper layers of the TMN layered model. The large number of fully-featured implementation kits supporting different programming languages, and the accessibility of IDL, make CORBA a compelling choice for implementing higher level managers. Additionally, CORBA IIOP enables legacy applications to access the next generation of web services and Java technology services hosted as EJB components.

Trends in Telecom Network Management

Current telecom network management applications make heavy use of transaction monitoring, protocol and information model translation, database access, directory access, and publish-subscribe messaging. Consequently, developing telecom network management applications has become an exercise in developing complicated server-side applications.

Today, telecom network management applications must be developed and deployed in Internet time. To meet this requirement, more and more telecom network management applications are based on the server-side application model provided by J2EE technology and the EJB component model. This trend is driven by:

- Move to Internet Protocol (IP) for core carrier networks
- Ever-increasing diversity
- Adoption of web-based technologies
- Adoption of the Java programming language
- Separation of business logic implementation from systems programming

Move to IP for Core Carrier Networks

As voice and data networks converge, IP is likely to become the underlying protocol throughout the next-generation telecom carrier network. Circuit-switched voice traffic will become just one application among many layered over an IP-based telecom carrier network.

The core of the next generation network will be primarily built from routers and servers. Core applications, including switching and call processing applications, are already being moved from switches to servers.

As a result, applications must be composed of distributed software objects on an inherently IP-based communications infrastructure.

Ever-Increasing Diversity

Diversity of equipment, supported protocols, and applications is increasingly a feature of today's telecom networks. Telecom customers typically access services through a variety of media, such as wireless, wireline, cable, or broadband networks. Consequently, no single technology—whether it is TMN or SNMP— can provide a complete telecom network management solution. The many different protocols used for communication with network elements will remain in use for many years to come.

As a result of the diversity of today's telecom networks, building network management solutions is increasingly a problem of integration. Integration is no longer a single-shot process but a continuous one in which all roles involved must be clearly specified. As IP-based and circuit-switched networks converge, software integration becomes an even more vital part of telecom network management. For example, end-to-end path management requires integration of existing network management systems for the circuit-switched part and specific add-ons for the IP-based part of the network.

Adoption of Web-Based Technologies

Technologies developed initially for the Web are increasingly a feature of telecom network management applications. For example, web browsers are the user interface of many telecom network management applications, providing not only the benefits of a familiar graphical user interface, but also the possibility of easy remote access via the Hypertext Transfer Protocol (HTTP).

The EJB-based model simplifies the adoption of web-based technologies because EJB-based management frameworks provide ready access via web-based technologies to legacy data sources.

Adoption of the Java Programming Language

The Java programming language is becoming increasingly popular for new telecom network management applications. In part, this situation is due to widespread uptake of the Java language as the programming language for the Internet and for the development of enterprise applications.

Programmers' experience has shown that the Java programming language helps shorten application development times compared to other languages such as C++, while approaching the performance of these languages.

Not only is the Java language object oriented, it is also designed for networking, in particular for networking over the Internet. Furthermore, the Java language was first released to coincide with the emergence of the Internet as a popular networking technology. Distributed object computing concepts are built into the language. As a result, Java objects are able to make use of TCP/IP services over the Internet.

Embedded, real-time, and server-based versions of the Java language are available. These versions are useful for many classes of telecom network management applications. The built-in server side component model provided by the EJB component model further enhances the suitability of the Java language for many classes of telecom network management applications.

The Java language also offers a rich set of standard fully featured APIs. The many vendors of computer aided software engineering products for the Java language collectively provide an extensive set of development tools and test suites.

Separation of Business Logic Implementation From Systems Programming

The development of telecom network management applications has become an exercise in developing complicated server-side applications. The development of such applications benefits from a clear separation between the implementation of business logic and systems programming. This separation promotes code reuse, enabling telecom network application vendors to capitalize on their code for business logic.

One way this separation enables telecom network application vendors to capitalize on their code for business logic is by using this code in different products. For example, the same code can be used in a simple low-cost solution and in a powerful, higher-cost solution. Reusing code in this way requires a network management framework versatile enough to support simple configurations with limited features and complex fully featured configurations.

Architecture of a Management Framework Based on EJB Technology

Server side development with J2EE technology and EJB components enables multitier telecom network management services to be rapidly deployed and easily enhanced while reducing cost and complexity.

Overview of Application Architecture Based on EJB Technology

An application architecture based on EJB technology simplifies the development of multitier applications. Multitier application architecture extends the standard two-tier client and server model by placing a multithreaded application server between the client and the database. Client programs communicate with the database through the application server by using high-level platform-independent calls. The application server responds to the client requests, makes database calls as needed into the underlying database, and replies to the client program as appropriate.

An application architecture based on EJB technology hides complexity and enhances portability by separating the various complexities inherent in multitier enterprise applications, such as transaction management, life-cycle management, and resource pooling, from specifics such as business logic and user interfaces.

To achieve its goal of hiding complexity and enhancing portability, the EJB specification defines the following architectural elements and the interactions between them:

- Component based on the EJB specification (“EJB component”)
- Container for EJB components (“EJB container”)
- Client enabled with EJB technology (“EJB client”)
- Connector for an EJB container (“EJB connector”)

These architectural elements are related as shown in FIGURE 1. They are described in the following subsections.

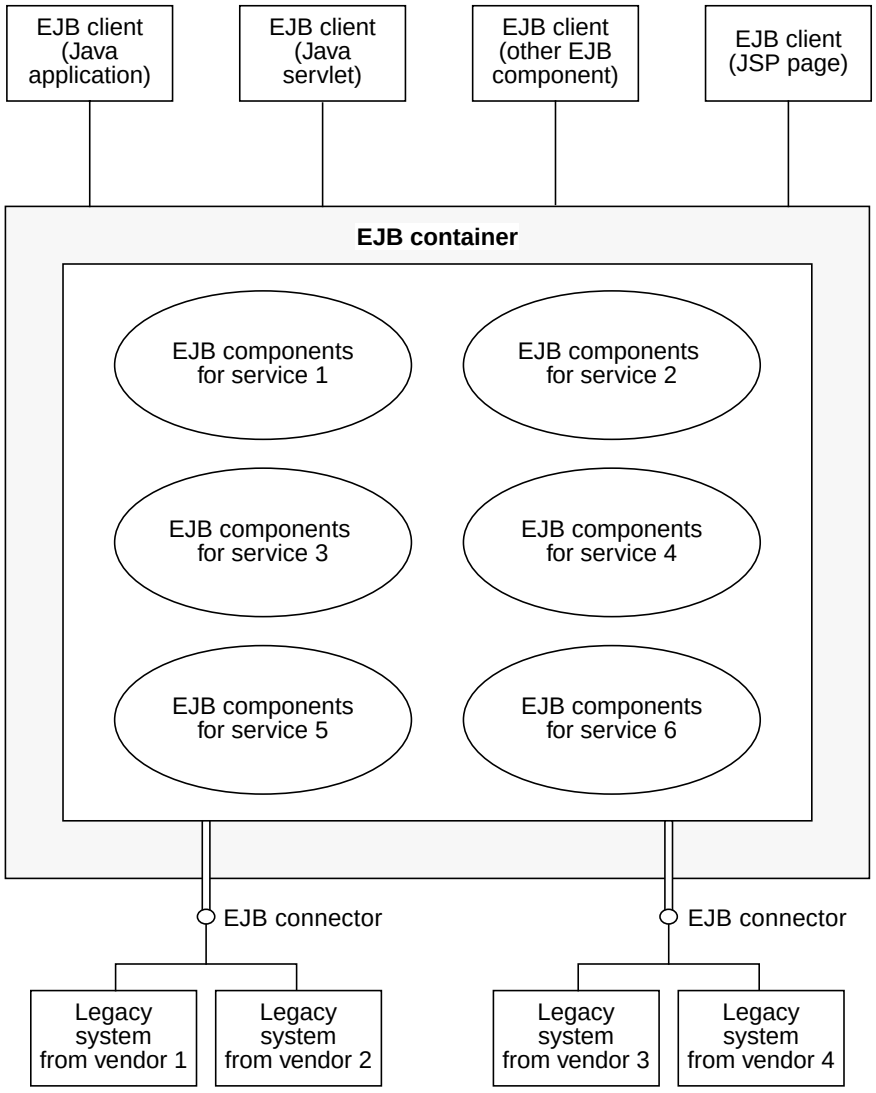


FIGURE 1 Architecture of an Application Based on EJB Technology

EJB Component

An EJB component is a Java class that encapsulates functionality that can be reused in many applications. This class is the only significant class that an EJB component development engineer writes. Other classes required when an EJB component is deployed are generated automatically. For more information see “Application Deployment” on page 26.

Container vendors implement containers and connectors to conceal complexity and promote portability. They also provide tools that automatically generate behavior for EJB components that makes them secure, transactional, and accessible over a network.

By buying an EJB container and tools from a container vendor, telecom network management application vendors are free to concentrate on providing business logic. They concentrate on providing business logic by writing EJB components specifically for management functions

EJB Container

An EJB container provides a runtime environment for EJB components. The container mediates between EJB clients and EJB components, providing services transparently to both. Container mediation enables many component behaviors to be specified at deployment time, rather than in application program code.

An EJB container co-ordinates and schedules operating system services on behalf of EJB components without exposing these services to the EJB components. An EJB container removes the need for a component developer to interact directly with operating system APIs by providing services for:

- Process management
- Resource management
- Multithreading
- Connection pooling
- Caching

An EJB container also provides services for server-side component hosting:

- Directory and name resolution
- Persistence
- Transaction monitoring
- Life cycle management
- State management
- Security

EJB Client

An EJB client application accesses EJB components through an EJB container. The J2EE specification requires EJB containers to support clients that are any of the following:

- A local application
- A remote application
- A Java servlet
- A page created with the JavaServer Pages™ technology (“JSP™ page”)
- Another EJB component, which may be in the same EJB container or another EJB container

Enterprise JavaBeans Specification, Version 2.0 requires EJB components to be fully accessible to CORBA clients. This requirement enables a CORBA client that is written in any language to use services of EJB components. To meet this requirement:

- EJB components must publish their services to CORBA clients through IIOP.
- EJB containers must support the propagation of CORBA transactions and security contexts to services of EJB components.

EJB Connector

The EJB connector architecture in *Enterprise JavaBeans Specification, Version 2.0* specifies a mechanism for EJB components to access data from legacy systems. The EJB connector architecture defines a portable service API to plug into existing legacy systems such as:

- Transaction processing (TP) systems
- Enterprise resource planning (ERP) systems
- Databases
- Protocol stacks
- Other data sources

EJB connectors promote flexibility by enabling a variety of implementations of specific services. They provide a standard mechanism, with a standard API, for EJB components to reach outside their container to obtain data. EJB connectors provide benefits for EJB container vendors and maintainers of legacy systems as follows:

- An EJB container vendor extends the container only once to support the connector architecture, thereby enabling the container to be connected to multiple legacy systems.
- The maintainer of a legacy system provides one standard connector for the legacy system, and can then connect that system to any EJB container that supports connectors.

EJB-Technology-Based Application Architecture in Network Management

A network management framework offers many services to agents and managers. Such services typically include:

- Event logging
- Alarm management
- Topology management
- Metadata
- Managed object access control
- Query and reporting

In a management framework based on EJB technology, the services the framework offers to agents and managers are implemented as EJB components.

Connectors perform protocol translation to enable legacy managers and agents to access these services and to interoperate with each other. In telecom network management, translation is most frequently required between:

- CMIP
- SNMP
- TL1
- Proprietary protocols

Client applications of the management framework use the services that the framework provides. Examples of such client applications include:

- Java-technology-based managers that interact with the EJB container through Remote Method Invocation (RMI) over IIOP
- CORBA managers that interact with the EJB container through RMI over IIOP
- Web-based managers that interact with the EJB container through HTTP

The architecture of a management framework based on EJB technology is shown in FIGURE 2.

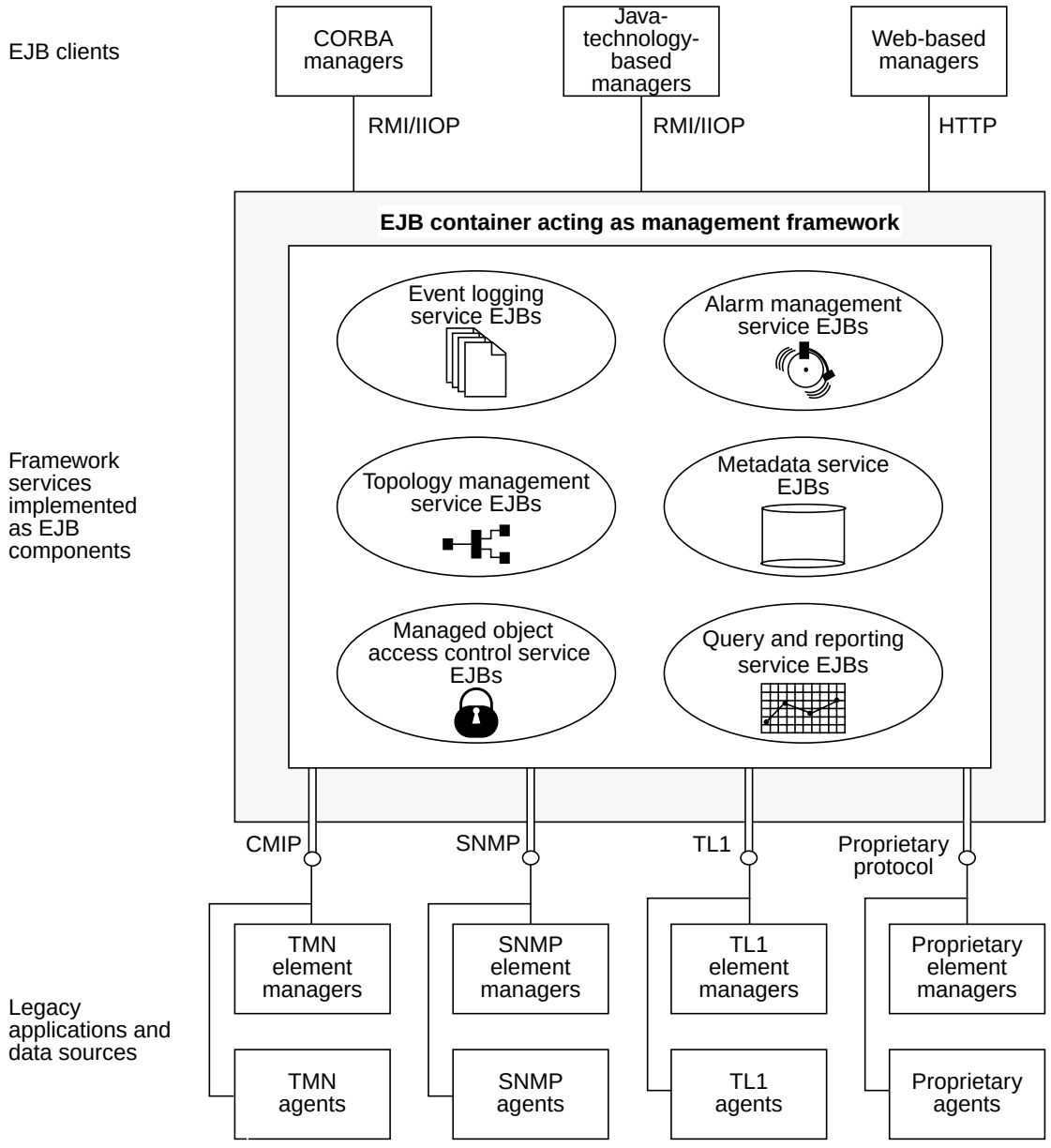


FIGURE 2 Architecture of a Management Framework Based on EJB Technology

Benefits of a Management Framework Based on EJB Technology

A management framework based on EJB technology simplifies telecom network management applications by handling many details of application behavior automatically, without complex programming. By employing an EJB-based management framework, application developers no longer need to develop applications against specific operating-system APIs or specific vendors' middleware APIs.

A management framework based on EJB technology handles many details of application behavior automatically by:

- Basing applications on standard, modular components
- Providing, through EJB containers, a complete set of system services to those components

Benefits of Standard Components

The component-based architecture of an EJB-based management framework provides competitive choices for developers of telecom network management applications through:

- Encapsulation of business logic
- Simplified application development

Encapsulation of Business Logic

The EJB component architecture encapsulates business logic in EJB components, enabling a simplified approach to multitier application development. Encapsulating business logic in EJB components conceals application complexity and enables component developers to focus on business logic, thereby shortening development time scales and reducing costs.

EJB technology gives developers the ability to model the full range of objects found in telecom network management by defining the following distinct types of EJB components:

- **Session beans.** Session beans represent behaviors associated with client sessions, for example, session information that enables a mobile user to resume a session when moving from one device to another.

- **Entity beans.** Entity beans represent collections of data, for example, alarm records or topology nodes, and encapsulate operations on the data they represent. Entity beans are intended to be persistent, surviving as long as the data they are associated with remains viable.
- **Message-driven beans.** Message-driven beans represent Java Message Service (JMS) consumers that implement business logic on a server. Message-driven beans are invoked by the EJB container as the result of the arrival of a JMS message. Clients communicate with message-driven beans by sending messages to a JMS destination.

Simplified Application Development

To keep pace with the demand for new and enhanced telecom services, developers of telecom network management applications need to be able build new solutions from existing components.

The component-based architecture of a management framework based on EJB technology enables developers of telecom network management applications to assemble applications from a combination of standard, commercially available components and their own custom components. Existing components can be reconfigured and redeployed without reprogramming them.

Industry experience has shown that this approach has resulted in faster development time, better quality and maintainability, and portability across a range of enterprise platforms. As a result, programmer productivity is increased, strategic use of computing resources is improved, and the return on technology investments is increased.

The life-cycle for applications built from standard components consists of the following phases:

- Component development
- Application assembly
- Application deployment
- Runtime administration

The roles in this life cycle are summarized in FIGURE 3.

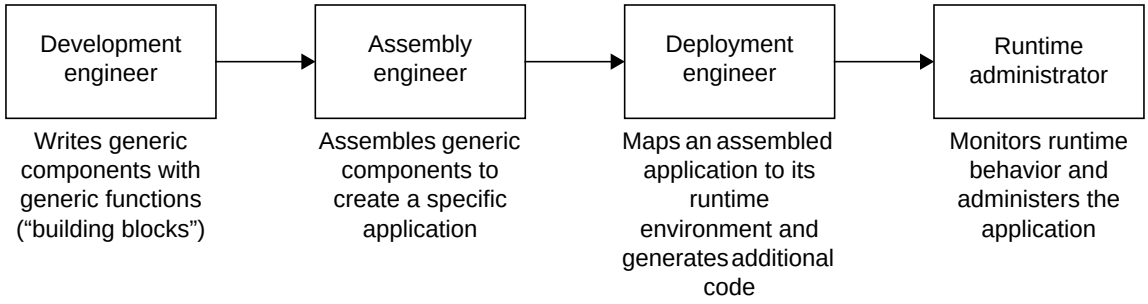


FIGURE 3 Roles in the Life Cycle of an Application Based on EJB Technology

Component Development

The development engineer develops an EJB component by writing the business logic in the EJB component, specifying its home and remote interfaces. Because EJB components are developed to do fine-grained tasks, the development engineer does not need any knowledge of how an EJB component will be assembled or deployed in completed applications.

The development engineer creates and partially populates the deployment descriptor of the EJB component. A deployment descriptor specifies the requirements of the component’s methods for services provided by an EJB container (for example, transaction management, security, and multithreading).

Deployment descriptors are typically produced by an interactive tool and expressed as an Extensible Markup Language (XML) document type definition (DTD).

Application Assembly

The assembly engineer creates complete applications by using scripts to assemble components together. From the same set of components, assembly engineers can create applications to meet many different market requirements by assembling components in different ways.

The assembly process:

- Resolves dependencies between components
- Organizes security roles
- Completes the deployment descriptor by specifying which methods of each component will be invoked on which other components

The completed deployment descriptor specifies how the components will work together in the assembled application.

Application Deployment

The deployment engineer maps the completed application to the runtime environment and uses code generation tools to generate classes that the application requires.

Mapping the completed application to the runtime environment involves:

- Mapping transactional, security, and threading functions to the capabilities of the run-time environment
- Mapping the capacity and performance parameters of the container to the capabilities of the run-time environment
- Providing parametric information (analogous to environment variables) required for each EJB component

The following classes are generated for an application at deployment time:

- Implementations of the home and remote interfaces of each EJB component
- The instance of the home class for each EJB component
- Other supporting classes as required, for example, Java objects that are independent of EJB components
- Client-side stubs (optional)
- Resource adapter classes to map application resources to specific resources in the runtime environment, for example:
  - TP systems
  - Relational database management systems (RDBMSs)
  - ERP systems
  - Protocol drivers

Runtime Administration

The runtime administrator monitors runtime behavior and administers the application.

Benefits of Standard Containers

EJB container technology extends the power and portability of EJB components by defining a complete infrastructure that includes standard clients and service APIs for their use. The benefits of such an infrastructure include:

- Easy enterprise application integration
- Distributed object access
- Scalability
- Consistency
- Minimization of system programming
- Enhanced support through telecom-class containers

Easy Enterprise Application Integration

Today's telecom networks contain many different types of equipment, which interoperate with each other through a variety of communications protocols and enterprise applications. As a result, enterprise application integration is a major challenge in network management. The J2EE platform and EJB architecture provide enterprise application integration (EAI) capabilities for server-side applications. Because the time to market for network management solutions is very short, new network management systems must build on existing platforms by adding specific parts in compliance with integration rules defined by the framework.

EJB container technology eases enterprise application integration through its support for legacy data processing and data sources. EJB containers support legacy data processing and data sources through connectors. Connectors provide a uniform interface to legacy data so that an EJB component can implement one interface and operate with applications from various vendors.

The flexibility of EJB container technology also eases enterprise application integration. Deploying an EJB container in one part of the network does not require all of the software that interacts with the EJB applications to be upgraded. A network service provider does not have to integrate every EJB component with legacy applications and data sources. Only one adaptor or interface needs to be written for each source of legacy data. Any other EJB component can access the methods of the adaptor or interface. Because the standards are open and public, customers are never locked into a closed platform.

Distributed Object Access

An EJB container provides distributed object access to any server system through IIOP. Built-in support for CORBA standards such as IDL and IIOP makes it easy to integrate EJB components with existing CORBA services. Moreover, the CORBA component model specification from the Object Management Group (OMG) includes Sun's *Enterprise JavaBeans Specification, v1.1* by reference.

As a result of its built-in support for CORBA standards, the EJB model is becoming an increasingly popular implementation mechanism for CORBA services. Because many new CORBA implementations now use the EJB model, many ORB vendors have also become EJB container vendors.

Scalability

The EJB container architecture scales from constrained environments near the network edge to large distributed servers in the data center. The power of the technology is that the EJB component development model and application programming model are the same for pared-down or very large EJB containers.

The deployment engineer chooses where and how it makes the most sense to deploy the EJB components with full knowledge of the runtime environment. By contrast, neither the EJB component development engineer nor the assembly engineer need to have any knowledge about the deployment environment. The size of the server, the number of other EJB components running, and any other vendors' components in the EJB container do not matter to EJB component development engineers. EJB component development engineers do not have to factor in runtime constraints because they only program to the abstract EJB container APIs.

The EJB container architecture also enables deployment engineers to use the same components in small-scale and full solutions. For example, an EJB component representing an alarm can be deployed on a Linux system with a free application server when low cost is a requirement and on powerful servers using an industrial application server where centralized management of many devices is a requirement.

The support of EJB containers for distributed computing helps the infrastructure to scale by enabling resources to be allocated where they are needed. Low-end entry servers or resource-constrained servers can run part of an application and use the IP network to access the services of EJB components in other, remote EJB containers more capable of heavy computation.

The EJB container architecture also enables large amounts of legacy data to be managed because the Java Database Connectivity (JDBC™) API and the Java Naming and Directory Interface™ (JNDI) API of the J2EE platform are scalable.

Consistency

The EJB specification provides a consistent mechanism for building management frameworks. A management framework based on EJB technology provides ready access to the benefits of the Java programming language throughout the network management domain:

- Manager applications can be developed as graphical client applications built on JavaBeans™ components.The JavaBeans components can be clients of EJB server-side components
- Management framework services can be built as EJB components.
- Agent services can be built as EJB components.

In addition, because all frameworks are similar, developers do not need to relearn how to program and maintain every new framework they work on.

Minimization of System Programming

The J2EE application model and EJB architecture minimize system programming by building various complexities inherent in enterprise applications, such transaction management, life-cycle management, and resource pooling, into the platform and providing them automatically to the components the platform supports. Component and application developers are free to focus on business logic.

The platform provides the following to telecom network management applications:

- Container-managed persistence
- Transaction management
- Protocol and information model translation
- Database access
- Directory access
- Publish-subscribe messaging

Container-Managed Persistence in Network Management

Many managed objects, for example, log records, in a telecom network management system need to be stored persistently. Container-managed persistence hides the complexity of supporting persistence by shifting responsibility for persistence to the container.

Managed objects in telecom network management are often defined using object-oriented syntax and are, therefore, able to use container-managed persistence. For example, managed object classes defined in the following formats can be easily mapped to JavaBeans components:

- GDMO
- Common Information Model (CIM) Managed Object Format (MOF)
- SNMP MIBs

Once represented as JavaBeans components, these object classes can easily be mapped to RDBMS tables by using the JDBC API.

A further benefit is the ability to use container-managed persistence to facilitate operations on managed objects, for example:

- For Common Management Information Service (CMIS) requests such as `M-CREATE`, `M-SET`, or `M-DELETE`, container-managed persistence can be used to synchronize the in-memory EJB component representation of managed objects to their representation in persistent storage.
- For filtered CMIS requests, container-managed persistence can be used to generate the structured query language (SQL) `WHERE` clause to match the specified CMIS filter.

Transaction Management in Network Management

Telecom network management requires many transactionally sound operations. For example, a scoped CMIS `M-SET` or `M-DELETE` request with atomic synchronization requires that the managed object database be able to roll back to a previous state if the request fails on any of the selected managed objects.

To satisfy the transactional requirements of a CMIP agent represented by an EJB component, all the deployment engineer needs to do is declare these requirements for the `mSet` and `mDelete` methods at deployment time. The CMIP agent must support a distributed transaction protocol, such as that defined by *ITU-T Recommendation X.860 - Open Systems Interconnection - Distributed Transaction Processing: Model*.

Declaring these requirements causes the EJB container to generate the appropriate transactional context for each invocation of each of the `mSet` and `mDelete` methods. This transactional context enables the underlying transaction system to perform the operation with the required synchronization.

Protocol and Information Model Translation in Network Management

The management of mixed networks containing many devices requires translation between the different protocols that the devices use for communication. For example, translation between the SNMP and CMIP protocols is required in a network where an SNMP device is managed by a CMIP manager.

An framework based on EJB Technology removes the need for direct translation from one protocol to another. Instead, managers and agents that use many different protocols can all communicate with each other through a shared management framework. EJB connectors perform the protocol and information model translation required to enable agents and managers to access the management framework.

Database Access in Network Management

Telecom network management applications frequently require access to data stored in a data warehouse, for example, log records.

The JDBC API enables EJB components to access data in a data warehouse through SQL. Each record in the data warehouse can be represented as an entity bean.

Information such as alarm records and topology nodes can also be stored in databases that are tuned for specific data and are not based on SQL. EJB connectors provide access through existing specific APIs to data stored in such databases.

Directory Access in Network Management

In addition to providing name resolution, directories are frequently a repository for a large amount of information required in telecom network management, for example:

- Policy information
- Operator and authentication credentials
- Access control information

The JNDI API provides name resolution for managed object lookup in a telecom network management framework based on EJB technology.

The EJB architecture eases the implementation of a directory-centric approach for enabling policy-based networking and quality of service (QoS) enforcement.

Publish-Subscribe Messaging in Network Management

In a telecom network management system, agents send messages (for example, event notifications) to inform managers of important events on the network. In a network management framework based on EJB technology, such messages are handled by message-driven EJB components, which implement publish subscribe message queues. These EJB components provide asynchronous messages through the JMS API. This API provides a uniform interface to other EJB components.

Publish-subscribe messaging through the JMS API also enables loose coupling between systems. Loose coupling provides one way of enabling a network management and an operations support system (OSS) to interoperate. For example, an alarm management system and a trouble ticket system can interoperate by exchanging messages asynchronously via a single publish-subscribe messaging bus.

Enhanced Support Through Telecom-Class Containers

Telecom network management demands high availability and reliability of network management frameworks. EJB-based network management frameworks achieve high availability and reliability through telecom-class EJB containers.

Features of a Telecom-Class EJB Container

A telecom-class EJB container has high availability (HA) and redundancy features built-in, such as:

- Primary and secondary containers
- Hot standby EJB components with fully replicated state information
- Active checkpoints of the state of EJB components
- Clustered container deployment with availability monitoring and failover

- Support from an underlying HA framework that provides APIs for:
  - Cluster membership
  - Cluster configuration
  - Distributed processing environment (DPE) for messaging
  - Availability monitoring

Furthermore, container vendors could offer additional value-added features for telecom-class EJB containers, for example, load balancing.

The architecture of a highly available network management framework based on telecom-class EJB containers is shown in FIGURE 4.

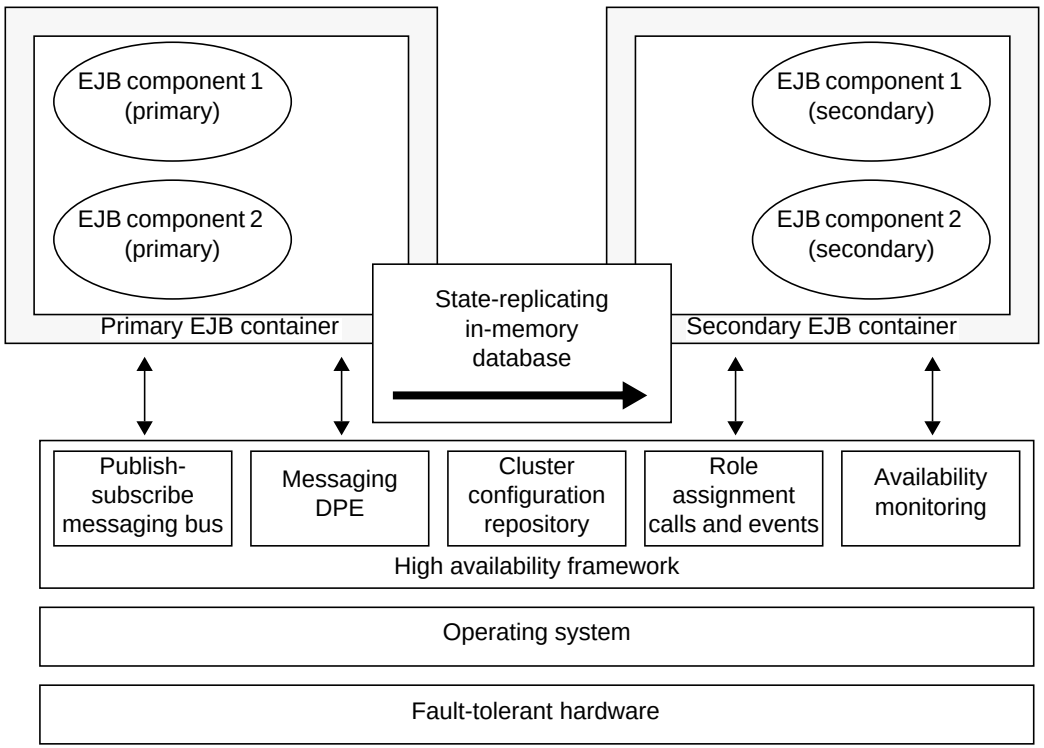


FIGURE 4 Network Management Framework Based on Telecom-Class EJB Containers

Benefits of Container-Managed High Availability

Container-managed high availability offers advantages over bean-managed high availability to telecom network management application vendors and container vendors.

Telecom network management application vendors benefit because:

- Container-managed high availability hides the complexity of HA support by shifting responsibility for HA to the container vendor.
- Telecom network management application vendors can employ application developers who are experts in business logic. Such developers do not need to be experts in high availability because they are not required to code to HA APIs.
- Hiding the complexity of HA support results in a much shorter time to market for applications.

Container vendors benefit because they can offer HA support as a value-added feature to their customers. Support for HA logically belongs in the container, not in the application, because the development of HA frameworks is a systems programming activity.

Telecom network management application vendors and container vendors benefit because there is no need to write applications that combine business logic with explicit calls to HA APIs. Writing such applications is very difficult, time consuming, and error prone.

Avoiding Potential Drawbacks of the EJB Model

Many of the drawbacks of the EJB model are not inherent in the EJB model itself, but are the result of a particular application server implementation. To avoid these drawbacks, all that application developers need do is choose a scalable, reliable, highly available, high performance application server that supports all J2EE APIs.

Sample Network Management Applications

The EJB component model can be used to develop telecom network management applications for:

- Network element management
- Wireless network management
- Alarm management
- Event correlation
- Policy administration
- Quality of service (QoS) enforcement

Network Element Management

A network element management system controls and monitors the states of managed resources in a network.

A network element management system based on EJB components integrates legacy databases, directories, element management systems, network management systems, and network elements. Such an integration enables management information for a mixed network to be presented through a browser-based operator interface.

An EJB container provides the framework for an element management system based on EJB technology. Required element management functions are provided by subsystems implemented as sets of EJB components, for example alarm management and topology subsystems.

In this example, entity beans represent data objects in an element management system, such as:

- Alarm logs
- Alarm records
- Topology nodes
- Managed objects (MOs)

JDBC drivers connect the EJB container to relational databases, providing a single consistent point of access to all records for many different types of network elements.

The JNDI API provides lookup for:

- Managed objects stored in managed object directories
- Operator access privileges and authentication information stored in access control directories

Legacy network and element management systems expose proprietary APIs that the EJB-based management framework accesses through EJB connectors.

EJB connectors perform protocol and information model translation, enabling EJB components in the EJB container to interact with agents, network management systems (NMSs), and element management systems (EMSs) that communicate by using:

- CMIP
- SNMP
- American Standard Code for Information Interchange (ASCII) character interchange
- TL1

FIGURE 5 shows an example of an network element management application based on EJB components.

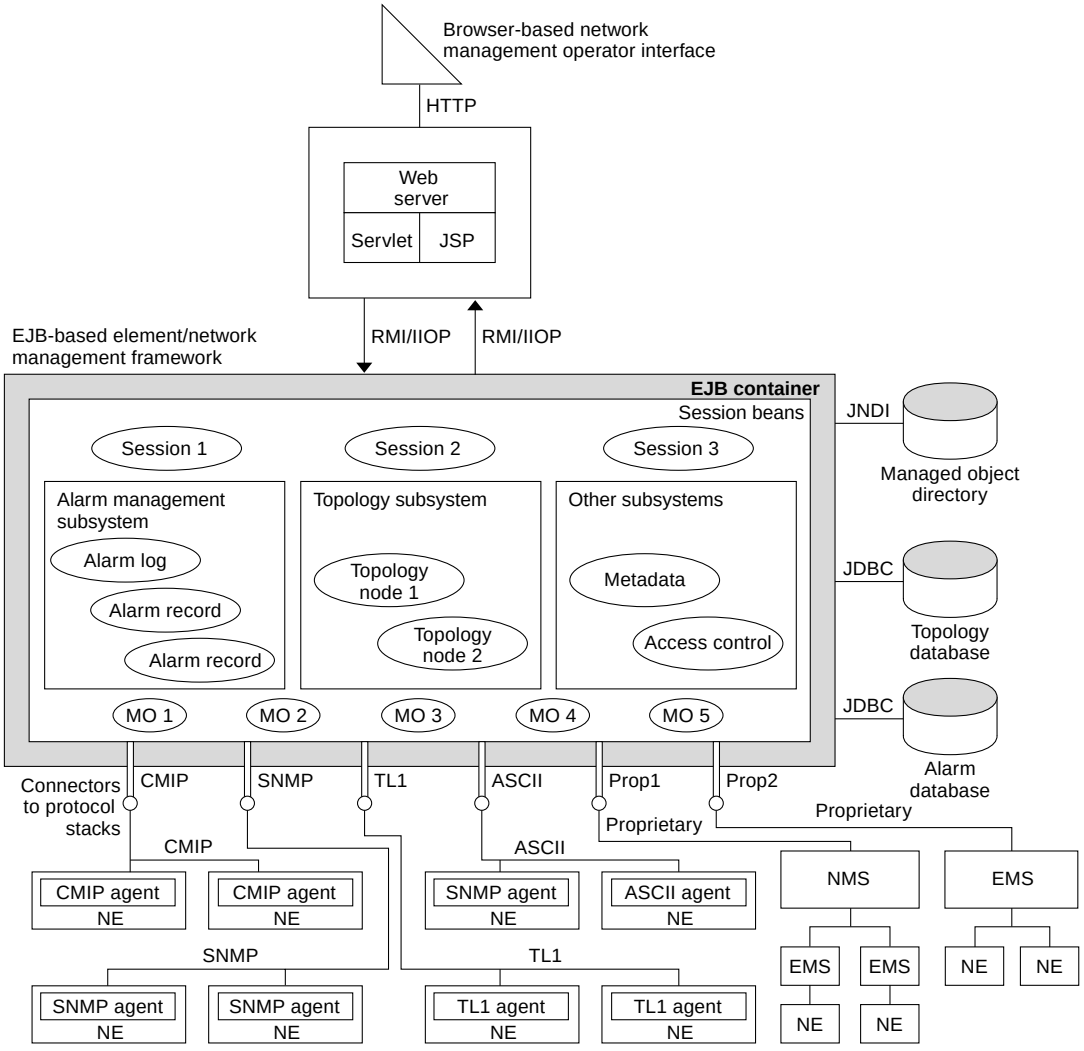


FIGURE 5 Network Element Management Based on EJB Components

Wireless Network Management

A wireless network management system controls and monitors the states of managed resources in a general packet radio service (GPRS) wireless network.

In a wireless network management system based on EJB technology, an EJB container provides the framework for each operation and maintenance center (OMC), for example:

- Radio access network (RAN) OMC
- Core network OMC
- Packet data network OMC

In this example, entity beans represent managed objects in each OMC as follows:

- In the RAN OMC:
  - Wireless network cells
  - Base transceiver stations (BTSs)
  - Base station controllers (BSCs)
- In the core network OMC:
  - Mobile switching centers (MSCs)
  - Equipment identity registers (EIRs)
  - Authentication, authorization, and accounting (AAA) systems
  - Home location registers (HLRs)
  - Visitor location registers (VLRs)
- In the packet data network OMC:
  - Serving GPRS support nodes (SGSNs)
  - Gateway GPRS support nodes (GGSNs)
  - Routers

EJB connectors perform protocol and information model translation, enabling EJB components in the EJB containers to interact with network elements that communicate by using different protocols:

- Network elements in the radio access network and the core network typically exchange network management information by using CMIP.
- Routers in the packet data network typically exchange network management information by using SNMP.

FIGURE 6 shows an example of a wireless network management application based on EJB components.

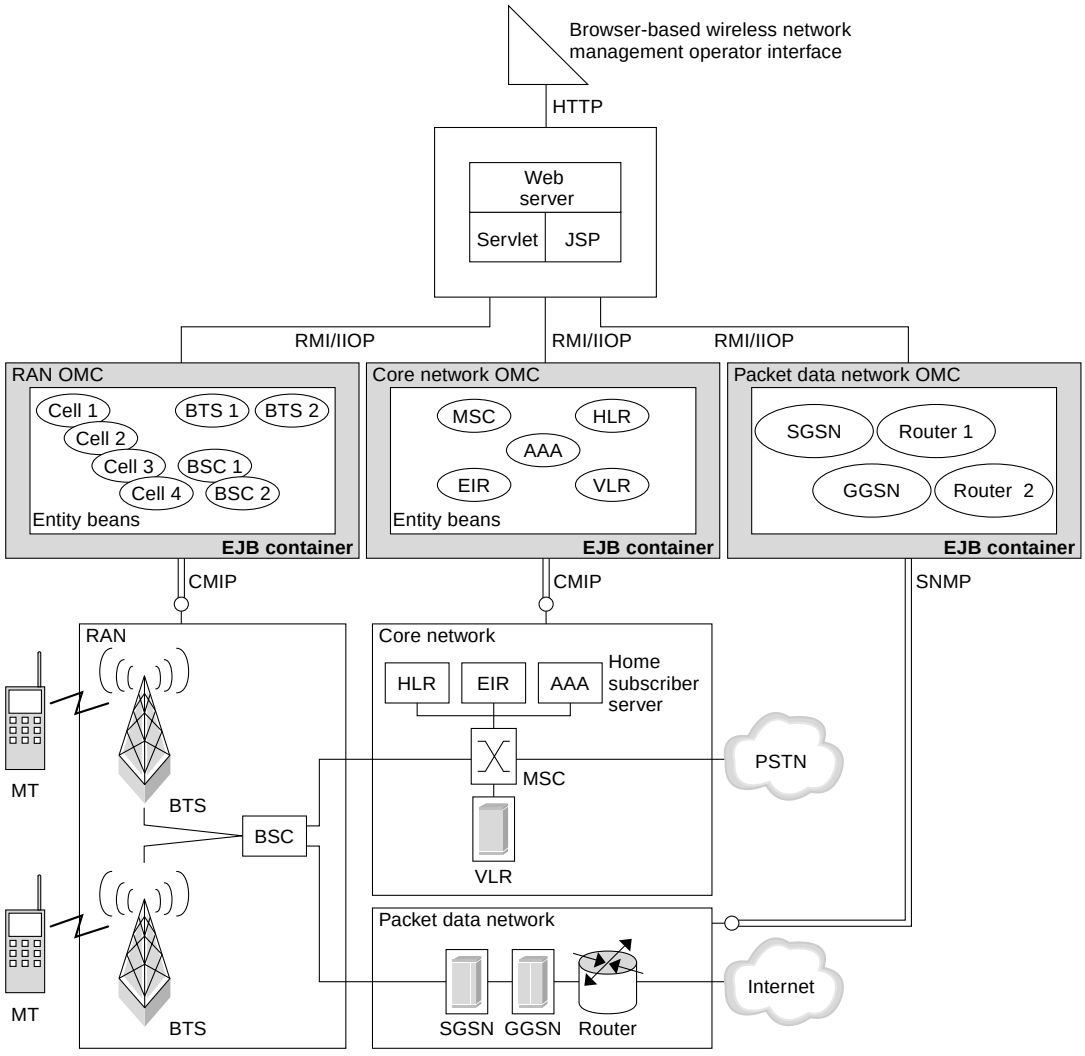


FIGURE 6 Wireless Network Management Based on EJB Components

Alarm Management

An alarm management system receives alarm messages from network elements, filters them, and presents them to operators who are interested in receiving them. An alarm management system based on EJB components simplifies the management of alarm messages from many different types of network elements in a mixed network.

Messages in the form of CMIP event notifications, SNMP traps, and Java Management Extensions (JMX™) events are published to a publish-subscribe messaging bus. A JMS driver forwards these messages to an EJB container. Messages that correspond to alarms are stored in an alarm management database. A JDBC driver connects the EJB container to the alarm management database. Alarms from many different sources are stored in and retrieved from a single alarm management database.

An entity bean is created to represent an alarm record only when required. An entity bean is required when operations are performed on an individual alarm record (for example, acknowledging or clearing alarms). If necessary, an alarm is retrieved from the alarm management database when the entity bean that represents the alarm is created. An alarm record needs to be retrieved from the alarm management database if an entity bean to represent the alarm was not created when the EJB container received the alarm.

Individual alarm records that are logged for statistics reporting purposes only are represented as array-valued attributes of an alarm log entity bean.

Filtering logic implemented in the entity beans enables alarms to be filtered so that only alarms that an operator has subscribed to receive are forwarded to the operator. This filtering logic also helps ensure that each alarm is written to the appropriate alarm log.

Alarm messages that the EJB container receives are published to a publish-subscribe messaging bus. A JMS driver forwards these messages to any operators that have subscribed to receive them. The alarm management system presents alarms to operators in a web browser.

Note – As an alternative to a publish-subscribe messaging bus, an EJB container can receive alarms by an asynchronous call to a notifier session bean. The use of a notifier session bean is not illustrated in this example.

FIGURE 7 shows an example of an alarm management application based on EJB components.

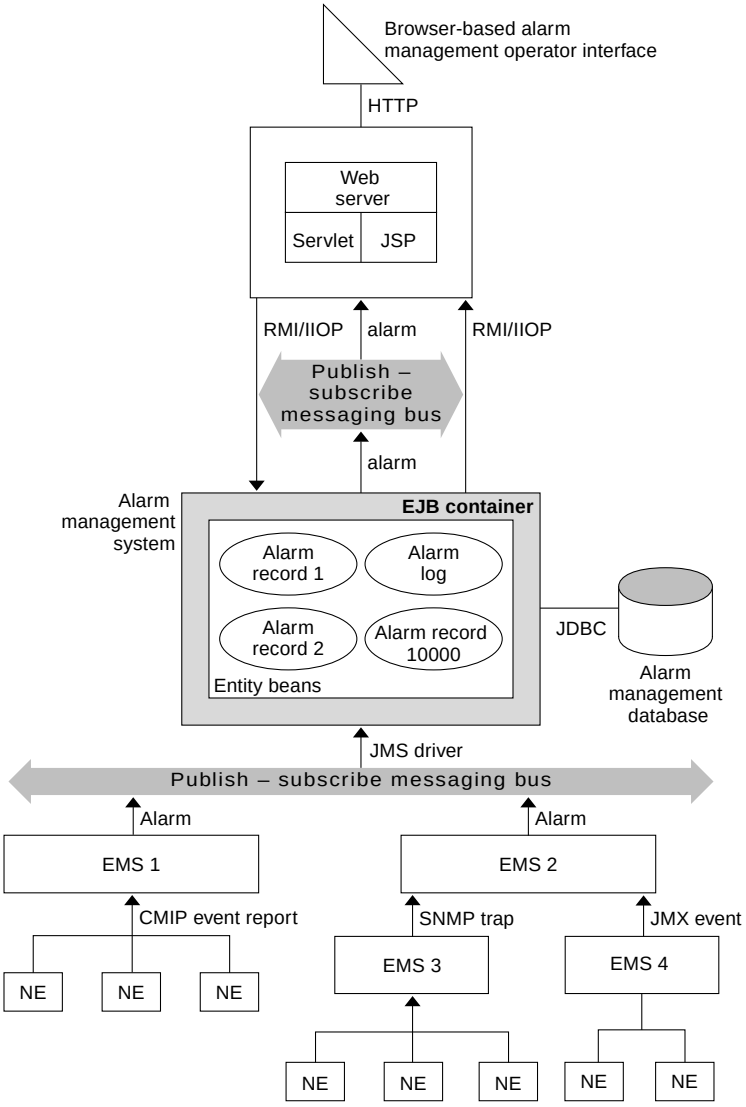


FIGURE 7 Alarm Management Based on EJB Components

Event Correlation

An event correlation system receives event notifications from network elements and analyzes each event notification to determine which other event notifications it is related to. An event correlation system helps operators respond quickly to a failure that leads to an event storm by:

- Presenting summarized and correlated event notifications
- Suggesting corrective action based on the root cause of the failure that led to an event storm

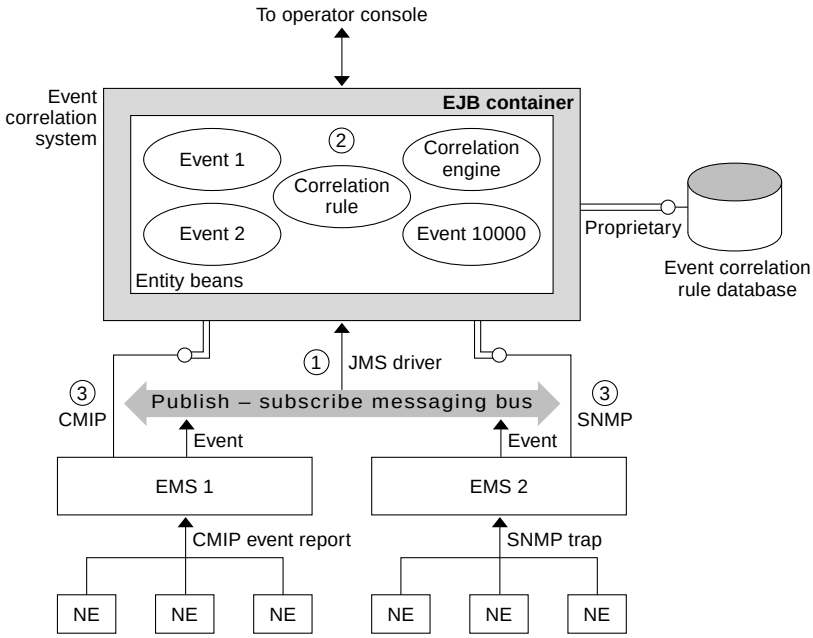
Messages in the form CMIP event notifications and SNMP traps are published to a publish-subscribe messaging bus. A JMS driver forwards these messages to an EJB container.

A correlation engine EJB component performs event correlation between notifications contained in the messages that the EJB container receives. Because these messages are received from sources that use different protocols, the correlation engine EJB component performs event correlation in the abstract domain.

The correlation engine EJB component bases its event correlation on correlation rules that are represented as EJB components. An EJB connector connects the EJB container to an event correlation rule database.

EJB connectors map corrective actions from correlation results in the abstract domain onto individual protocol-dependent primitives. These primitives are propagated to element management systems via EJB connectors.

FIGURE 8 shows an example of an event correlation application based on EJB components.



1 Events are received in the EJB container from different sources
2 A correlation engine bean does event correlation in the abstract domain
3 Actions are propagated via connectors

FIGURE 8 Event Correlation Based on EJB Components

Policy Administration

A policy administration system sets priorities for network traffic according to factors such as the type of traffic, the application that is sending the traffic, and the user of the application.

To create policies to be enforced on the network, an administrator uses a browser-based interface of a policy server. The policies are stored in a policy directory. They are represented as entity beans in an EJB container. The JNDI API and the Lightweight Directory Access Protocol (LDAP) provide lookup services for this directory.

Topology node information required for enforcing policies is stored in a topology database. A JDBC driver connects the EJB container to the topology database.

An EJB connector enables policy information represented as entity beans to be distributed to Common Open Policy Service (COPS) client applications.

EJB connectors also enable network management information to be exchanged between the EJB container and agents by using:

- SNMP
- ASCII character interchange though a command line interface (CLI)

FIGURE 9 shows an example of a policy server based on EJB components.

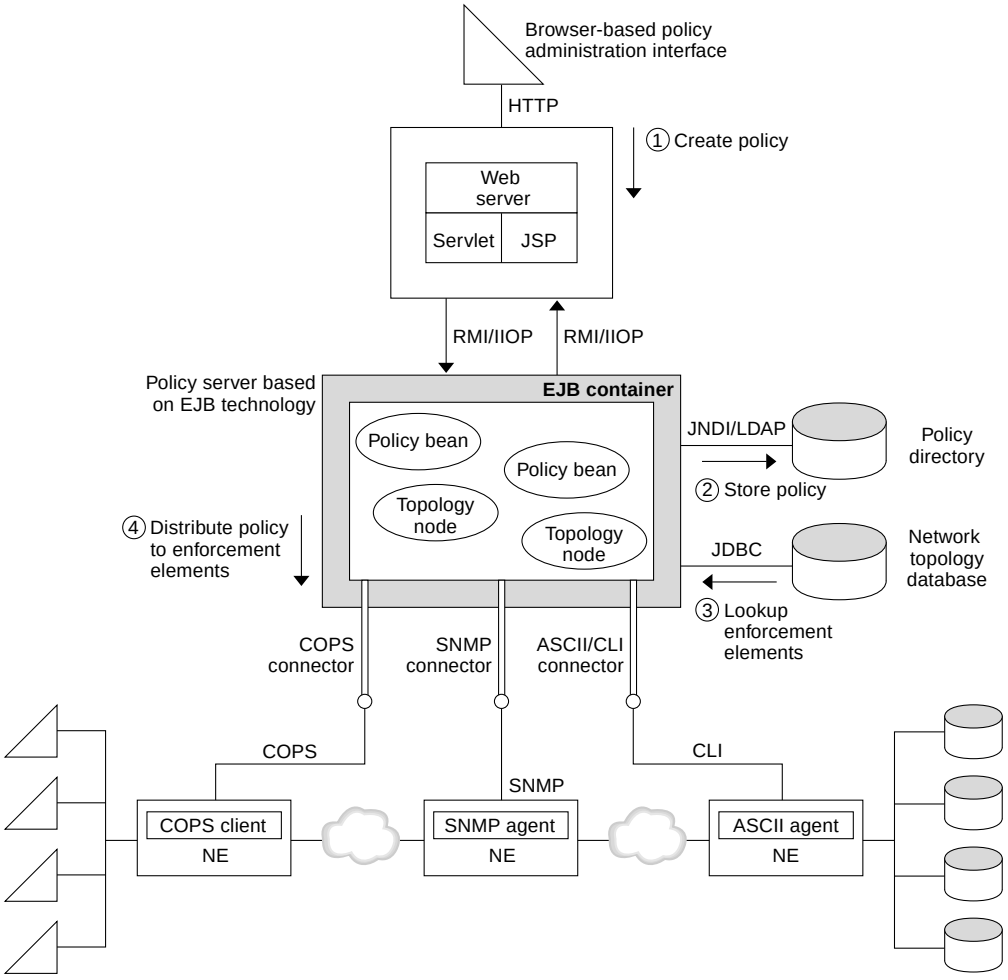


FIGURE 9 Policy Administration Based on EJB Components

Quality of Service (QoS) Enforcement

A QoS enforcement system provides a quality of service for subscriber requests according to the priority assigned to each request. This priority is based on the level of service a subscriber requested from the service provider.

To administer QoS information for users on a network, an administrator uses a browser-based interface of a QoS server. Specific QoS policies are represented as entity beans in an EJB container. They are stored in a QoS directory. The JNDI API and LDAP provide lookup services for this directory.

Topology node information required for enforcing QoS policies is stored in a topology database. A JDBC driver connects the EJB container to the topology database.

An EJB connector enables QoS information represented as entity beans to be distributed to a COPS client application.

In this example, the following levels of QoS are enforced:

- **Gold** for subscribers to the highest level of service
- **Silver** for subscribers to the intermediate level of service
- **Bronze** for subscribers to the basic level of service
- **Deny** for subscribers who are in default with their service provider

The sequence of events for enforcing QoS is as follows:

1. Subscribers send HTTP requests to a router.
2. To find out the QoS level of each HTTP request it is receiving, the COPS client sends a QoS lookup request to the QoS server.
3. The QoS server retrieves topology node information that identifies the subscriber who sent each HTTP request.
4. The QoS server retrieves the priority of each subscriber's requests from the QoS directory.
5. The QoS server assigns QoS labels to each request based on the priorities retrieved from the QoS directory.
6. The router takes action based on the QoS labels assigned for example:
   - The router applies the highest QoS to packets from a gold subscriber.
   - The router applies a lower QoS to packets from a silver subscriber.
   - The router installs a packet filter denying access to packets from a subscriber who is in default with the service provider.

FIGURE 10 shows an example of a QoS enforcement application based on EJB components.

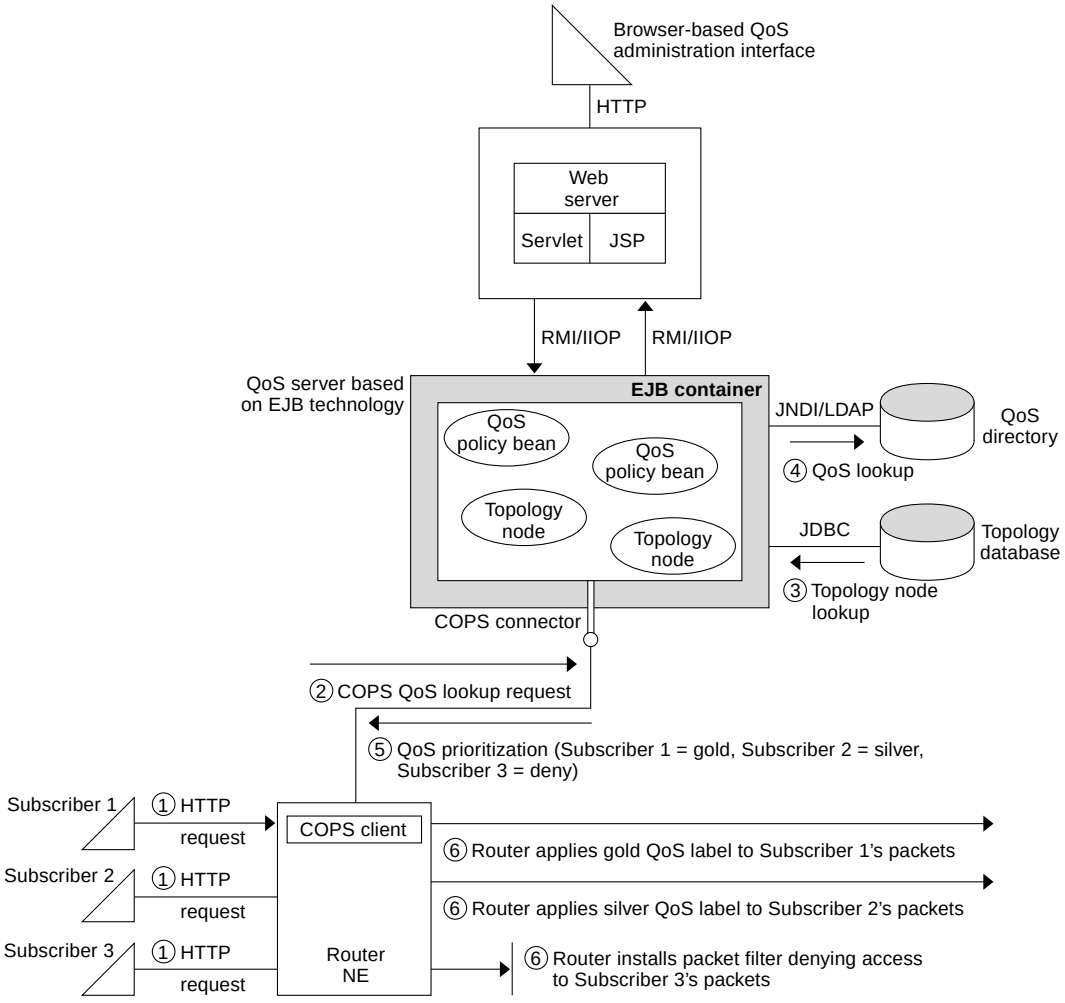


FIGURE 10 Quality of Service Enforcement Based on EJB Components

Conclusion

The development of telecom network management systems presents a unique set of challenges that are difficult to address adequately in a legacy development environment. To meet the demands of today's highly competitive telecom market, a new approach to telecom network management is needed.

An approach based on the server-side application model provided by J2EE technology and the EJB component model shortens development time scales and reduces costs by:

- Enabling suppliers of telecom network management systems to assemble systems from reusable components
- Keeping systems programming to a minimum

This approach also enables easy integration of the many different elements in today's telecom network management systems.

Acknowledgements

Parts of this white paper are based on content supplied by Nortel and Alcatel and used with their permission.

List of Abbreviations

AAA	authentication, authorization, and accounting
ACID	atomic, consistent, isolated, and durable
API	application programming interface
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BSC	base station controller
BTS	base transceiver station
CIM	Common Information Model
CLI	command line interface
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
COPS	Common Open Policy Service
CORBA	Common Object Request Broker Architecture
DPE	distributed processing environment
DTD	document type definition
EAI	enterprise application integration
EIR	equipment identity register
EJB™	Enterprise JavaBeans™
EMS	element management system
ERP	enterprise resource planning

GDMO	Guidelines for the Definition of Managed Objects
GGSN	gateway GPRS support node
GPRS	general packet radio service
HA	high availability
HLR	home location register
HTTP	Hypertext Transfer Protocol
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
ITU-T	International Telecommunications Union - Telecommunications Standardization Section
J2EE™	Java™ 2 Platform, Enterprise Edition
JDBC™	Java Database Connectivity
JMS	Java Message Service
JMX™	Java Management Extensions
JNDI	Java Naming and Directory Interface™
JSP™	JavaServer Pages™
LAN	local area network
LDAP	Lightweight Directory Access Protocol
MIB	management information base
MML	Man-Machine Language
MO	managed object
MOF	Managed Object Format
MSC	mobile switching center
NE	network element
NMS	network management system
OMC	operation and maintenance center

OMG	Object Management Group
ORB	object request broker
OSI	Open Systems Interconnection
OSS	operations support system
PDU	protocol data unit
QoS	quality of service
RAN	radio access network
RDBMS	relational database management system
RMI	Remote Method Invocation
SDH	Synchronous Digital Hierarchy
SGSN	serving GPRS support node
SNMP	Simple Network Management Protocol
SQL	structured query language
TCP/IP	Transmission Control Protocol/Internet Protocol
TL1	Transaction Language 1
TMN	Telecommunications Management Network
TP	transaction processing
VLR	visitor location register
WAN	wide area network
XML	Extensible Markup Language

THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 USA
650 960-1300 Fax 650 969-9131

For U.S. sales office locations, call: 800 821-4643

In other countries, call:

Corporate headquarters:	650 960-1300
Intercontinental sales:	650 688-9000