SUN BLUEPRINTS APPROVED

[illegible]

Enterprise Management Systems Part I: Architectures and Standards

Deepak Kakadia, Sun Microsystems, Inc., Dr. Tony G. Thomas, AdventNet, Inc., Dr. Sridhar Vembu, Adventnet, Inc., Jay Ramasamy, AdventNet, Inc.

Sun BluePrints^{TM} OnLine - April 2002

`http://www.sun.com/blueprints`

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650-960-1300

Part No. 816-4525-10
Revision 1.0, 04/03/02
Edition: April 2002

Please
Recycle

Adobe PostScript™

Enterprise Management Systems, Part I: Architectures and Standards

Enterprise Management Systems (EM Systems) are Network Management Systems (NMSs) capable of managing devices, independent of vendors and protocols, in Internet Protocol (IP)-based enterprise networks. Element Management Systems (EMSs), on the other hand, are NMSs that are designed to manage a particular device, often implemented by the device manufacturer. A summary of typical architectures and a clarification of the myriad of standards are presented to help the reader better understand the implementations of various third-party vendor EM Systems solutions. The architectures of two enterprise management software products, Sun^{TM} Management Center (Sun MC) 3.0 software, an EMS solution, and AdventNet WebNMS 2.3, a standards-compliant development environment are presented.

This article is the first of a two-part article that describes how to manage services in Service Driven Networks (SDNs). This article provides an introduction to EM Systems and provides the reader with a good understanding of the fundamental architectures of both software products. These products solve the problem of managing complex heterogeneous SDN environments. Enterprise Management Systems Part II: Enterprise QoS Provisioning due in the May 2002 issue continues to build on this knowledge, describing how these products are integrated to provide a complete solution that can effectively manage a multivendor environment, and describes how to provision end-to-end services.

This article details the following:

- Introduction to EM Systems
- Overview of architectures
- EM Systems standards
- Sun MC 3.0 software architecture
- AdventNet WebNMS 2.3 architecture

Introduction

This article defines EM Systems as NMSs capable of managing devices, independent of vendors and protocols, in IP-based enterprise networks. This method is in contrast to NMS for Carrier Networks, which have different architectures and are not covered in this article. EM Systems are software solutions that allow systems administrators to manage a vast set of heterogeneous devices in their data centers.

In the last two decades, we have witnessed three major phases of evolution in enterprise networking technology. The first phase moved from a centralized mainframe and dumb terminal architecture to a distributed architecture. A distributed architecture was composed of islands of departmental local area networks (LANs). The second phase involved linking all these disparate departmental LANs and creating an enterprise-wide network. This configuration had two major implications. First, the complexity of managing the enterprise-wide network increased profoundly. Second, a heterogeneous environment emerged from this integrated architecture. The third phase arose as a result of Web-based enterprise services. This changed the enterprise traffic patterns and increased the dependency on mission-critical enterprise services and infrastructures.

EM Systems have tried to keep pace with these changes. In the early days, proprietary systems were created mainly by vendors of computer equipment; however, these systems could not interoperate with other vendor's systems or devices. This conflict created the need for standards. In 1988, Simple Network Management Protocol (SNMP) was created to address some of the interoperability issues. However, as the requirements for more intelligent management systems arose; the limitations of SNMP were soon discovered. Two recent NMS initiatives, Java^{TM} Management extensions (JMX^{TM}) and Solaris^{TM} Web-based Enterprise Management (WBEM) Services, attempt to address these concerns.

Architectures

This section provides an overview of the network and software architecture of typical EM Systems from two perspectives.

1. Network Architecture—This perspective describes the physical network topology of the managers and agents.

2. Software Architecture—This perspective describes the software construction of the manager and agent components.

Network Architectures

The network architecture describes how the EM Systems is deployed. There are several models that can be employed to organize how the managers are organized. Models can be a single central manager, hierarchical managers, distributed peer managers, etc. Network architecture also includes the management protocols that are used to communicate information about the management resource between the managers and agents.

EM Systems can be organized in a variety of architectures, and can communicate management information using one of two standard network management protocols: SNMP for IP-based networks and Common Information Model Protocol (CMIP) for Open Systems Interconnection (OSI)-based networks. Due the industry-wide acceptance of SNMP, CMIP is not discussed in this article. This section provides an overview of the possible network architectures, while SNMP is discussed in the "Standards" section.

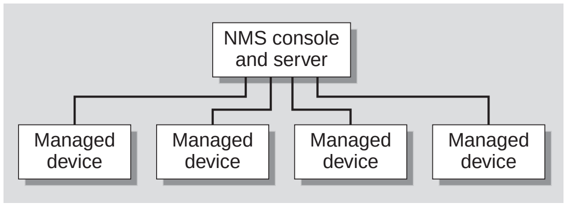
FIGURE 1 describes an overview of the main types of ES System architectures. The choice of architecture has a direct impact on scalability, availability, performance, and security. FIGURE 1 A) describes a centralized architecture, where a single NMS manages all the devices on an enterprise network. The single NMS has limited performance and scalability, in terms of network and computing capabilities. All services, S1, S2, S3, are executed on the central server. These services are management applications that perform one or more Fault, Configuration, Accounting, Performance, and Security (FCAPS) functions, which are described later.

The single network connection becomes congested as the number of managed devices increases. The management server also reaches its limits in terms of polling for events and processing traps. This is the case of the early model EM Systems, such as the initial release of SunNet Manager^{TM} platform. The single management server model is a single-point-of-failure (SPOF).

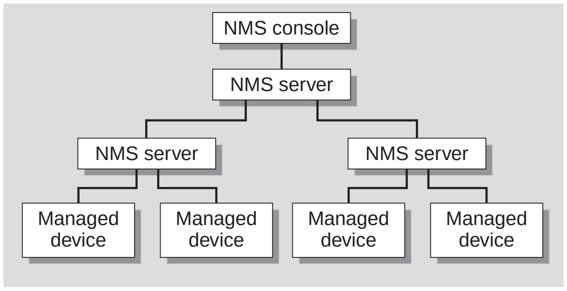
FIGURE 1 B) describes a hierarchical architecture, where there are many local management servers managing small local networks and propagating important events to a higher central management system. This architecture is also referred to as the "Manager of Managers". This model offers better network and server processing performance capabilities. The bulk of the network traffic is localized, because only filtered and correlated events and information are forwarded to the central server. Availability is increased as a local management server failure does not impact the entire system. If the central server fails, the local server can still be accessed for local management information.

FIGURE 1 C) describes a highly distributed system, where any management server can communicate with any managed device. This architecture offers a highly available solution. If one management system fails, there is a backup system to assume responsibility of that domain. This approach also permits specialization of services. Only one expert in each service can be required to manage the entire

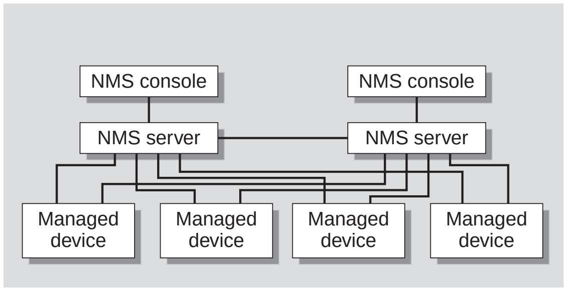
network. This approach is scalable; more management servers can be added as required to alleviate overloaded systems. In most cases, the management systems are geographically remotely located, resulting in increased network traffic in the wide area network (WAN) links, creating a source of performance penalty.



A) Centralized network architecture,
all services reside on central server



B) Hierarchical network architecture,
(manager of managers)



C) Distributed network architecture,
(network of managers), specialization of services

FIGURE 1 NMS Topologies

Software Architectures

The software architecture describes the EM Systems internal construction. This architecture includes the information model that is the software representation of the managed resources and the functional capabilities of the network management system, such as FCAPS functions.

EM Systems software architectures can be classified into the following categories:

1. Element Management Systems (EMSs)—This class of systems are developed by computer and network switch manufacturers, and are specialized to manage only a particular device.
2. Management Platforms—This class of systems are actually development frameworks for NMSs. There are two development frameworks available, one for the agent side and the other for the management side.
3. Management Applications—This class of systems can implement one or more FCAPS functions and may implement these functions in both categories, depending on the scope of the managed resources.
4. Management Systems—This class of systems provides core services, which are accessed via APIs, to the management applications.

Element Management Systems

This class of systems exploit vendor-specific management information base (MIB) variables. A MIB is a set of data objects that are logically grouped, describing the attributes that form the management interface of a device, either hardware or software. There is one standard SNMP MIB, and all vendors extend the standard MIB to add device-specific management objects. MIB definitions have standard syntax and encoding—for example, Abstract Syntax Notation (ASN.1) and Basic Encoding Rules (BER)—which, essentially, allow interoperability.

Management Platforms

In the Management Platforms class of systems, agent development toolkits, such as Windriver, facilitate device manufacturers to build SNMP agents that run on Real-Time Operating Systems, such as VxWorks. This article focuses on the management side of frameworks that facilitate communicating with agents and the development of applications that process this agent information. These platforms provide core functions (see FIGURE 2) such as Event Services, Topology Services, etc. Applications access these core services through APIs. The communication protocols communicate management information between agents and managers. Management Platforms also are able to integrate various vendor-specific EMSs to create a complete enterprise-wide solution. Interoperability is a major issue among vendors and is the main factor driving the call for standards. Another key integration issue is how the data that represents managed resources is represented and decoded so all platforms can understand the information. The information model describes the logical representation of managed resources and is another important consideration for integration. Adherence to a standards-based information model allows for lossless vendor interoperability. If vendors do not adhere to standards, then only common data can be integrated among vendor systems.

There are some key technologies that have eased the implementation of integrated solutions.

- Extensible Markup Language (XML) has proved to be a helpful technology, drastically simplifying customization and integration (detailed in Part II of this article, May issue). Previously, significant coding efforts were required to integrate, and customize solutions.
- J2EE™ technology has proved to simplify the development of sophisticated management applications, where previously, tightly coupled APIs required significant coding efforts.

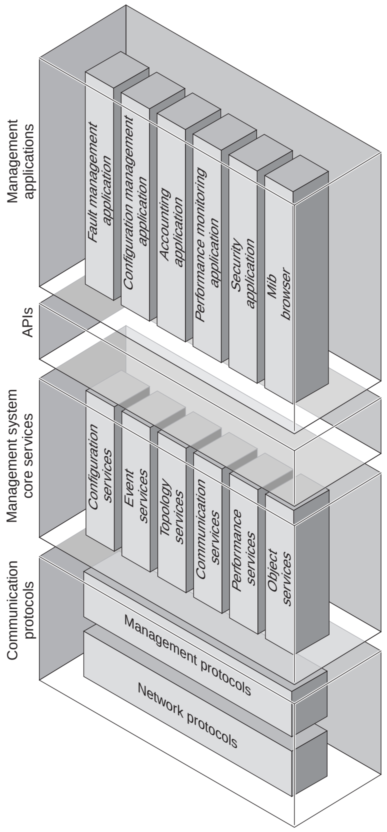


FIGURE 2 Management System Software Architecture

Management Applications—Fault, Configuration, Accounting, Performance, and Security

The management applications are the set of high-level Graphical User Interface (GUI) based applications that are used by operators to manage the enterprise network. A managed application implemented on an EMS can only manage resources on that local device; whereas a management application implemented on a management platform can access local and remote, and can call upon the services implemented at the EMS. Most EM Systems use the following common functions:

- Fault Management
- Configuration Management
- Accounting Management
- Performance Management
- Security Management

Fault Management applications include processing all events and determining if a fault is detected. Fault detection requires other functions including filter events, logging to maintain historical records that detect long-term trends, monitoring, notification, and reporting by generating alarms.

Configuration Management allows the operator to verify and modify the configuration of managed devices. To configure one service that spans only one device is a matter of setting some vendor variables or performing a set of simple tasks. However, service-based networks that provide more sophisticated services, such as Quality of Service (QoS) or virtual private network (VPN), may span several devices, plus need frequent modifications, pose new challenges. This challenge is discussed in detail in the Sun BluePrints Online article, "Enterprise Management Systems for Service Driven Networks: Part II: QoS Provisioning an Integrated Approach", available in the May 2002 issue.

Accounting Management is more important for telecommunication networks rather than enterprise networks. The Accounting function maintains usage-based statistics for billing purposes.

Performance Management provides utilities to the operator to define and periodically measure performance-related variables. These measurements are then used to compare against service level agreements (SLAs). Resources are monitored for bottlenecks and for user-defined thresholds that exceed the limits. These measurements can be saved, and the historical performance data collected used for capacity planning.

Security Management is a massive topic in itself, however, these major features allow network services to be accessed in a secure manner in a distributed network:

- Authentication—Verifies the person attempting to access a resource.
- Authorization—Verifies that the user is permitted to perform certain operations offered by a resource.

- Data integrity—Verifies the integrity of the cryptographic data checksum that confirms the integrity of unaltered data.
- Auditing—Historical tracking of logs used in postmortem investigations as a result of security incidents or proactive precautionary measures.

Although, not strictly belonging to FCAPS, there are essential utilities that most practical NMSs include, such as MIB browsers. The typical MIB browser allows the operator to view the MIB tree, using the point-and-click GUI of a particular device. The display shows the MIB variable, the values, and the structure of the MIB for a particular device.

Management System

The Management System provides core services which are accessed through APIs to the management applications. This last section describes the basic high-level applications that provide the functionality of most EMSs, and that pull data from various functions from within the NMS. The NMS has a set of core services that continuously retrieve and process raw data. The following is a summary list of the basic service modules provided by most platforms.

- Configuration Services module
- Event Services module
- Topology Services module
- Communication Services module
- Object Services module

The **Configuration Services module** translates generic high-level configuration commands to device-specific mappings, generating the appropriate commands to configure the actual device. To accomplish the task of configuring a device, the configuration module may log in and use the command line interface (CLI) of the device, or possibly configure the device using vendor-specific MIB and SNMPs.

The **Event Services module** receives all traps from the managed devices and events created by other modules. Internally created events may be generated for important notifications, such as an exceeded polled threshold.

The **Topology Services module** maintains relations between managed resources. This module is not only used for graphically displaying topologies in the GUI, but it is also used by applications, such as event correlation, to determine root causes by analyzing relationships and dependencies among devices. The topology service also performs the initial population of the managed resource database where EMSs start by performing an auto discovery (or read text configuration data from a file). As devices are discovered and probed, raw data is used to instantiate and populate the managed resource objects, then is saved in persistent storage.

The **Communication Services module** contains the protocols and encoding to send and receive management information between the management system and the agents.

The **Performance Services module** is the set of processes that continuously probe managed devices for performance-related data and forward it to the performance management applications

The **Object Services module** provides persistent storage for all the managed resources and related data. Larger implementations use relational databases, such as Oracle. In contrast EMSs are object-oriented databases. Upon the initial startup, as objects are instantiated, the attributes are populated from the relational database. Other EMSs use object-oriented databases, such as Versant, and can have serious integrity issues on large deployments.

Standards

Standards are specifications for implementing a particular network management function. By adhering to standards, different vendor implementations are guaranteed to interoperate. The following sections detail SNMP framework (also known as the Internet standard management framework).

Simple Network Management Protocol

The SNMP framework was developed for IP-based networks in the late 1980s (for additional details see the "Related Information" section), and is the dominant framework in the computing industry for the management of these devices. The SNMP management framework has undergone three major revisions, with the latest being SNMP V3.

The initial release of SNMP had several weaknesses:

- Access control such as, Community Strings, were transmitted in clear text
- Transfer of table data required multiple small operations
- Limited trap types
- The "Urgent trap message" was the only command the agent could initiate

These limitations were addressed in SNMP V2, which is currently the dominant protocol in the industry and the version detailed in this article.

FIGURE 3 illustrates an overview of the SNMP V2 framework. The "manager" communicates with various managed devices, which are running an SNMP agent. The SNMP framework consists of four major components:

- Managed devices hosting an embedded SNMP agent.
- Management server communicating with the managed devices and providing services to management applications.

- Management protocol utilizing the SNMP protocol for exchanging messages between the manager and agents by encoding management information in BER.
- Management Information model defining all managed resources in a pseudo object-oriented manner—a manner where all objects are stored virtually in a MIB.

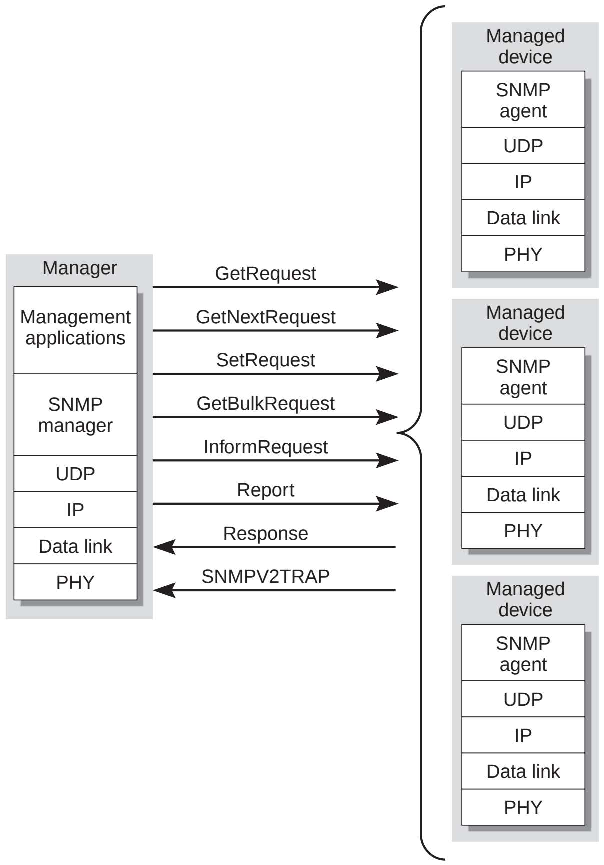


FIGURE 3 SNMP V2 Framework

SNMP Operations

FIGURE 3 also illustrates the following SNMP V2 operations.

- **Get-Request** (or read) obtains information from the agent about an attribute of a managed object.
- **GetNext** does the same thing as the Get-Request operation for the next object in the tree of objects on the managed device.
- **Set** (or write) sets the value of an attribute of a managed object.
- **TRAP** tells the manager about some event on the managed device.

- **Get-Bulk-Request** gets large amounts of data in one single operation, for example all the data in a large table. A new feature in SNMP V2.
- **Inform-Request** allows a manager to communicate management information to another manager.
- **Response** returns by the agent in response to a Get-Request, GetNextRequest, GetBulkRequest, SetRequest, or InformRequest protocol data unit (PDU).

Management Information Base

The MIB contains a description of the object hierarchy on the managed device, the name (called Object ID), syntax, and access privileges for each variable in the MIB. For example, when the MIB module is loaded in a MIB browser, the label of the variable, for example `sysDescr`, can be used to identify it, because the MIB browser uses MIB module to translate this label to an Object ID.

Specifying SNMP Variables in GetRequests

To specify an object to an SNMP agent, both the Object Name or ID (which defines the type of object) and the instance (the specific object of the given type) need to be provided. From the MIB, you get the Object ID, to which an instance needs to be added. For non tabular (or scalar) objects, this is simply an instance of 0 (for example, `sysDescr.0`). For tabular objects, the instance is defined by the MIB and is a sequence of one or more variables (for example, `interfaces.ifTable.ifEntry.ifIndex.1 = 1`)

In order to get and set SNMP variables, you need to specify the Object ID plus the instance. You can also use GetNext, and specify just the Object ID from the MIB (for example, `sysDescr`), and get the first instance of that type from the SNMP agent. This method works for all types of objects.

When using the MIB browser, select the MIB node that you need, and either select GetNext or add the instance you need at the end of the Object ID and use the Get command.

Web-based Enterprise Management/Common Information Model

The standard Web-based Enterprise Management (WBEM) was originally a consortium of several manufacturers that wanted to develop an open platform and protocol-neutral Web-based network management architecture. The original goals were to create a set of standards that allowed Web browser-based applications to share management information from any vendor. This application originally required Web-based technologies and an HTTP-based management protocol. Over the years, the consortium developed a highly successful standard information model, called the Common Information Model (CIM). The CIM allows any vendor to extend their vendor-specific resources from a common set of superclasses. The CIM is composed of the following components:

- MetaSchema is the object model.
- Managed Object Format (MOF) is the standard textual syntax for specifying the description of managed resources.
- Extensible Markup Language (XML) is the standard language that replaced the original Hypermedia Management Protocol (HMMP), which was developed by Distributed Management Task Force (DMTF).

MetaSchema is partitioned into the following object (class) hierarchies:

- Core Model—A set of classes that are common to all management systems. This includes objects such as Managed System Element, Physical Element, Logical Element, System, and Service.
- Common Model—A set of classes that subclass from the core model and are related to one vertical segment, such as Physical, Systems, Devices, Applications, and Networks.
- Extensions Model—A set of vendor-specific classes that extend from the common model classes.

The DMTF developed a MetaSchema mapping from the CIM MetaSchema to an XML MetaSchema, which is now used in the management protocol.

FIGURE 4 presents an overview of the WBEM/CIM architecture.

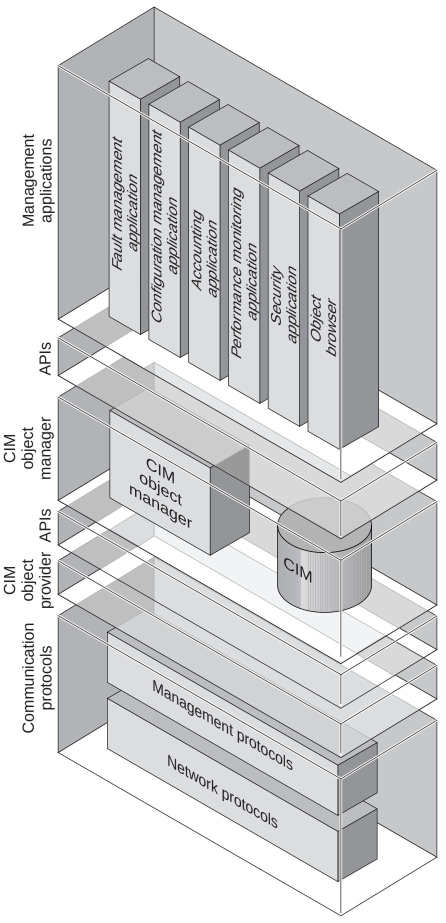


FIGURE 4 WBEM/CIM Architecture

The management applications access the services of the CIM object manager, which accepts requests for creating namespaces, manipulating objects, retrieving/storing management information into the CIM persistent store. If the CIM object manager is unable to service a request, that request is forwarded to the CIM object provider. The CIM object provider translates from the CIM format to a device-specific format, such as structure of management information (SMI)/SNMP, and performs that operation.

The Solaris^{TM} WBEM services and Sun^{TM} WBEM software development kit (SDK) provide these APIs, a query language, and other features that assist developers in creating complete WBEM solutions. Please see the “Related Information” section for additional information on WBEM.

Java Management Extensions

Java Management extensions (JMX) was created to address some of the limitations of SNMP. For instance, SNMP only permits primitive operations, so if you need an additional agent, side intelligence is required. For example, figuring out whether a spurious trap should be sent or not. The limitations of SNMPs can be overcome by using a JMX technology-based agent. The Sun Microsystems product, Java™ Dynamic Management Kit, facilitates the creation of intelligent agents using JavaBeans™ architecture. A MIB compiler included in the kit, reads in a MIB, creates the beans and stubs, and then allows you to fill in the logic.

JMX provides developers of Java technology-based applications, across all industries with the means to instrument Java platform code, create smart agents and managers in the Java programming language, implement distributed management middle-ware, and smoothly integrate these solutions into existing management systems. For additional information on JMX technology, please see the "Related Information" section. FIGURE 5 illustrates how the various components of JMX technology fit together. The JMX specification architecture is divided into three levels, or layers, which are used interchangeably.

- Instrumentation layer—Provides manageability to any Java technology-based object. This level is aimed at the entire developer community utilizing Java technology. This level provides management of Java technology, which is standard across all industries.
- Agent layer—Provides management agents. JMX technology agents are containers that provide core management services, that can be dynamically extended by adding JMX technology resources. This level is aimed at the management solutions development community and provides management through Java technology.
- Distributed Services layer—Provides management components that can operate as a manager or agent for distribution and consolidation of management services. This level is aimed at the management solutions development community and completes the management through Java technology provided by the agent layer.

In addition, JMX technology provides a number of Java technology APIs for existing standard management protocols. These APIs are independent of the three-level model, yet they are essential because they enable JMX technology applications, in the Java programming language, to link with existing management technologies.

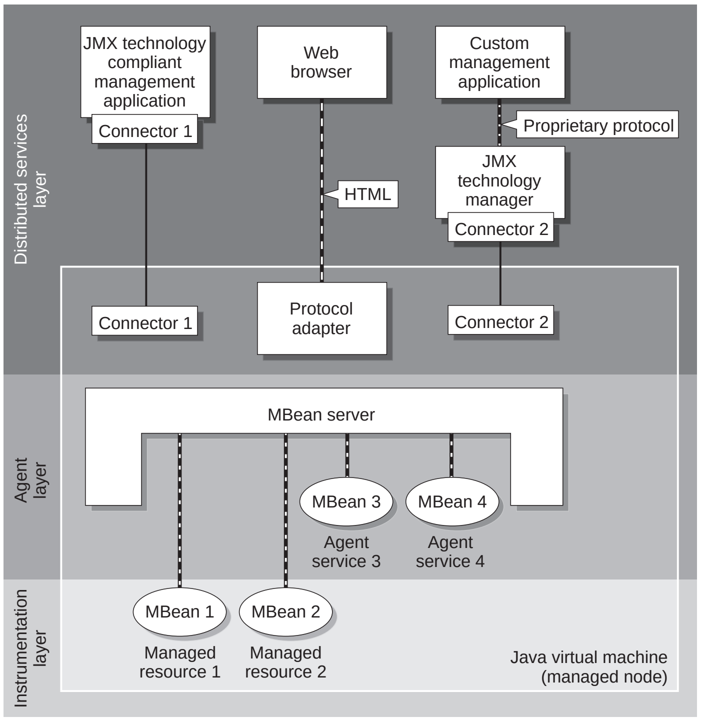


FIGURE 5 JMX Specification Architecture Components

A JMX technology agent is a management entity implemented in accordance with the JMX agent specification, and tested against the Agent Level Compatibility Test Suite. A JMX agent specification is composed of objects from the Java Dynamic Management Kit called Management Beans (MBeans). A JMX agent specification consists of an MBeans server, a set of MBeans representing managed resources, and at least one protocol adaptor or connector. A JMX agent specification can also contain management services, also represented as MBeans. The MBeans server is a registry for MBeans in the agent. The MBeans server is the component that provides the services allowing the manipulation of MBeans. All management operations performed on the MBeans are done through Java technology-based interfaces on the MBeans server.

There are two concepts used in accessing JMX technology agents: protocol adapters and connectors. Protocol adaptors give a representation of the MBeans directly in another protocol, such as hypertext markup language (HTML) or SNMP. Connectors include a remote component that provides end-to-end communications with the agent over a variety of protocols (for example HTTP, HTTP, and internet inter-orb protocol (IIOP)). Protocol adaptors and connectors let management applications access a JMX agent specification and manipulate the MBeans it contains. Because all connectors have the same Java technology-based interface, management applications use the connector most suited to their networking environment, and even change connectors transparently as needs evolve.

JMX technology and MBeans provide:

- Standard manageability for any Java application, sometimes in just three to five additional lines of code.
- The ability to embed all necessary management information in a standard way in the resource to be managed.
- The ability to provide a wrapper for instrumented resources not based on Java technology (even proprietary or custom solutions) with Java technology-based management systems.

A JMX technology compliant agent is automatically capable of managing JMX technology resources. A non-JMX technology agent may also support JMX technology resources. Finally, the JMX technology instrumentation layer does not introduce any dependencies on external classes, a resource is entirely self contained.

The JMX specification agent and manager levels provide a flexible, distributed, and dynamic management infrastructure to be implemented in the Java programming language. By extending the Java programming language, JMX technology enhances the capabilities of existing solutions, and enables the rapid creation and deployment of new types of management solutions. These solutions can be extended dynamically to incorporate new equipment and services in a plug-and-play manner.

The JMX specification agents and managers developed using Java technology offer the following benefits:

- Platform independence
- Protocol independence
- Information model independence
- Management application independence

Products

The architectures of two enterprise management software products, Sun MC 3.0 software—an EMS solution and a development environment for extending the capabilities—and AdventNet WebNMS 2.3—a standards compliant development environment that has gained acceptance in the Telco environments. The following sections detail the features that produce the benefits of these two products.

Sun MC 3.0 Software Architecture

Sun MC 3.0 software evolved from several generations of Sun Network and Systems Management Tools. Starting with SunNet Manager platform, in the late 1980's, the architecture consisted of a simple manager-agent-manager model. SunNet Manager platform was extensible and included a remote procedure call (RPC)-based communication model between a manager and agents. SunNet Manager platform served its purpose well for LAN deployments, but reached limitations in larger enterprise solutions. Sun's subsequent generation product, the Solstice Enterprise Manager^{TM} software, was designed for large enterprise deployments. Solstice Enterprise Manager software was scalable, flexible, object-oriented, and was targeted as a development and management platform out of the box. Solstice Enterprise Manager was highly extensible, and supported SNMP, CMIP, and Telecommunications Management Network (TMN) protocol. The rapid adoption of Java technology-based management applications created another next-generation product, Sun Management Center software, (version 3.0 is the current release as of the date of this article). Sun MC 3.0 software is a comprehensive systems management tool for managing Sun servers, storage, and the Solaris^{TM} Operating Environment. Sun MC software is also a flexible framework with independent distributed services that are available to management applications and third-party management products. The architecture includes a modular pluggable architecture that can add and remove Java technology and non Java technology services.

Sun MC software is designed primarily to manage Sun Systems well. AdventNet WebNMS is designed to manage a variety of devices by integrating other vendor-specific EMS products. The integration of Sun MC and AdventNet WebNMS softwares, permits a flexible and effective solution, because every vendor knows how to manage their own devices best.

FIGURE 6 shows a high-level overview of the Sun MC 3.0 software architecture. Sun MC software consists of following tiers:

- Client Tier—Management clients access the server through a variety of protocols and technologies, such as a Web browser and Java application.
- Server Tier—Interfaces with the client accepting and processing requests, and routes to the appropriate service or agents. The server also propagates alarms and event messages to the clients. The server is a central control center for all distributed information. Sun MC 3.0 software offers a comprehensive set of APIs for integration with other management systems, and empowers developers in creating sophisticated management applications.
- Agent Tier—Resides on the managed device and provides the communication interface between the native device and the Sun MC software server process.

Sun MC 3.0 software also has many of the core NMS functions, such as Event Services, Topology Services, Configuration Services, etc. The Sun MC software agents are intelligent agents, far more sophisticated than the traditional simple SNMP agents that only perform limited operations. The Sun MC software agents are dynamically configurable and autonomous.

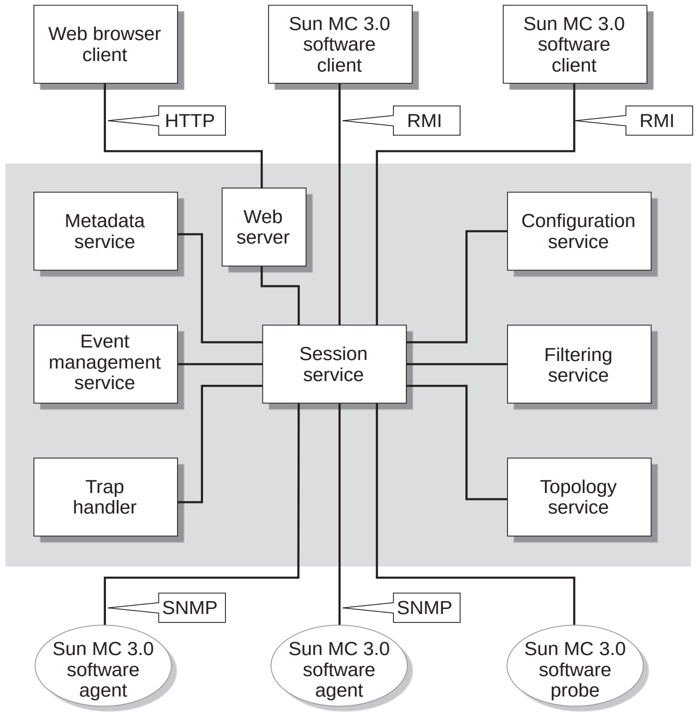


FIGURE 6 Sun MC 3.0 Software High-Level Architecture Overview

Sun^{TM} Developer Management Center is a separate product, which provides various APIs and tools to create not only management applications using the exposed client APIs, but also to extend the capabilities of the Sun MC software by creating loadable modules that run in the server tier. These modules can access the services shown in FIGURE 6 and create sophisticated event filter and correlation services. The server APIs allow the integration and creation of new services, such as emitters of notifications, receptors of events, protocol modules, etc.

AdventNet WebNMS 2.3 Architecture

FIGURE 7 shows a high-level overview of the AdventNet WebNMS architecture.

The clients can be Java application or browser-based. They interface with the front-end Web servers and are typically load balanced by a multilayer switch. The core functions include a discovery engine that maps discovered devices to a managed object and is defined by a software class. This information is then stored in the database. The event manager includes a hierarchical structure starting from a trap manager that identifies the trap and creates an event object. This object is then parsed to create alarm objects. At each level, redundant data is discarded and events are correlated. The map server controls the display, including the displayed objects and appropriate backgrounds, when zooming in and out. The configuration server stores the configuration of all discovered devices, and is capable of reconfiguring a device as needed. The polling engine performs periodic polling for status and performance data.

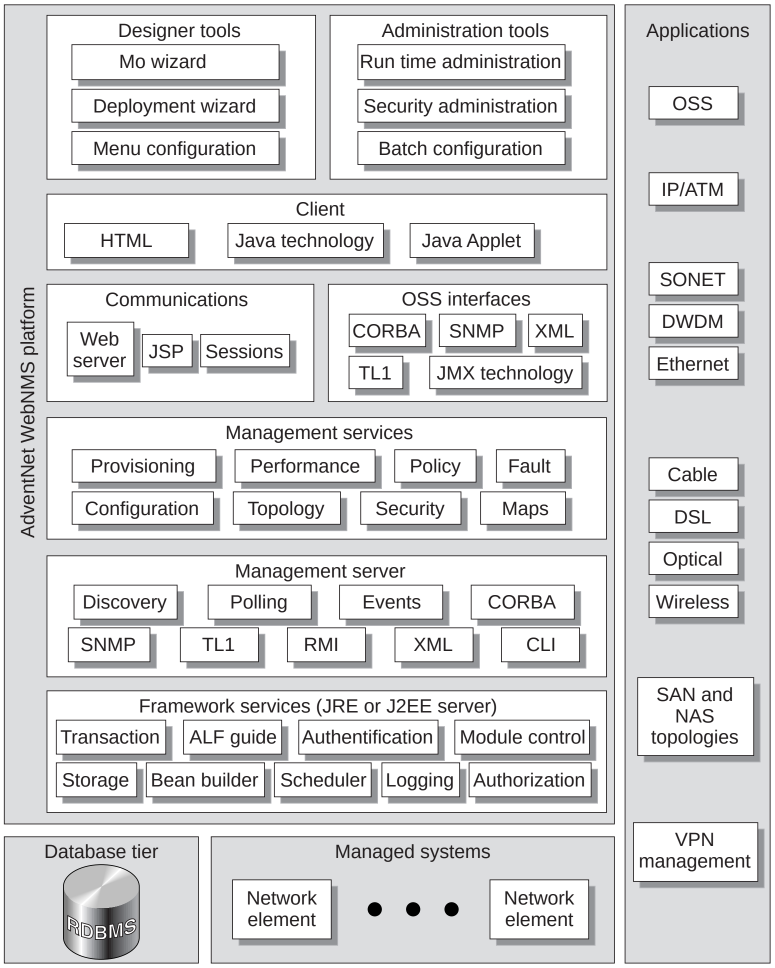


FIGURE 7 AdventNet WebNMS Platform Overview

FIGURE 8 presents an overview of the components in AdvenNet WebNMS and the interactions between them.

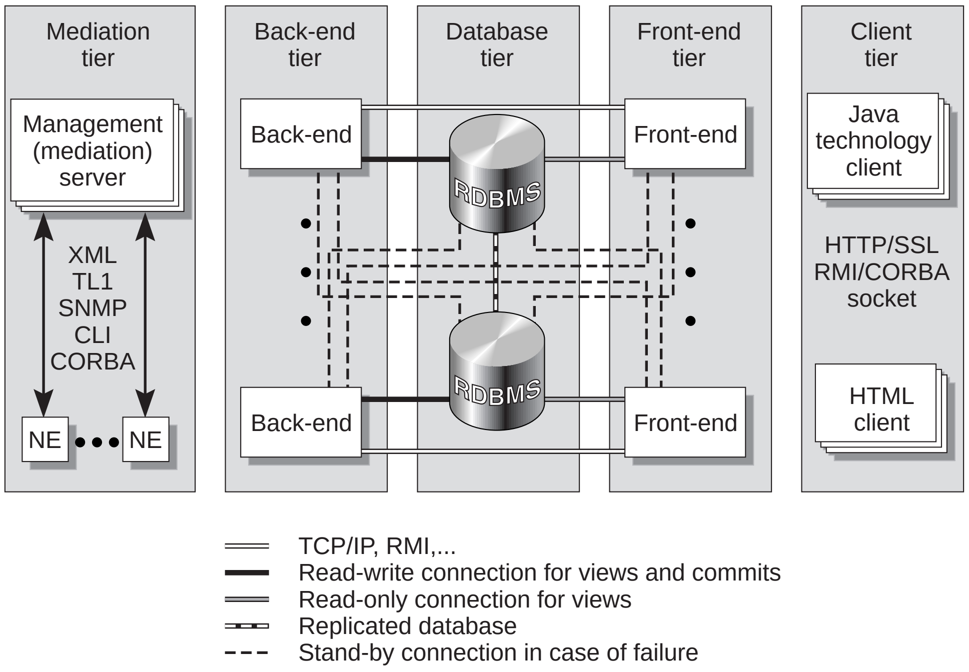


FIGURE 8 AdventNet WebNMS Platform Architecture

Each component can be distributed. There can be one or more components of each type based on performance, scalability, and availability requirements. There are five tiers that are detailed in the following sections.

Mediation Server Tier

The mediation server (or management) tier provides XML mediation for all southbound management protocols like SNMP, TL1, CORBA, TFTP, XML, CLI/ Telnet, etc. A provider interface facilitates integration of other protocols. Common management functions are handled in this layer, making the protocol providers limited with protocol-specific operations only. The management server provides multiple interfaces for application development like XML message interface, Java technology, APIs, etc.

Back-end Tier

The back-end server tier consists of the core business logic related to management functions like fault, configuration, performance, security, service provisioning, etc. The main aspects of the back-end server tier are:

- Core Management —Performs functions like event correlation, alarm notification and management, template-based provisioning, batch configuration and rollback, etc.
- Module Management—A module control interface facilitates the starting and stopping of modules independently.
- Security and Audit—All the back-end tier modules support authorization and audit, so that the various administrative operations can be traced.

Database Tier

Any relational database management system (RDBMS) that provides a Java Database Connectivity^{TM} (JDBC^{TM}) software driver is supported. State information can be maintained in a database, making it easier to distribute the components and handle failover. This tier allows a solution to leverage the benefits of transaction support, database synchronization, and object locking for data integrity, security, and availability.

Front-end Tier

The front-end tier consists of the Web container that provides Web access to management information, the client communication management module, and the Enterprise JavaBeans^{TM} (EJB^{TM}) technology session beans for the management functions. These items generate views for the clients and forward commit requests to the back-end tier. The main aspects of the front-end server tier are:

- Client communication—Various transport protocols, such as TCP, Java^{TM} Remote Method Invocation (Java RMI) technology, HTTP, HTTPs, SSL, etc., are used.
- Session beans—The stateless, EJB technology deployable session beans generate views from the database based on client requests.
- Updates to client—A subscription-based notification used to allow the front-end tier to handle high rates of updates to the clients. In this model, clients register for the updates they are interested in, and the front-end tier server notifies them of any changes.

Client Tier

The J2EE configuration client (for rich GUI) and the HTML client (for Web access over low speed links) models are provided.

Related Information

Please see the following related information for additional details on the technology presented in this article.

Case, J., et al. *Introduction to Version 3 of the Internet-standard Network Management Framework*, RFC 2570, April 1999.

Harrington, D., et al. *An Architecture for Describing SNMP Management Frameworks,* RFC 2571, April 1999.

Rose, M., McCloghrie, K. *Structure and Identification of Management Information for TCP/IP Based Internets*, RFC 1155, May 1990.

Rose, M., McCloghrie, K. *Concise MIB Definitions*, RFC 1212, March 1991.

Rose, M. *A Convention for Defining Traps for Use With the SNMP*, RFC 1215, March 1991.

McCloghrie, K., et al. *Structure of Management Information Version 2 (SMIv2)*, RFC 2578, April 1999.

McCloghrie, K., et al. *Textual Conventions for SMIv2*, RFC 2579, April 1999.

Case, J., et al. *A Simple Network Management Protocol (SNMP)*, RFC 1157, May 1990.

Case, J., et al. *Introduction to Community Based SNMPv2*, RFC 1901, January 1996.

Solaris Web-Based Enterprise Management (WBEM) Services Developer's Guide, Sun Microsystems, Inc., Part Number 806-6828-05, August 2001.

Java Management Extensions Instrumentation and Agent Specification, v1.0, Sun Microsystems, Inc., May 2000; [`http://www.jcp.org/aboutJava/communityprocess/first/jsr003/index.html`].

Sun Management Center 3.0 Developers' Reference Manual, Sun Microsystems, Inc., Part Number 806-5945-10, November 2000.

About the Authors

Deepak Kakadia is a staff engineer, network architect in the Enterprise Engineering Group, for Sun Microsystems, Inc., located in Menlo Park, California. Deepak has been with Sun for seven years. He previously worked for various companies including Corona Networks as a principal engineer; Network Management Systems, as a team leader for the QoS Policy-based NMS subsystem; Digital Equipment Corp, where he worked on DEC OSF/1; and with Nortel Networks (Bell Northern Research), in Ottawa Canada, as member of the technical staff. Deepak received his B. Eng. in Computer Systems, M.S. in Computer Science, and completed his Ph.D qualifying exams and course work. Deepak has filed two patents: 1) Event Correlation and 2) QoS in the area of Network Management.

Dr. Tony G. Thomas is the COB and chief architect at AdventNet, Inc. Prior to founding AdventNet, Inc. he was a member of the technical staff at AT&T Bell Laboratories in Holmdel, New Jersey. He was instrumental in developing advanced Frame Relay network management solutions at Bell Labs. He obtained his Ph.D. in Electrical Engineering from Johns Hopkins University, Baltimore, Maryland.

Dr. Sridhar Vembu is the CEO of AdventNet, Inc. He started his career at Qualcomm, Inc., in San Diego, where he was a wireless systems designer working on satellite-based personal communications systems. He has applied for three patents in this area. He obtained a Ph.D. in Electrical Engineering from Princeton University, Princeton, New Jersey.

Jay Ramasamy is a Senior Software Engineer with AdventNet, Inc. and has several years of Enterprise JavaBeans software development experience. He obtained his M.S. from Case Western Reserve University, Cleveland, Ohio.