

ABSTRACT

The increasing need for standardized and simplified systems and network management tools together with the hype for web-based applications have increased the demand for web-based network management systems. Today there are two competing technologies: Web Based Enterprise Management, WBEM, and Java Management API, JMAPI. Microsoft develops WBEM and Sun develops JMAPI. They have different approaches to data modeling, presentation, and technical solutions and this master thesis investigates the differences between the approaches and explains corresponding technologies, such as protocols, data modeling, and environments.

CONTENTS

1 PROBLEM DESCRIPTION 4

2 BASIC TERMINOLOGY 5

3 BACKGROUND TO SYSTEM AND NETWORK MANAGEMENT 6

3.1 CHARACTERISTICS OF FUTURE NETWORKS 6

3.2 IMPORTANT TASKS IN MANAGEMENT 7

3.3 THE DESKTOP MANAGEMENT TASK FORCE MANAGEMENT INTERFACE 8

3.4 WEB-BASED MANAGEMENT CONCEPTS 8

3.5 OUTLOOK..... 8

4 SURVEY OF THE DOMAIN..... 10

4.1 SIMPLE NETWORK MANAGEMENT PROTOCOL 11

4.1.1 Messages..... 11

4.1.2 Trap-directed Polling 12

4.1.3 Management Information Base 13

4.1.4 SNMPv2..... 15

4.1.5 SNMPv3..... 16

4.2 REMOTE NETWORK MONITORING..... 16

4.2.1 RMON Goals..... 16

4.2.2 Control of Remote Monitors..... 17

4.3 DESKTOP MANAGEMENT INTERFACE..... 17

4.3.1 Background..... 17

4.3.2 Structure 18

4.3.3 Existing DMI applications 19

4.4 WEB BASED ENTERPRISE MANAGEMENT..... 20

4.4.1 Introduction 20

4.4.2 WBEM architecture 21

4.5 JAVA MANAGEMENT APPLICATION PROGRAMMING INTERFACE..... 23

4.5.1 JMAPI high level model..... 23

4.5.2 JMAPI features..... 24

4.5.3 JMAPI Elements..... 24

4.6 REMOTE METHODS 28

4.6.1 Basic concepts..... 28

4.6.2 CORBA 29

4.6.3 Distributed Common Object Model..... 30

4.6.4 Remote Method Invocation 30

4.7 COMMON INFORMATION MODEL 31

4.7.1 Core Model 33

4.7.2 Common Model 34

4.7.3 Extension schema 34

5 RESULTS 35

5.1 DATABASE 35

5.2 DOCUMENTATION 35

5.3 EVENTS 35

5.4 DATA MODELING..... 36

5.5 USER INTERFACE 36

5.6 TECHNOLOGY 36

5.7 THE IMPLEMENTATION OF NEMAPRO..... 37

6 CONCLUSIONS..... 38

A THE INTERNET DRAFTS.....	40
B THE OSI MODEL.....	42
C TCP/IP COMPARED TO OSI.....	44
D ORGANIZATIONS.....	45
D.1 OBJECT MANAGEMENT GROUP.....	45
D.2 INTERNET ENGINEERING TASK FORCE	45
D.3 DESKTOP MANAGEMENT TASK FORCE.....	46
D.4 THE OPEN GROUP.....	47
E UML NOTATION	48
F REMOTE PROCEDURE CALL	51
F.1 OVERVIEW	51
F.2 ONC RPC.....	52
F.3 DCE RPC	52
F.4 MS RPC	52
G ABBRIVATIONS	54
H REFERENCES.....	55
I INDEX	57

1 Problem description

There is no standard client application today that manages embedded systems, like AXIS products, and other network peripherals. Every product on the network needs its own vendor specific application to be installed and managed. Today AXIS products support network management with SNMP, Wimp and HTTP. Wimp is an AXIS proprietary protocol and therefore not covered in this thesis. Problem arises when the network grows and the different versions of products are connected to the network.

The direction of management solutions is towards web-based management and a couple of vendors have presented their solutions. A generic platform that the market has accepted does not exist.

We have investigated the market, researched the Internet and adequate literature in systems and network management to find out which standards there are and what will come in the future. The investigation has included both protocols and environments. After the investigation we have concentrated in one solution and tried implementing a prototype to manage AXIS products.

The goal of this master thesis project was to manage an Axis StorePoint CD using the next generation web based management tools. We wanted to show that there is no need to install any software besides the web browser at the client, and that the managed server provides information about itself using automatically downloaded modules to the management tool. The manager does not have to know about individual units since they are all available from within the tool. These goals were hard to reach and in the section 5.7 we will explain why.

The master thesis is a report and an implementation of a prototype to show the capabilities of the environment in which we have concentrated. The report also compares the different methods and explains the corresponding philosophies.

2 Basic terminology

Throughout this document, the term *system* means an arbitrary object attached to a network such as a computer, router¹, hub, or software. The term enterprise means a computer network of size of a larger company and enterprise management is system and network management in an *enterprise* network. The different technologies are described with a three-tier model (Byte [9708]) with client, server and managed devices (see Figure 2-1). The client is the managing application, in the web-based technologies this is a browser, the server is an application server with a database connected, and the application is connected to the networked objects. A networked object has an agent that communicates with the application server.

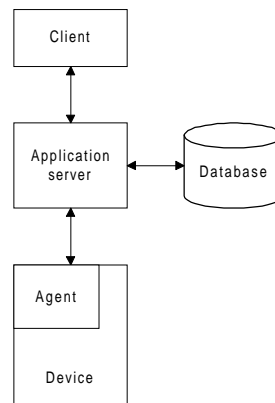


Figure 2-1 Basic terminology

¹ For description of routers, hubs and other devices needed for network communication see (Tanenbaum [96])

3 Background to system and network management

From the very beginning the personal computer (PC) was intended to be a standalone device. Today, PCs have become a central part of data processing and information technology (IT). It has become one of the most innovative markets for technical devices, dominated by a few manufactures. This concentration hinders an integrated management of networks, systems, and applications using open standards.

Client/server and other downsizing strategies are increasingly popular alternatives to mainframe-based, centralized computing. The increasing performance and user-friendly interface of PCs make them a natural part of these strategies. From the viewpoint of integrated management these distributed infrastructures increase the complexity of the environment in three ways. First, the diversity of the involved equipment increases; second, the sheer number of pieces of equipment increases; and third the probability of malfunction due to hardware and software problems increases. Networks can be characterized into three different groups:

- Small, homogeneous networks can easily be administrated manually, since their management is reduced to a few users, printers and file systems as well as a few hardware devices.
- Medium-sized, slightly heterogeneous networks whose administrators are facing the task of taming their chaotic growth in order to allow further expansion.
- Large, heterogeneous networks might consist of UNIX-based client/server solutions, Novell Netware networks, Windows NT/95/3.11, and VMS hosts. Protocols may come from diverse areas such as IPX /SPX, TCP /IP, NetBEUI, and SNA. Because of the large diversity, and resulting complexity, the need for management support is greatest in these networks.

3.1 Characteristics of future networks

In the future, network complexity and flexibility is increasing, resulting in higher demands for management. Some of the problems are discussed here.

Mobile and nomadic computing

Portable computers such as notebooks are becoming more and more common in today's networks. The moving around of these systems between their "home" network and the linking up of foreign system create new management tasks based on user oriented-services (i.e. printing, e-mail, software licensing, and network services).

Communication protocols

TCP/IP has become the protocol of common choice in Local Area Networks (LANs), especially in larger networks. With the next generation IP (IPv6) on the rise, as well as an increasing need for renumbering networks (Heilbronner-Wies [9708]), more automated configuration management is needed.

Security

The integration of former isolated LANs to intranet and connections to Internet increases chances of unauthorized access to internal systems. In larger networks with many different systems centralized user management is necessary. Otherwise users will be confused by multiple logons and multiple passwords and use "shared" accounts with passwords written on stickers beneath the keyboard.

Distributed data storage and processing

One of the main elements in downsizing is to distribute the processing of data. This distribution of business-critical data results in higher probability of failure and increase the demands for a centralized backup strategy.

3.2 Important tasks in Management

The characteristics of today's heterogeneous networks increase the priorities of the tasks explained in this section.

Automated Configuration Management

The usually large number of systems in a network allows for a large benefit if some sort of automated configuration is introduced. The current standard in this area is Dynamic Host Configuration Protocol (DHCP) which allows peripherals to obtain configuration of network parameters from centralized servers. DHCP server management needs to be tightly integrated with the overall management system.

Inventory

The need to discover, record, trace, and log the hardware and software inventory of an organization is rising. Fault management is drastically simplified if the administrator has access to inventory information. Also, financial considerations related to taxation and budgeting as well as adequate management of software license provide reasons for integrated and automated inventory management.

Software distribution

The distribution of operating system software as well as basic application software suffer from the lack of standardization the description of mutual dependencies between software components. This also poses problems with deinstallation of software.

Remote monitoring and control

To support users when problems arise as well as during day-to-day computing tasks, a member of the help desk staff might need to take control over the user interface. If support has been outsourced care should be taken so that the organization's security policy is followed.

License management and application metering

It is important to trace and log software usage, especially for economic reasons. User licenses should be administered centrally but with no negative effects on flexibility. Sophisticated schemes can be applied if the software supplier's marketing strategy suggest this. These user-based schemes can, for example, involve parameters such as the average or maximum usage of the software during certain time periods. For this purpose standardized licensed managing is important. The only standard related to license managing so far is the License Management Service (LMS) within the Open Software Foundation (OSF) standard Distributed Management Environment (DME). Problems to be tackled are how to solve license management for homogenous applications across heterogeneous platforms, i.e., MS-DOS, Apple System 7, NT, UNIX, MVS, and the integration of nomadic and mobile systems, as well as standards based and automated exchange of accounting information between licensee and licensor. An example of homogenous application is Netscape's web browser which has been ported for a number of platforms and operating systems.

3.3 The Desktop Management Task Force management interface

The tasks mentioned above can be solved efficiently by using principles of integrated network and systems management. However, integrated management of computer systems in PC networks has only started to emerge. There already exist some interesting and valuable approaches to management, but due to the complexity of Telecommunications Management Network (TMN) and Open Systems Interconnection (OSI) management and the software market having a few dominant players, these approaches have not been successful or implemented.

Management standardization efforts are focused on Internet SNMP and OSI management architectures, and newer approaches such as Common Object Request Broker (CORBA)-based techniques (see section 4.6.2), Web Based Enterprise Management (WBEM) (see section 4.4), and Java Management Application Programming Interface (JMAPI) (see section 4.5). One interesting role in standardization of PC-related management is played by the Desktop Management Interface (DMI) (see section 4.3) defined by the Desktop Management Task Force (DMTF) (see appendix D.3).

The DMTF was founded with the idea of defining management of small to medium-range heterogeneous end systems such as PCs and UNIX workstations. Until the foundation of the DMTF, only partial and proprietary solutions of management tasks involving this group of end system existed. For defining the management information to be exchanged an information model has been defined, which provides a functional superset of the capabilities of Internet management. The model is called Common Information Model (CIM) and is described in section 4.7.

The idea of basing integrated management on a single protocol is promoted by DMI's support for SNMP; however, at the same time, the introduction of a new information model obstructs integration. In any case, due to the previous lack of architectural standards for management software on the agent's side, DMI must be considered as a big step towards better manageability of networks and the peripherals.

3.4 Web-Based Management Concepts

Web-based management is one of the buzzwords when it comes to simplifying the management of complex and distributed IT environments. In spite of the hype around the Web, its application within the field of network and systems management is far from being a revolution. There are two different approaches currently found in this area: first, the Web-based Enterprise Management (WBEM) initiative by Microsoft, Intel, Compaq, and BMC (WBEM [971022]); and second, the Java Management API (JMAPI) activities by JavaSoft, BMC, Cisco, Bay Networks, and others (JMAPI [971013]).

The WBEM initiative defines a typical management framework, a management protocol (HMMP; Hyper Media Management Protocol), an information model, CIM, and security proposals. The JMAPI defines components for a distributed management platform based on an implementation in Java. Management applications such as Management Information Base (MIB) (see section 4.1.3) browsers, status monitors and performance monitors are implemented as Java applets.

3.5 Outlook

It becomes obvious that management is a critical and corporate-wide task only to be tackled by use of sophisticated tools and automated processes.

Manufacturers and users have basically agreed to use the DMI standard defined by DMTF.

Additionally, the developments in the area of network computers by Oracle and Sun and trends towards Web-based management products promise a significant reduction of the complexity in management tasks.

4 Survey of the domain

In this section we describe the different technologies we have studied through this masters thesis project. **Fel! Hittar inte referensskälla.** is an overview. In the beginning of every section the relevant part is highlighted. The technologies we have studied are the following:

- **Protocols** - In systems and network management a couple of protocols are used and they are described in this section. The reader need to have basic knowledge in TCP/IP communications to understand this section, the interested reader might read Stevens [9603] for deeper knowledge in TCP/IP. The protocols are SNMP, RMON and DMI.
- **Environments** - This section describe the two environments, WBEM and JMAPI, we have concentrated on. During the research we have found a number of other management environments but they do not have the same support as WBEM and JMAPI and they do not claim to be future standards. The two environments are also in line with Axis strategies.
- **Remote methods** - When management literature is studied different remote methods appear and we describe the most common ones, which are CORBA, RMI, and COM/DCOM.
- **Data models** - There is one model that is accepted by almost every manufacturer in the management area, namely CIM.

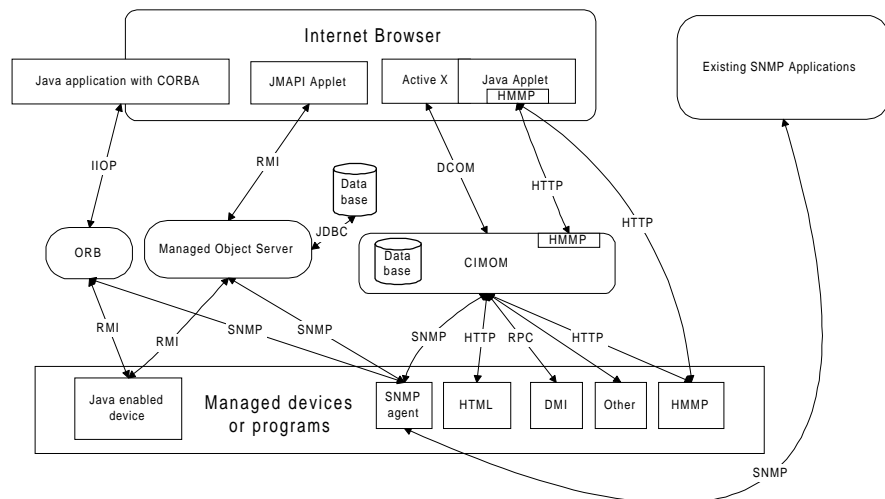


Figure 4-1 Technological overview

Rejected technologies

This section describes technologies often mentioned in network management literature. These technologies or protocols may be obsolete or not implemented in Axis products. In the case of HTTP it is not a management protocol at all but it is used in Axis products to administer and present useful management data. **Common Management Information Protocol (CMIP)** is the protocol used for Open System Interconnection (OSI) management. CMIP management is more complex than TCP/IP management and takes approximately 10 times more bandwidth than SNMP (Tyler). CMIP is very uncommon and not implemented in AXIS products.

HyperText Transfer Protocol (HTTP) is the protocol that is used in web technology to transfer hyperlinked text documents. HTTP is implemented in many of Axis products to manage and present relevant information about the

device. It is possible for the WBEM and JMAP technologies to analyze and process HTTP documents.

4.1 Simple Network Management Protocol

SNMP (Simple Network Management Protocol) is the most widely spread protocol for management of network elements. The SNMP architectural model is a collection of network management stations and network elements. Network elements are devices such as hosts, gateways, terminal servers, and the like, which have management agents¹. The Simple Network Management Protocol is used to communicate management information between the network management stations and the agents in the network elements. SNMP exchanges network information through messages, known as protocol data units (PDUs). The message is an object that contains variables that have both titles and values. This section is built on documents from Stallings [96], Stallings [93], and Seemann [97].

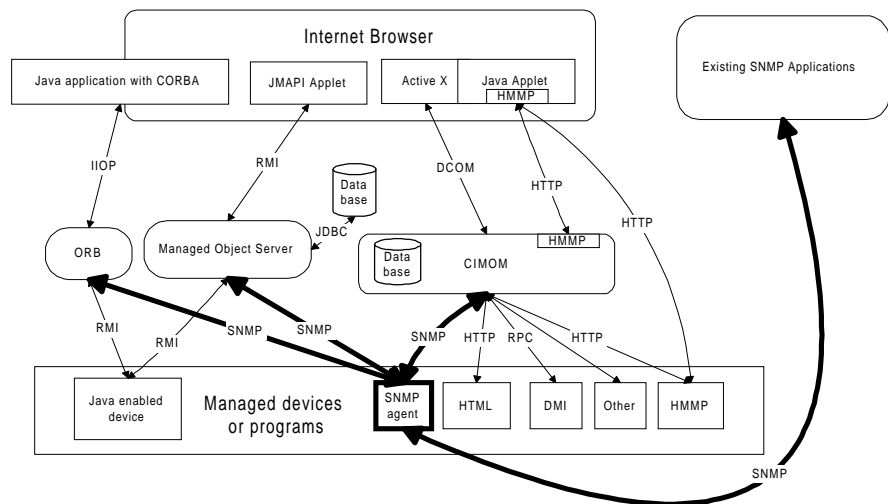


Figure 4-2 Where to use SNMP

4.1.1 Messages

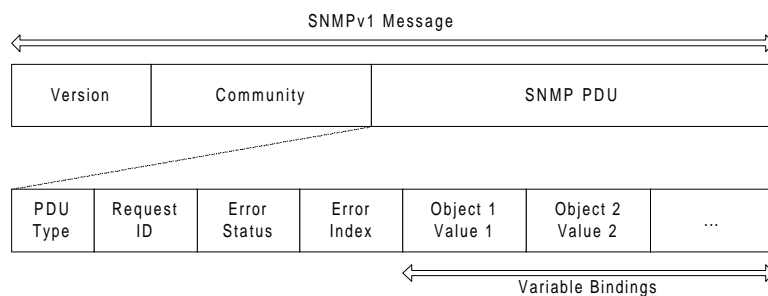


Figure 4-3 SNMP message

¹ An agent is a program that answers SNMP requests and collects data in a network device. An example of SNMP agent can be found in section 4.3.3.

SNMP messages are transported using UDP which is faster than TCP. The message contains two parts. The first part contains the version and the community name. The version secures that all network elements use the same SNMP version. The community name is used to separate messages that belong to different management units in the network.

The second part of the SNMP message specifies the operation and the variables. The PDU type specifies the type of operation. There are five types of PDUs: get and getNext is used to get information from network elements, set modifies variable values in the node, trap is used to monitor network events and getResponse is used in the response from the three first PDUs. Request-ID combines incoming answers with requests. Error-status indicates errors and Error-index specifies which variable the error is in. Variable-bindings contain the variable name and the values. Both integers and strings are used and can be read-only or read-write.

4.1.2 Trap-directed Polling

If a management station is responsible for a large number of agents it becomes impractical for the management station regularly to poll all agents. Instead, SNMP and the associated Management Information Base (see section 4.1.3) are designed to encourage the manager to use a technique referred to as trap-directed polling.

At initialization time, and perhaps at infrequent intervals such as once a day, a management station can poll all of the agents it knows for some key information. Once this baseline is established, the management station refrains from polling. Instead, each agent is responsible for notifying the management station of any unusual event, for example if the agent crashes and is restarted. These events are communicated in SNMP messages known as traps. Once the management station is alerted it may poll the agent in order to diagnose and to gain more specific information about the exception condition.

Trap-directed polling can save network usage and agent processing time.

4.1.3 Management Information Base

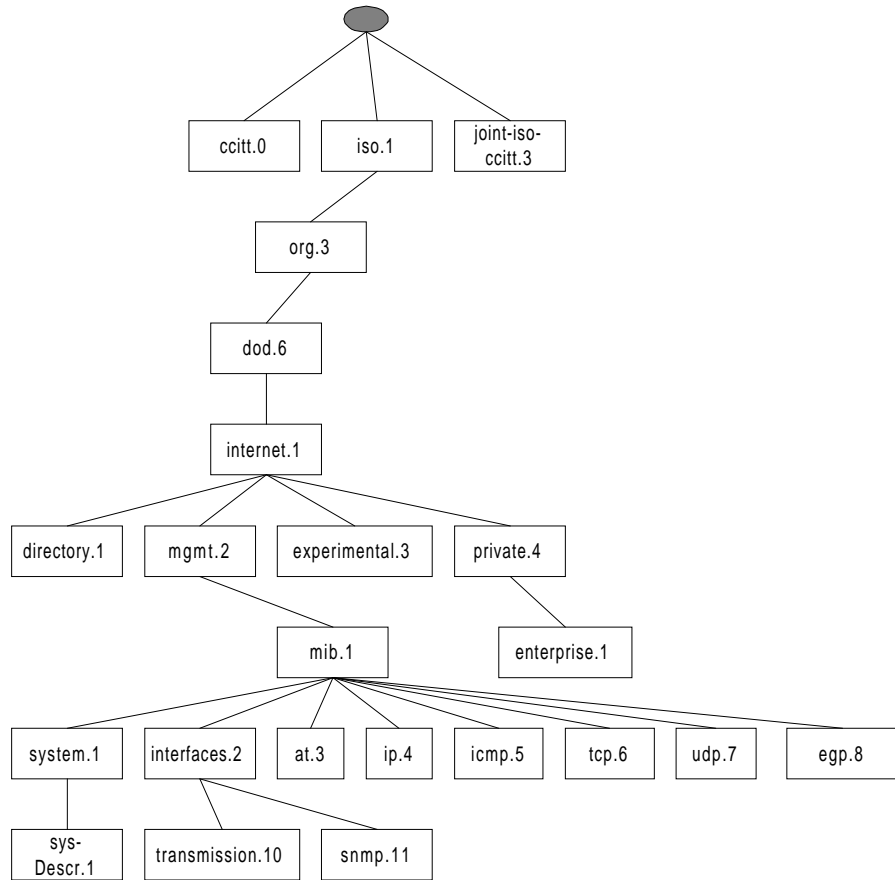


Figure 4-4 SNMP MIB

The network information that SNMP can get and manipulate is defined in a Management Information Base (MIB). A MIB can be described as a tree structure where the variables are located at the end of the branches, see Figure 4-4. The present standard uses MIB-II which contains 171 types of objects and is defined in RFC 1213 (see appendix A). Object identifiers are used to provide a unique name to every object in the tree. The MIB tree has three major branches: CCITT¹, ISO, and the joined ISO/CCITT. The biggest part of the tree is located under the object identifier 1.3.6.1 which is dedicated to Internet, see Figure 4-5. Any company can put their own definitions under one of the branches in the tree. Network administrators can add their own modifications to the MIB. After such a modification the SNMP agent also needs to be modified according to the new MIB. This is the reason why many of SNMP management applications on the market are configurable by the users.

¹ International Consultative Committee on Telegraphy and Telephony

MIB-II

A large number of standard MIB documents are available. In this chapter we look at MIB-II, which is the most important document, which covers a broad range of managed objects.

MIB-II (RFC 1213) is subdivided into the following groups:

- *system*: overall information about the system.
- *interfaces*: information about each of the interfaces from the system to the subnetwork.
- *at* (address translation): description of address translation table for internet-to-subnet address networking.
- *ip*: information related to the implementation and execution experience of IP on the system.
- *icmp*: information related to the implementation and execution experience of ICMP on this system.
- *tcp*: information related to the implementation and execution experience of TCP on this system.
- *udp*: information related to the implementation and execution experience of UDP on this system.
- *egp*: information related to the implementation and execution experience of EGP on this system.
- *dot3* (transmission): information about the transmission schemes and access protocol at each system interface.
- *snmp*: information related to the implementation and execution experience of SNMP on this system.

The *system* group is an interesting group which provides general information about the managed system (see Figure 4-5).

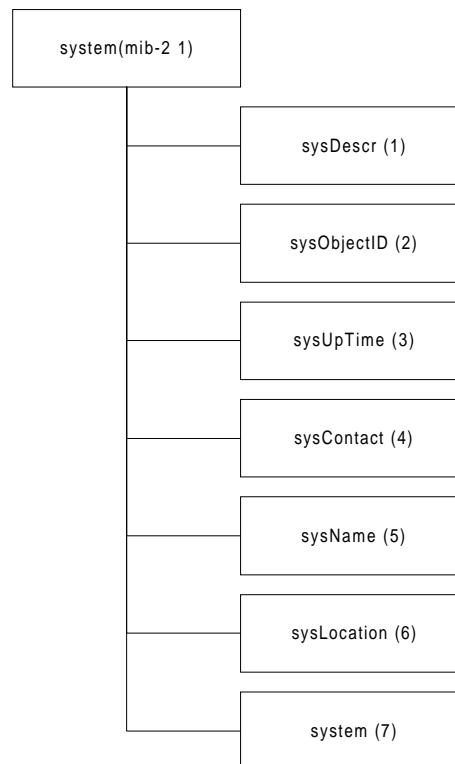


Figure 4-5 MIB-II system Group

Each of the branches contain data about the managed object, see Table 4-1

Table 4-1 System Group Objects

<i>Object</i>	<i>Syntax</i>	<i>Access</i>	<i>Description</i>
sysDescr	DisplayString (SIZE (0 ... 255))	RO	A description of the entity, such as hardware, operating system, etc
sysObjectID	OBJECT IDENTIFIER	RO	The vendor's authoritative identification of the network management subsystem contained in the entity
sysUpTime	TimeTicks	RO	The time since the network management portion of the system was last reinitialized
sysContact	DisplayString (SIZE (0 ...255))	RW	The identification and contact information of the contact person for this managed node
sysName	DisplayString (SIZE (0 ... 255))	RW	An administratively assigned name for this managed node
sysLocation	DisplayString (SIZE (0 ... 255))	RW	The physical location of this node
sysService	INTEGER (0 ... 127)	RO	A value that indicates the set of services this entity primarily offers

4.1.4 SNMPv2

Background

The security framework and the administrative framework in SNMP have had a long and bumpy history (SNMPv2 [960306]). After many years of development and research, three RFCs were presented in July 1992. The new proposal dealt with security and administrative problems but there were many complaints about the complexity. The work with SNMPv2 started the same month. In April 1994 a proposal was published. Although the Internet Engineering Task Force (IETF) granted SNMPv2 the status of Proposed Standard it was not accepted by the industry as a de facto standard. There were many complaints about the complexity and the security, and the framework for administration was often criticized as being too complex for the average network administrator. In late 1994 the working group recommended IESG (Internet Engineering Steering Group) to advance the status of SNMPv2 to Draft Standard. In a meeting in December 1994 people argued changes needed to make SNMPv2 easier to configure and use. However, the group decided not to make any fundamental changes. After a number of meetings and discussions the Working Group was divided into four fractions:

1. The USEC (User-based security) which took a minimalistic approach and focused on the needs for a simple, small agents.
2. The SNMPv2* (SNMPv2 star) which was based on the USEC proposal, but more complete and more correct.
3. The SNMPv1.5/SNMPv2t which couples SNMPv2 protocol operations with the SNMPv1 administrative framework, and as such did not include provisions for authentication, privacy, nor remote configuration.
4. The last major clique was the largest of the four. It contained the "great silent majority" which allowed a relatively small number of individuals to dominate.

As the deadline approached, no decision was made between USEC-style and SNMPv2-style. The Working Group agreed to publish the specification on which there was consensus, named it SNMPv1.5, and immediately continued work on the specifications relating to the security and administrative aspects.

Advantages

SNMPv2 is not implemented in Axis products and that is why it is briefly discussed here. Some of the advantages compared to SNMPv1 are:

- GetBulk, a new operator allowing efficient retrieval of large blocks of data.
- 64-bit counters, for instrumenting rapidly increasing variables such as octet counters in high-speed networks
- SNMP over Novell IPX, AppleTalk DDP, and OSI.

4.1.5 SNMPv3

The SNMPv3 Working Group is now working on the specifications for SNMPv3, and all of the specifications are in draft status which means that they can change in the final version. Some of the goals are:

- Keep SNMP as simple as possible.
- Use existing materials as much as possible i.e., SNMPv2u, and SNMPv2*
- Make it possible to move forward in the standards track, even if consensus has not been reached on all pieces.
- Make it inexpensive to deploy a minimal implementation.

4.2 Remote Network Monitoring

The most important addition in SNMP is the Remote Network Monitoring (RMON) specification. It provides the network manager with important information about the network and is an MIB that supplements MIB-II. Even if it is simply a specification of an MIB it provides remarkable features and a significant expansion to SNMP functionality without any changes to the underlying SNMP protocol.

4.2.1 RMON Goals

The RMON MIB is an attempt to define a standard for network monitoring functions and interfaces for communication between SNMP-based management consoles and remote monitors. RMON provides an effective and efficient way to monitor subnetworks while reducing the burden on both other agents and management stations. Some of the design goals that RFC 1757 lists are:

- **Off-line operation:** It is desirable to limit the polling of a monitor done by a network manager. Limited polling saves communication costs, especially when dial-up lines are used. In general, the monitor should collect fault, performance, and configuration information continuously even if the polling have ceased. The monitor continues to accumulate statistics that can be retrieved later and it can also attempt to notify the management stations if an exceptional event occurs.
- **Proactive monitoring:** If the monitor has sufficient resources it can continuously run diagnostics and log network performance. In the event of a failure somewhere on the Internet, the monitor can notify the management station with relevant information to diagnose the failure.
- **Multiple managers:** A network may have more than one management station and the monitor can be configured to handle this.
- **Value-added data:** The network monitor can analyse the data it has collected and relieve the manage station. For example: On one subnet a workstation may be generating the most traffic, which the monitor can tell the manager. This is not possible for the managing station to determine without being connected to the subnet.

Not all network monitors support all of these goals, but the RMON specification provides the base for it.

4.2.2 Control of Remote Monitors

A remote monitor can either be a dedicated device or a function available in a system. The RMON MIB contains features that support extensive control from the management station. These fall into two general categories: configuration and action invocation.

Configuration

Typically, a remote monitor will need to be configured for data collection. The MIB is organized into a number of functional groups. Within each group, there may be one or more control tables and one or more data tables. At configuration time, the management station sets the appropriate control parameters to configure the remote monitor to collect the desired data.

Action invocation

SNMP provides no specific mechanism for issuing a command to an agent to perform an action. The only capabilities of SNMP are to read and set object values. It is possible to use the SNMP set operation to issue a command; a process called Action invocation. An object can be used to represent a command, so a specific action is taken if the object is set to a specific value. The RMON MIB contains a lot of such objects.

4.3 Desktop Management Interface

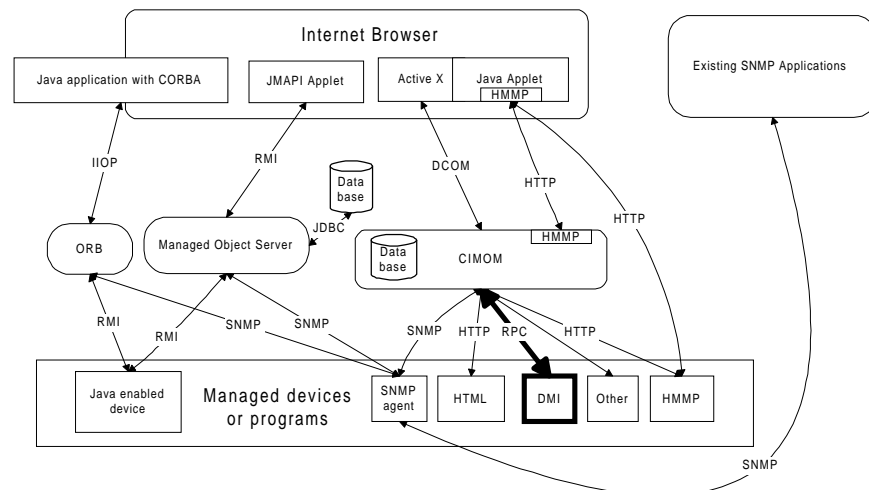


Figure 4-6 Where to use DMI

This section is about the Desktop Management Interface (DMI) (DMTF [960326]). Throughout this section DMI ver 1.0 and ver 1.1 are called DMI ver 1.x.

4.3.1 Background

Since there is a gap between managing network peripherals with SNMP and managing a desktop computer, DMTF has produced the standard DMI. It is designed to both be used locally, without a network connection, and remotely (with ver. 2.0). The desktop system often appears in a wide variety of configurations in an enterprise and from different vendors, all with their own proprietary configuration tools. It would be of significant improvement if a computer administrator could use the same tool to configure and maintain the different systems even though they are not networked.

More than 200 products from major vendors are already DMI-enabled. (Byte [9710]).

4.3.2 Structure

DMI can be divided into four elements:

- **Management Information Format (MIF)** This is the format DMI uses to describe the managed component and its properties. DMTF provides MIF files for PCs, servers, printers, software applications, and mobile devices. Vendors can DMI-enable products by providing the appropriate information in MIF files.
- **Service Provider (SP)** This is the engine and is placed in the middle of the architecture. The SP acts as a database server and updates a database with information on the current status of the managed component. The initial state is the MIF. During the managing session both the components and the managing application can access the database, and the SP has to take care of serialisation of commands from both sides and check the database for inconsistency. It is up to the programmer of the SP to design an internal database.

Events and indications travel through the SP and the SP delivers the different events to the components that have registered their interest in receiving specific information.

The SP is placed in the managed object.

- **Two Application Programming Interfaces (APIs)** for communication, one between Managed Components and the Service Provider and one between Managing Applications and the Service Provider.

This is the interfaces of DMI and an implementation has to consider the specifications of return values (i.e., error codes) that DMI uses. DMI version 1.x uses a block interface to communicate (“DmiInvoke”) and has the same syntax, regardless of the device, the software or the hardware that it is communicating with. In DMI version 2.0 there are several procedures defined and each managed component has to implement the appropriate Component Interface (CI) (see Figure 4-7)

- **Remote Procedure Call (RPC)** interface for remote communication. DMI supports DCE/RPC, ONC/RPC, and TI/RPC. See appendix F for more information about RPC.

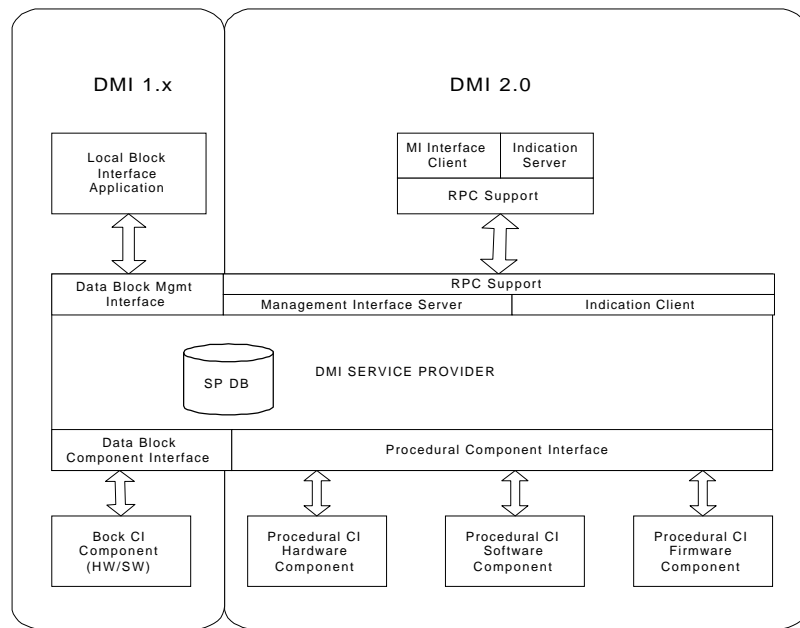


Figure 4-7 Functional block diagram of DMI

4.3.3 Existing DMI applications

A typical use of DMI 1.x is represented by the IBM SystemView SNMP agent (IBM [9604]). The agent installed at the workstation speaks DMI with the underlying devices and translates the information to appropriate SNMP information readable for any generic SNMP management application.

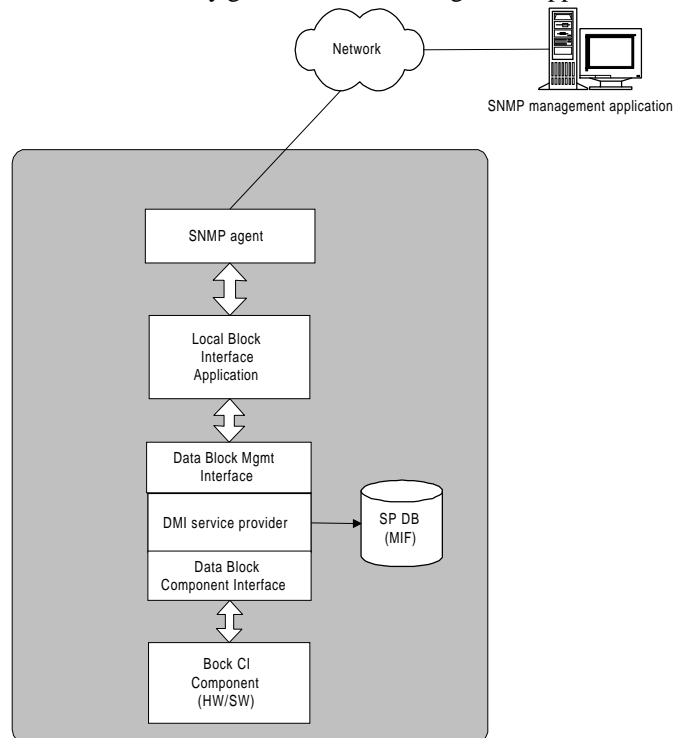


Figure 4-8 Functional block of IBM SystemView SNMP agent

The Tivoli Management Framework (Lendenmann [9701]) exports inventory information of an enterprise using another part of DMI: the MIF. The MIF format makes the information about the installed devices across the company network accessible for other applications.

4.4 Web Based Enterprise Management

This section describes the Web Based Enterprise Management (WBEM) architecture (Microsoft [971031]). WBEM is Microsoft's efforts to form a standard for managing devices across a network.

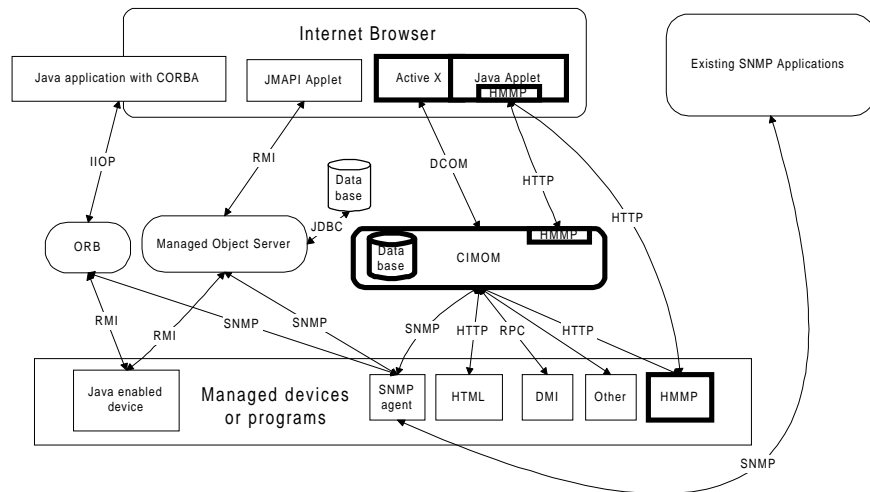


Figure 4-9 Where to use WBEM

4.4.1 Introduction

WBEM is designed to be compatible with the major existing management protocols, including SNMP, DMI, and CMIP. The idea with using WBEM is that network administrators manage the network and the connected nodes using a web browser.

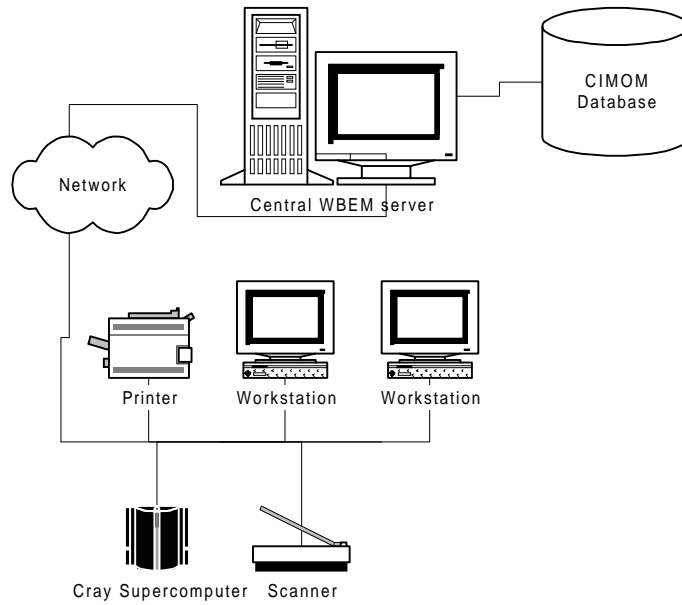


Figure 4-10 WBEM Overview

4.4.2 WBEM architecture

The WBEM architecture, see Figure 4-10, defines the structure of and the conventions to access information about managed objects, store information for analysis and methods for managed objects manipulation. The major components are:

- **Common Information Model (CIM)** A model defined by DMTF used to define and manipulate managed objects. CIM is described in a separate section, see section 4.7.
- **CIM Object Manager (CIMOM)** An object manager providing the interface between management applications and managed objects.
- **HyperMedia Management Protocol (HMMP)** A protocol that defines the way object managers, management applications, and managed objects communicate with each other.

The WBEM architecture is designed to allow management products from multiple vendors. No single product will provide the whole range of management services, such as network inventory mapping, software distribution, capacity planning, and node managing. Therefore WBEM allows multiple instrumentation at the client side.

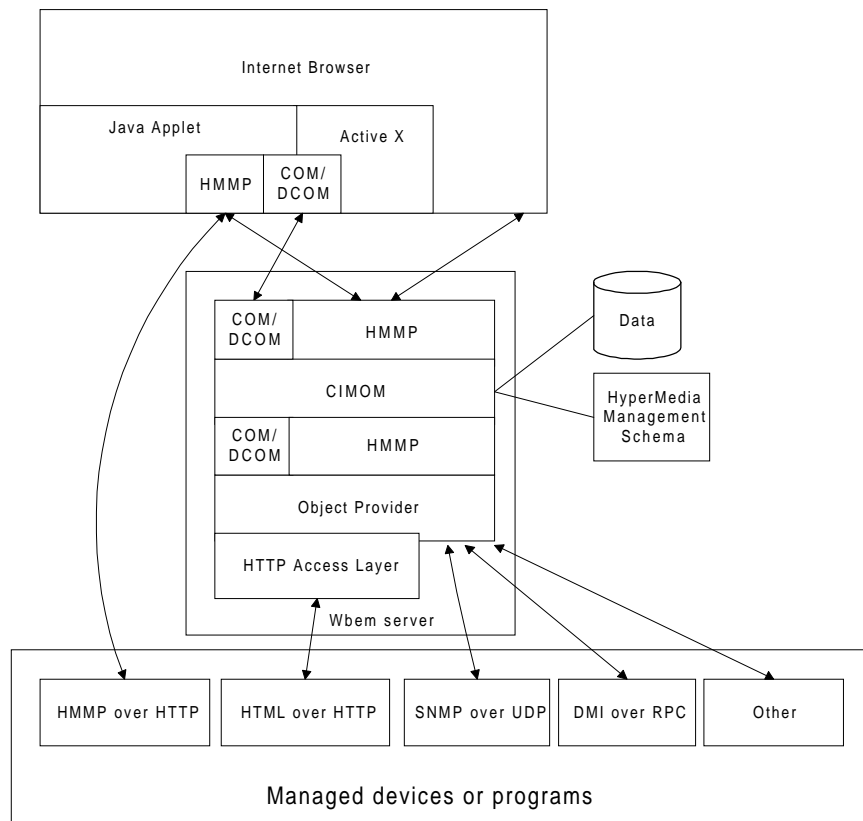


Figure 4-11 WBEM detailed architecture

Internet Browser

An Internet browser such as Netscape Communicator or MS Internet Explorer is the WBEM operator’s user interface. The browsers communicate with the CIMOM database or directly with the managed objects via a Java applet, see Figure 4-11. An Internet Browser controls even the WBEM Developer studio user interface. It contains Active X and COM requests, so the only supported browser here is MS Internet Explorer.

Hyper Media Management Protocol (HMMP)

HMMP is the protocol used to communicate between different parts of WBEM. Microsoft’s goal is to get HMMP to RFC status but it has not yet come through proposal stage. The discussion in hmmp-list@freerange.com indicates that the HMMP developing is stalled. An example of this is that hmmp-list has changed name to wbem-list@freerange.com. For more information about HMMP, see WBEM [971022].

WBEM CIM Object Manager (CIMOM)

CIMOM is a service that responds to requests from applications to change the information in the database. A change request might be a request to change the data for a managed object, restructure the database, create classes and instances, or add and retrieve data. A method or operation is a small program stored in the CIMOM database that is called under special circumstances. This is not supported in the current version of WBEM.

The database communicates with the managed objects via providers.

The Managed Object Format (MOF) makes it possible to add data to CIMOM from a textual file. For example, it is possible to translate a MIB to a MOF and load it into the CIMOM database.

Object provider

The object provider translates CIMOM requests to an appropriate way to communicate with the managed objects. The WBEM SDK 2.0 Beta includes a SNMP provider that makes it possible for CIMOM to communicate with SNMP agents in a managed device.

WBEM SDK provides a framework to develop a customized provider, but current versions of SDK do not support this.

HTTP access layer

Some devices have their own HTTP servers that are used to manage the device. The HTTP access layer translates CIMOM requests to appropriate HTTP calls.

4.5 Java Management Application Programming Interface

The Java Management API (JMAPI) is an extension to Java and a collection of Java language classes and interfaces that allow developers to build systems and network and service management applications (JMAPI [9705]).

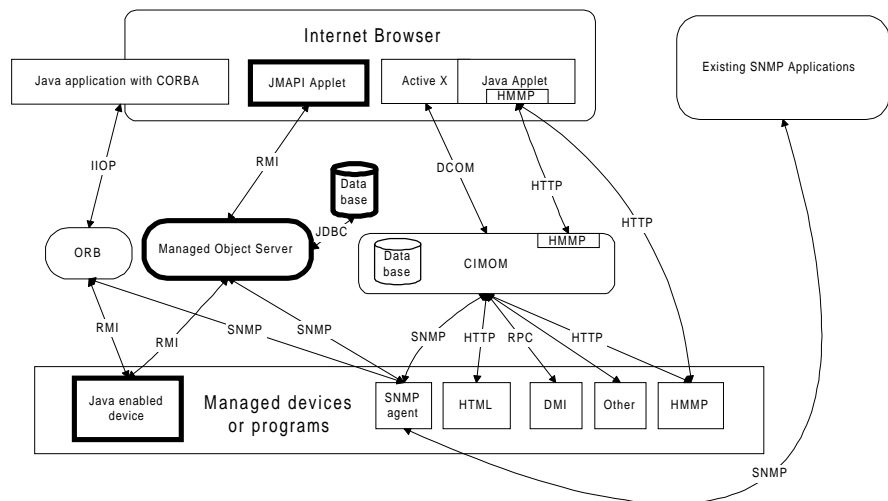


Figure 4-12 Where to use JMAPI

4.5.1 JMAPI high level model

The architecture contains the following components:

- **Java-enabled browser**, this is the user interface from where the administrator issues management operations.
- **Managed Object Server**, provides management objects to applications. It includes agent object interfaces, SNMP interface, managed data interface, Java DataBase Connectivity (JDBC) interfaces, a database and an HTTP server.

- **Appliance** consists of the resource that is being managed, which can range from large servers to a network element.

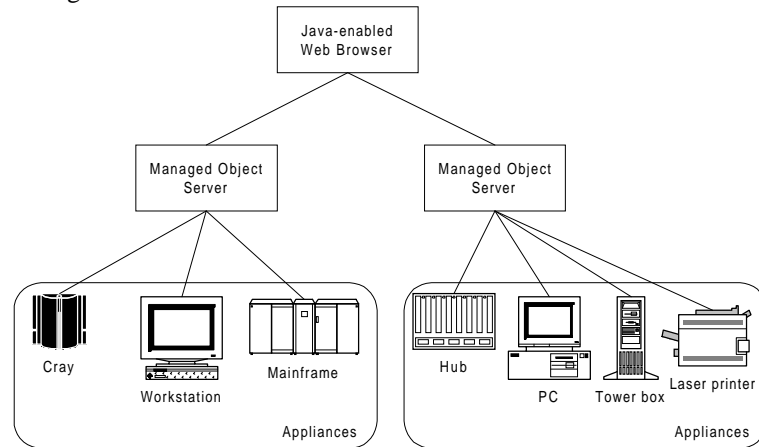


Figure 4-13 JMAPI High-Level Architectural Components

Figure 4-13 shows a configuration with two Object Managers and one Java-enabled Web browser. It also shows a number of different managed types.

4.5.2 JMAPI features

By using JMAPI we get the following benefits:

- **Write Once, Run Anywhere.** Java Virtual Machine (JVM) runs on a number of operating system.
- **Eliminates agent versioning and distribution problems.** The enterprise management environment is truly distributed. This causes problems for the administrators who have to perform operations throughout the network. If a new capability is added, the agent has to be replaced.

JMAPI provides a solution for these versioning and distribution problems. The correct versions of classes and libraries are securely downloaded.

- **Secure distributed management operations.** The JVM ensures that only trusted Java code runs on a client. This means that JMAPI allows secure management operations. It safeguards remote managing from unauthorised accessing and spoofing by demanding authorisation and authentication for all remote invocation.
- **Protocol independence.** It is possible to develop an agent that uses different protocols. Standard protocols, like SNMP, are already implemented in JMAPI and ready to be used immediately.

4.5.3 JMAPI Elements

A programmer must write components for each of the elements in the JMAPI model. For the Java-enabled browser the programmer must construct Java applets using classes from the Admin View Module (AVM) and Managed Object Server (MOS) packages. On the MOS, managed objects must be implemented. Figure 4-14 shows the elements and their relationships.

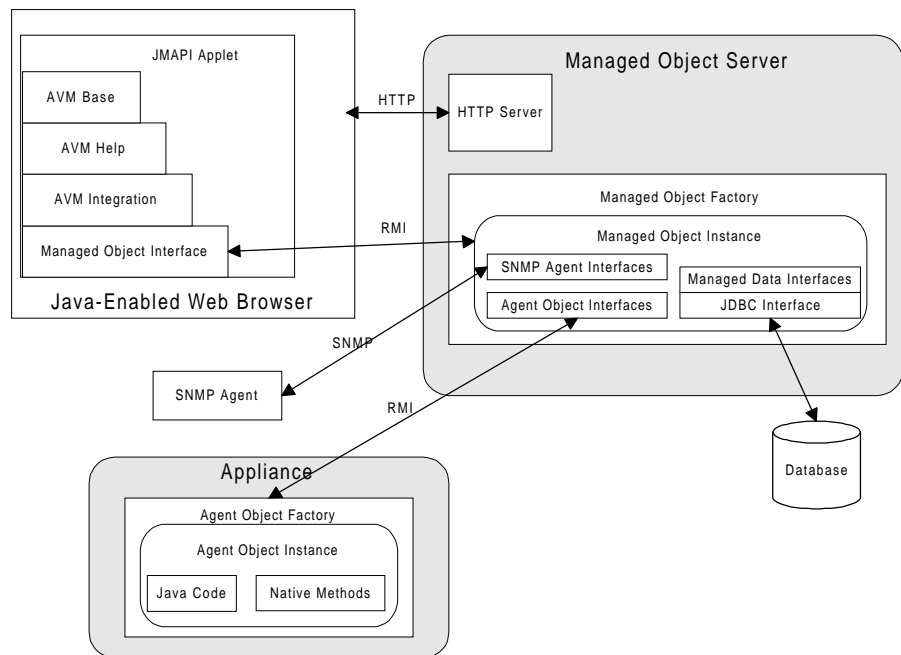


Figure 4-14 JMAPI Elements and Interfaces

The components can be running in different address spaces and on different machines. This requires the ability to communicate. The different components use Remote Method Invocation (RMI), see section 4.6.4, to communicate with each other. Java support sockets for basic communication, but then a protocol for exchanging information between the client and the server needs to be designed. The implementation of such a protocol is often cumbersome and hard to debug. RMI has been designed to operate in a Java environment and is relatively easy to use.

Java-enabled Web Browser

The browser is the interface between the JMAPI user and JMAPI. The JMAPI applets are downloaded into an administrator's Web browser. The administrator then interact with the applet and performs management operations.

Admin View Module

The Admin View Module (AVM) consists of classes and methods for the client-side. AVM's role is to provide the user interface classes and the application-level functionality. The AVM is built on top of Java's Abstract Window Toolkit (AWT) and is therefore portable and window-system independent. The AVM is divided into three parts: AVM help, AVM Base, and AVM integration.

The *AVM Base* is specifically designed for developers creating management solutions and is divided into three packages: the user interface, the power user interface, and the management user interfaces classes.

The user interface part is used to build rich user interfaces and defines the following classes:

- Image button
- Scrolling windows and panels
- Toolbar
- Image canvas
- Convenience dialogues
- Busy tool.

The power user interface part is used to develop more graphical components and the following classes are defined:

- Table
- Hierarchy browser
- Charts and meters.

The management user classes support user interface style and interaction model and the following classes are defined:

- Property books
- Page header
- Link label
- View configuration.

The primary goal of the *AVM Help* is to provide a general help environment, which enables JMAPI developers to develop on-line help. The AVM help classes depend on AVM base classes, Java AWT, and Java base classes. Therefore, the AVM help can be used without dependencies on the underlying managed objects. In the future AVM Help will be deleted.

The *AVM integration* provides integration between managed object interfaces and AVM base classes. AVM integration classes allow management applications to register and unregister their components.

Managed Object Interfaces

Managed objects are used to provide abstraction of resources and services; they use RMI to perform remote management.

4.5.3.1 Managed Object Server

The managed object server (MOS) is a host that provides three basic services:

- An HTTP server
- An Managed Object Factory
- A Managed Object Interface.

The *HTTP server* handles HTTP requests for a JMAPI domain. An HTTP request is sent from the web browser to the server and the http server sends the requested document or applets to the client.

The interfaces to the *Managed Object Factory* (MOF) allow clients to create new object instances or to retrieve sets of objects through the database interface. Managed objects are RMI remote objects that are created and maintained by the MOF. JMAPI applets access managed objects through interfaces that allow the following basic operations:

1. Establish a transaction connection to perform the operation
2. Prepare the database for any updates
3. Invoke perform action (which is add, delete or modify)
4. Commit database updates
5. Close transaction connection.

The *Managed Object Interface* consists of four parts:

1. **The Managed Data Interfaces** To manage all the instances of managed objects some Java classes are introduced and put in the Managed Data Interface. These classes constitute the interface between the database API, the SNMP agents, and other agent interfaces.
2. **The JDBC-Interface** The Java DataBase Connectivity (JDBC) API defines Java classes that are used to represent database connections, SQL statements, result sets, and database metadata. It allows a Java programmer to use SQL statements and process the results. JDBC is the primary API for database access in Java.
3. **The Agent Object Interfaces** Agent objects are RMI objects that run on the appliance (see section 4.5.3.2). When a method is called in an agent object, it runs Java code or native methods to perform the management

operation.

The agent object interfaces are similar to those provided by the managed object factory.

4. **The SNMP Agent Interface** JMAPI has built-in support for SNMPv1 agents and the following parts are provided for the integration of JMAPI with existing SNMPv1 agents:
 - A set of Java classes that implement the SNMP protocol and the interfaces to use SNMP.
 - A set of JMAPI-managed objects to facilitate the usage of the Java classes that implement SNMP.
 - A JMAPI-managed object that acts as an SNMP trap handler that receives all SNMP traps and converts them into instances of JMAPI events.

4.5.3.2 Appliance

Appliance is a device with a Java machine inside which communicates with the agent object interface via RMI. For most managed object classes there are some values that will need to be changed. These may be changes to parameters and configurations.

Agent Object Factory

The agent object factory provides instantiation service on appliances. Requests are made to the agent object factory for agent objects of a particular class. The factory performs the instantiation of the RMI object and returns a remote handle to the requesting client.

Native Methods

Most often the agent object methods are concerned with updating or retrieving system-specific states related to managed object. You may not always be able to update system-specific state information in Java. In many cases, management operations are platform specific. In these cases, the agent object methods must be implemented as Java Native Methods.

Java Code

The Java code is the program running in the appliances JVM. It communicates with the native methods to retrieve and change parameters, and it also communicates with the MOS to present and store the values.

4.6 Remote Methods

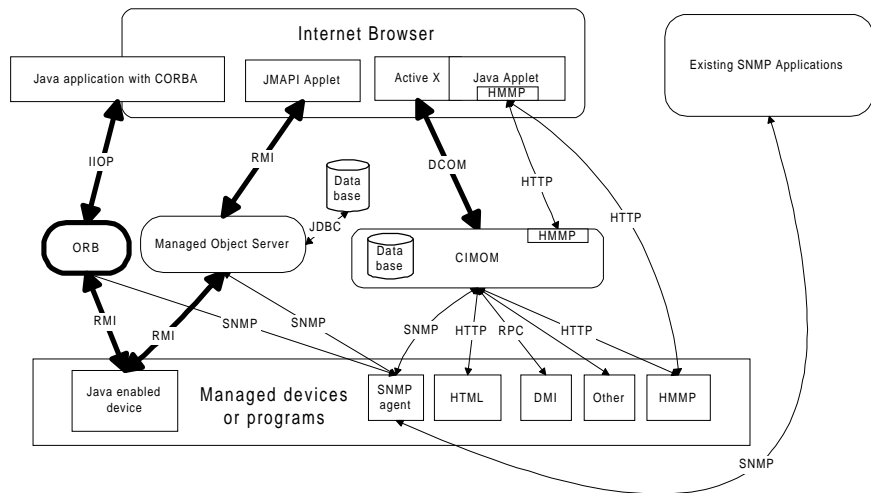


Figure 4-15 Where to use Remote Methods

The three different technologies Distributed Common Object Model (COM), Common Object Request Broker Architecture (CORBA) and Remote Method Invocation (RMI) solve the same problems but in slightly different manners. COM and CORBA are language-independent interface specifications and RMI is a native method for remote methods call in a specific language (Java). Which method to use is a popular discussion topic in the computer press. The different bridges between the technologies populating the market are not making the decision easier. This section will first describe the main principles of COM and CORBA (these are equivalent concepts) and later we will point out the differences. In the end we will describe RMI and its connection to the other methods.

The reason to use remote methods comes from the diversity of the equipment that is available in today's enterprise networks. To be able to use resources on different platforms from different vendors seems to be a good reason to use remote methods and thereby minimize the cost of hardware and software. Remote Procedure Calls (RPC) (see appendix F) is a popular method for invoking remote functions and procedures, but it is based on the procedural way of seeing the world (or the "C - way"). Today we live in an object-oriented world and therefore "plumbing tools" have to be object-oriented as well; and suddenly there are two competing technologies to spread objects around an enterprise. One way of using the remote methods is to implement reusable components such as ActiveX (<http://www.microsoft.com/activex>) and JavaBeans (<http://www.javasoft.com/beans>) but this very interesting subject is beyond the scope of this master thesis.

4.6.1 Basic concepts

The B in CORBA stands, as mentioned earlier, for Broker and this is what it is all about: brooking objects around an enterprise. Both COM and CORBA use an Object Request Broker (ORB) to communicate with the clients (see 4-16). The client simply asks the ORB for an object it needs and hopefully it gets an answer about were the object is and how it can be used because both models support methods to investigate an object. The calling application does not need to know how to use the object. The basic principle is very simple: ask for an object and, if you are lucky you will get it.

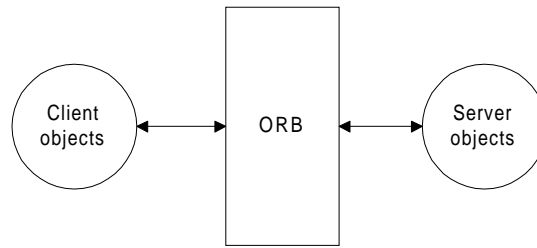


Figure 4-16 Basic concept of remote methods

4.6.2 CORBA

Object Manager Group (OMG) (Byte [9708]) began the work with CORBA 1991 but a working implementation has not been available for too long. OMG consists of more than 800 companies and organizations today, which may be the reason why. The first specification does not explicitly specify which network protocol an implementation must use only a General Inter-ORB Protocol (GIOP); it was left as a matter for the implementor. To meet the demands of the increasingly growing Internet OMG defined CORBA 2.0 with suitable Internet standards.

An ORB should be able to handle requests from different clients and platforms at the same time and transmit the request to the appropriate server. The Internet version of the GIOP is called the Internet Inter-ORB Protocol (IIOP) and the fact that Netscape Communicator uses IIOP has made CORBA a technology available to the masses. IIOP is basically GIOP through TCP-sockets. Since CORBA is an object-oriented technology it supports inheritance that is, different interfaces can form an inheritance hierarchy. The interfaces are specified in an Interface Specification Language (OMG IDL) which is based on the Open Software Foundation Distributed Computer Environment Interface Definition Language (OSF DCE IDL)(The Open Group). The standard specifies mappings between OMG IDL and several languages (C, C++, Smalltalk, Cobol, and ADA) and between CORBA and COM. It is not necessary for a developer to implement all the functionality of the CORBA specifications, especially with the language mappings it is common to only implement the C++ and Java mappings (Jacobsson [9803])

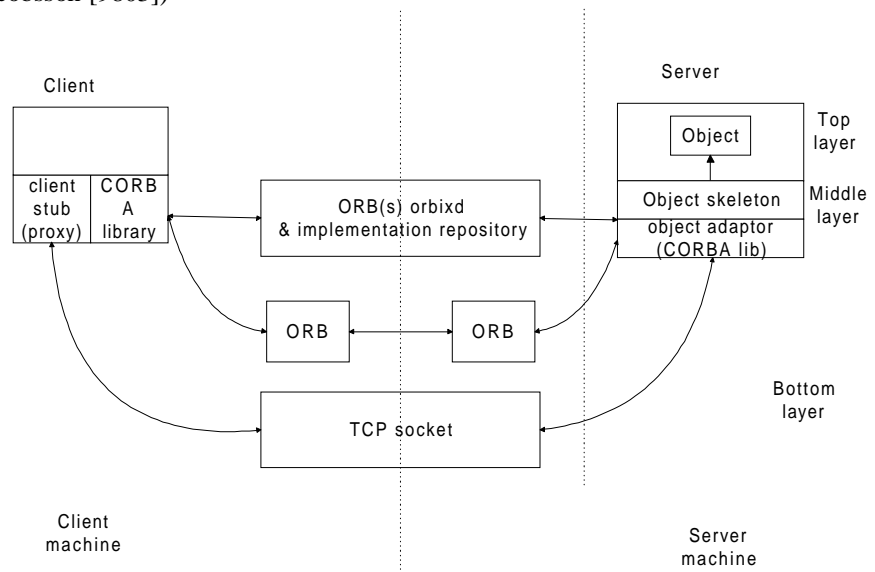


Figure 4-17 Overall structure of CORBA

4.6.3 Distributed Common Object Model

Microsoft and Digital are developing the Common Object Model (COM) (Rogerson [97]). In this thesis we look at the distributed version of COM called Distributed COM (DCOM). Today DCOM is used on Microsoft Windows platforms, but versions for several other operating systems such as Linux and Solaris (Byte [9704]) are coming. DCOM has received its credibility via the ActiveX components and the fact that all of Microsoft's products use the technique.

On the low level DCOM uses OSF DCE RPC (The Open Group) to transport its messages and in the Internet draft (Microsoft [9801]) DCOM is called Object RPC (ORPC). Microsoft and Digital have extended RPC with an Object Pinging mechanism for the possibility to check if an object is alive. The ORB in CORBA has its counterpart in DCOM and namely the Registry¹ in Windows 95/98 and Windows NT 4.0/5.0. Even though DCOM is an object-oriented technology it does not support inheritance but instead it is possible to use several different interfaces to each object. The tool to define an interface is Microsoft Interface Definition Language (MIDL), which is based on OSF DCE IDL, the same parent as OMG IDL with some added features. Exception handling is not implemented as event handling in DCOM but instead every object has to return an HRESULT, a 2-byte integer.

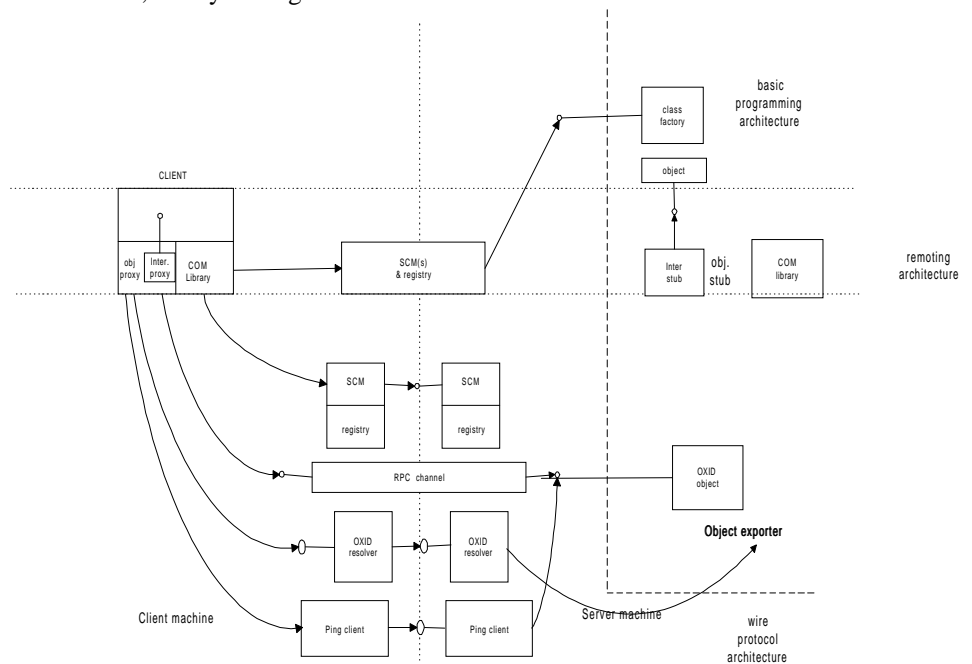


Figure 4-18 Overall structure of DCOM

4.6.4 Remote Method Invocation

In contrast to CORBA and DCOM, RMI is tightly connected to a specific language: it implements remote methods in Java. The main benefit of a

¹ The Registry in Windows is used to locate different components of the Windows (Windows 95 and above) operating system and tuning the different parameters.

language-dependent technology is that it is easy to implement distributed computing in a program. The syntax does not differ from a local method, the interface definition language is the language itself, and especially with Java things like marshaling¹ is not a problem due to the virtual machines. Obviously there is at least one big disadvantage; Microsoft dislikes it. DCOM is the official way of implementing distributed computing on Windows platforms and is considered as a part of the operating system. Supporting RMI would be like making Windows unnecessary. Never the less there are workarounds to use RMI with Microsoft products.

Finding remote objects in RMI is like browsing the Internet. There has to be a server that the client program can access with an URL like naming:

rmi://lysator.liu.se:4711/acme (calling the object acme on host lysator.liu.se available on port 4711)

There are several more aspects of remote methods to deal with such as security issues, performance, and future possibilities but these are beyond the scope of this thesis. For the interested reader we recommend (Jacobsson [9803]).

4.7 Common Information Model

This section describes the Common Information Model (CIM) and is based on version 2.0 (DMTF [980303]) and its adjacent documents (DMTF [971222:1], DMTF [971222:2], and DMTF [971226]). CIM has not been implemented in JMAPI yet and is therefore colored in gray, see Figure 4-19.

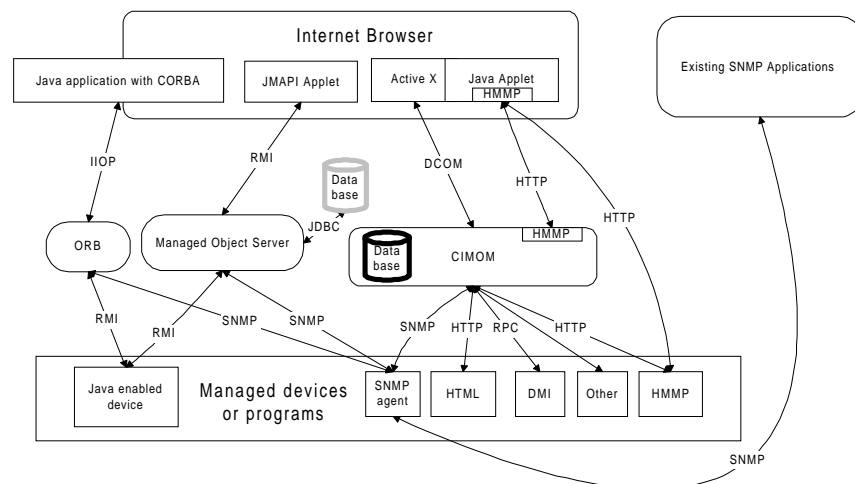


Figure 4-19 Where to use CIM

The DMTF Common Information Model represents an effort to formalize management information and make it an open standard accepted everywhere. The members in the CIM sub committee are:

- Compaq Computer Information
- Computer Associates Intl., Inc.
- Hewlett Packard Company
- Intel Corporation
- Microsoft Corporation
- Novell, Inc.
- Sun Microsystems, Inc.

¹ An example of marshaling is to convert between Little Endian and Big Endian.

- Tivoli Systems, Inc.

This collection of companies will probably guarantee the success of CIM.

There are no standards today for describing management information. There are several standards for managing communication between the managed object and the managing applications, two examples are SNMP and CMIP. These standards describes of how to communicate and what to exchange information about but nothing about how to store information and exchange information between management applications. DMTF hopes to fill this gap with Common Information Model

CIM uses an object-oriented paradigm to structure and conceptualize management data. The approach uses the Unified Modeling Language (UML) (ses appendix E) guaranteeing the information to be presented in a well-known manner. The management schema is divided into three conceptual layers (see Figure 4-20):

- **The Core model** - an information model that captures notions that are applicable to all areas of management
- **Common model** - an information model that captures notions that are common to particular management areas but independent of particular technology or implementation. The common areas are systems, applications, databases, networks, and devices. The information model is specific enough to provide a basis for the development of management applications. This model provides a set of base classes for extension into the area of technology specific schemas. The core and Common models together are expressed as the CIM schema.
- **Extension schemas** - represent technology-specific extension of the Common model. These schemas are specific to environments, such as operating systems (for example, UNIX or Microsoft Windows)

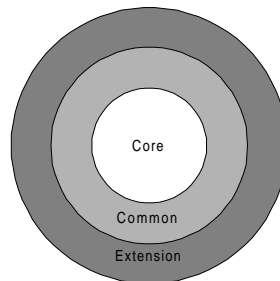


Figure 4-20 The layers of CIM

The mapping between CIM and different technologies is described in several generic ways. The mapping between the DMTF MIF (see section 4.3) and CIM has its own chapter in the specification. A mapping could be described as the actual instances of the classes described in CIM. For example, in an application such as a Data Base Management System (DBMS), the CIM is mapped into the physical schema of a targeted DBMS. The information stored in the database consists of actual instances of the schema.

The management information is described in a language based on an Interface Definition Language (IDL) called the Managed Object Format (MOF) The syntax of MOF is a considerably large part of the CIM specification and is the most concrete view of CIM.

4.7.1 Core Model

There are no mappings from the Core Model to existing technologies such as SNMP due to the abstract nature of the Core Model. The classes specified are considered as the very basic ones and should be a small set of classes. Figure 4-21 shows the existing classes and their relations and associations. The elements that constitute a system can be:

- **Physical** - they occupy space and must conform to the elementary law of physics
- **Logical** - they represent abstractions used to manage and co-ordinate aspects of the physical environment. Logical Elements typically represent system states or capabilities of a system.

Logical Elements can be:

- **Systems** - a grouping of other logical elements. Because Systems are themselves Logical Elements, a system can be composed of other systems.
- **Network Elements** - a set of classes used to provide a topological view of a network.

Every CIM conformant schema is expected to have inheritance relationships with the Core Model.

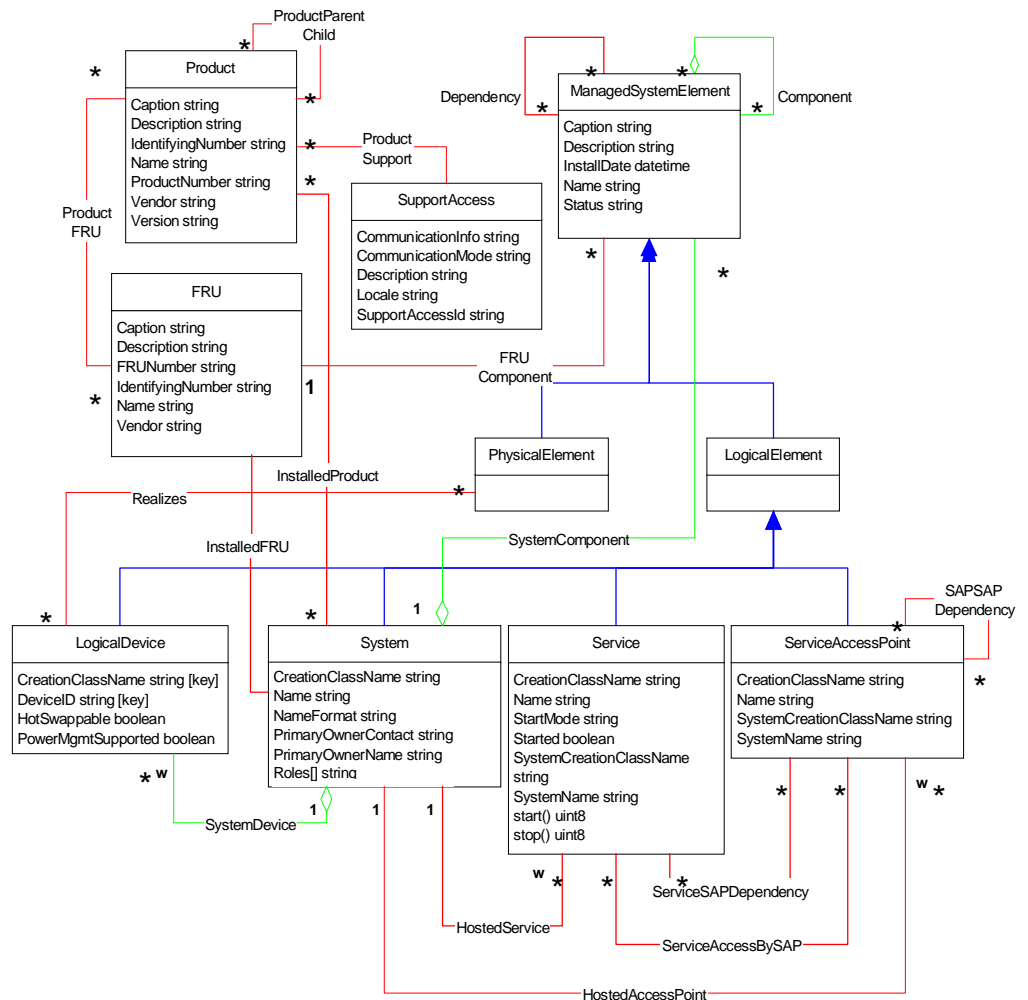


Figure 4-21 CIM Core Model (from DMTF [971222:1])

4.7.2 Common Model

At present time there are two proposals for the Common model: a Common model for application management and a Common model for systems management.

Application management - deals with issues such as software distribution and versioning. It describes which actions to do when installing software and were to install it (e.g., directories, registries, etc.). The model consists of four high level concepts: Software Product, Software Feature, Software Element, and Application System. Only Software Elements are thoroughly described in the most recent version of the schema.

Systems management - This model defines how to describe the management of computer systems and divides the class Computer System into the subtypes Unitary Computer System and The Cluster:

- The **Unitary Computer System** is a representation of a desktop, mobile, Net PC, server, or other type of single node Computer System. It is assumed to occupy a single physical location and to support the execution of a single operating system at a given point in time. Examples of Unitary Computer Systems are desktop or mobile systems running Windows or UNIX. Note that if a critical component of the Unitary Computer System (for example, the operating system) is removed, the Computer System ceases to exist, although the hardware and firmware still exist.
- The **Cluster** represents a Computer System (CS) that is made up of other Computer Systems. These other Systems operate together as an atomic, functional whole to increase the performance and/or resources of the component Computer Systems, related to some aspects of these Systems. Usually the components of the cluster, i.e., the node Systems, can be removed or added without breaking the cluster, as long as a cluster-specific minimum number of elements is maintained. The Computer Systems that participate in a Cluster are defined using the Participating CS dependency association. A Computer System can be a member of more than one Cluster.

4.7.3 Extension schema

Today there is only one extension schema available which is the Win32 schema provided with the WBEM distribution.

5 Results

This sections contains a comparison between WBEM and JMAPI in the present. The comparison may seem unfair because we have not tested both environments but this is most a theoretical comparison and in the future they will probably look different. In section 5.7 we explain what happened with the prototype we mentioned in section 1.

5.1 Database

Both WBEM and JMAPI use databases to store management data about the managed objects.

WBEM is using the CIMOM database included in the WBEM package. JMAPI supports databases that communicate via JDBC, the standardized database communication protocol defined in Java. CIMOM need NT or Win95 as platform and is not portable to other operating systems. A JDBC database can run on a number of platforms, for example NT, UNIX, AS/400, and MVS. Another advantage with using a standardized way to communicate with the database is that the developers of management environments, in this case Microsoft and Sun, can concentrate in only developing management environments and do not need to put a lot of effort in constructing the database. A problem with JDBC, that does not exist in CIMOM, is that it is harder to install. To make the JDBC connections to work properly there are a whole bunch of configuration files that need to be correct. To make CIMOM to work, just download the WBEM package, unpack it and run the installation program.

5.2 Documentation

The documentation differs a lot between WBEM and JMAPI. WBEM only has on-line manuals and they use the Microsoft help. This manual only gives a brief introduction to how the different parts interact with each other and there are a lot of errors. The API descriptions are scanty and full with errors, the example code in the documentation is not tested, it is hard to penetrate and full of bugs. It is almost impossible to get an overview of WBEM. The only documentation that is error-free is the example code that comes with the WBEM package. However, it is almost unintelligible.

The JMAPI documentation is better structured and easier to read. It is split into different parts that are described in a documentation roadmap. The documentation can be downloaded in different formats such as HTML, postscript, and PDF.

5.3 Events

A real important piece in management is events. With events it is possible to build a hierarchical architecture. For instance, a tree could be constructed to monitor financial markets (see Figure 5-1). You could register with the branch *Markets.Stocks.NASDAQ.SUNW* to be notified of events to SUNW stock.

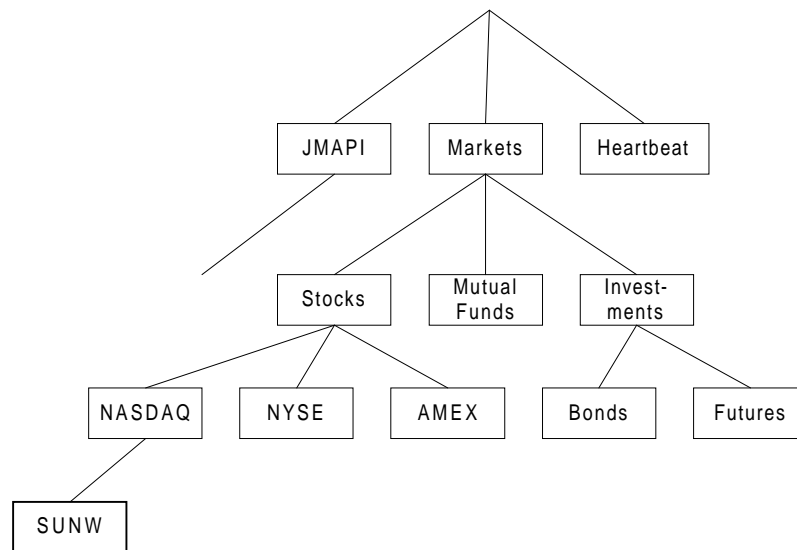


Figure 5-1 Event example

WBEM do not support events in the current version. This is probably a Beta problem which will be solved in later releases.

JMAPI supports events, which give the developers possibilities to build a hierarchical management environment.

5.4 Data Modeling

JMAPI and WBEM have different approaches to data modeling. WBEM uses the CIM 1.1 model (see section 4.7) which results in a tight data model. JMAPI use a special model, MO, which is looser coupled than CIM. The WBEM approach makes it easier to integrate different managed object with each other and also makes it easier to understand other's work.

In the future both WBEM and JMAPI probably will support CIM 2.0.

5.5 User Interface

JMAPI and WBEM have different approaches to the graphical user interface (GUI). JMAPI uses the GUI defined in Admin View Module (AVM). WBEM on the other hand does not have any GUI definitions.

5.6 Technology

JMAPI and WBEM use different technologies. JMAPI is pure Java and strives to live after the buzzwords, "Write once, run everywhere". This means that a Java program can be executed on a number of platforms and operating systems. To be able to use JMAPI the developer needs knowledge in the following areas:

- **Java** – JMAPI uses Java as the programming language
- **JMAPI structure** – it is necessary to understand how different parts in JMAPI interact with each other.

WBEM on the other hand uses Microsoft technologies, and the only fully supported platforms are Windows 95 and Windows NT. To be able to use the WBEM the developer has to have some basic knowledge in:

- **CIM** – WBEM is based on the Common Information Model.
- **WBEM structure** – To understand how the different parts in WBEM interact with each other it is really important to have a good insight in WBEM.
- **Java** - The applets in the Internet Browser are programmed in Java.

- **C++** - Providers are programmed in C++.
- **COM/DCOM** – Much of the programming in the APIs and the communication between different parts in WBEM is based on COM/DCOM.
- **Active-X** – The manager user interface in WBEM is based on Active-X.
- **WBEM API structure** – WBEM has a special structure which is important to understand.

Table 5-1 shows the differences and similarities between JMAPI and WBEM.

Table 5-1 JMAPI versus WBEM

	JMAPI	WBEM
Protocol	SNMP, RMI, and other TCP/IP protocols with socket programming.	SNMP, HMMS ,and any other ones with provider programming.
Database	JDBC	CIMOM
Events	Yes	No, not yet
Data Modeling	MO	CIM 1.1
Platforms	Any that supports Java	NT or Win95
Knowledge entrance requirements	Java and JMAPI structure	CIM, WBEM structure, Java, C++, COM/DCOM, Active-X and WBEM API structure.
User Interface	AVM	Not defined

5.7 The implementation of NemaPro

The goal of this master thesis project was to build a prototype based on one of the technologies, but we did not do very well. In the WBEM environment we had lot of struggle with the documentation and Microsoft's lack of interest to answer questions about strange behavior in WBEM. JMAPI supported only a small set of databases (Oracle, Sybase and Microsoft SQL Server) and the Axis system department was unable to deliver one of these, and we therefore never had the opportunity to try JMAPI.

6 Conclusions

In this section we draw some conclusions of our work and give some guidelines for future work.

Simple Network Management Protocol

We believe that SNMP will continue to be the protocol for managing network devices in the future. It is the only protocol that the market has accepted. Almost all the peripheral vendors provide SNMP agents in their devices. The development of SNMP is still in progress and the level of activity in the SNMPv3 working group mailing list is very high.

Desktop Management Interface

A lot of vendors support DMI, but even though DMTF has modeled standard MIFs for printers (<http://www.dmtf.org/tech/apps.html>) most of the implementations today are found on desktop systems such as Windows, Macintosh, and UNIX workstations. We do not think DMI will be a major protocol for managing devices like Axis servers. The other technologies that we recommend in this section are more interesting in the future.

Web Based Enterprise Management

Microsoft is shipping WBEM agents in Windows 98 and NT 5.0, (Microsoft [980101]). This guarantees that the development of WBEM will continue and there will be a market supporting it. Axis management tools¹ are in some parts easy to port to WBEM since they are written in C++ and use the Windows programming framework. WBEM and Microsoft are in line with the Axis Small Office Home Office (SOHO) direction. WBEM uses the CIM as its model for management. The most important actors in the systems management area support CIM (DMTF [980303]).

Java Management Application Programming Interface

The success for JMAPI relies on the success for Java, and the results of the ongoing lawsuits against Microsoft². The Java hype is helping JMAPI and other Java technology based projects to be successful. The level of activity on the Internet forum for JMAPI developers, the *jmapi-interest* mailing list, is high compared to the WBEM Internet forum, the *wbem-discussion* mailing list. This shows at least that there are several developers outside Sun developing JMAPI systems. According to JavaSoft developers on the *jmapi-interest* mailing list JMAPI will use CIM as its data model and major parts of JMAPI will be included in future Java Development Kits (JDK). One big advantage of JMAPI is that it is platform independent, even though the Beta version we studied is only supported for Sun Solaris and NT.

General Conclusions

It is important to note that neither the WBEM nor the JMAPI activities have yet produced any standardized architecture, protocols, or even documents, and that both approaches still have numerous issues to resolve. Hence, in terms of their

¹ Netpilot and Winpoint are the management tools that Axis ships with the print servers respectively CD and Jaz storage servers (Storpoint).

² There are several lawsuits against Microsoft. The two most important concerned the Microsoft implementation of Java (see <http://java.sun.com/aboutJava/info/index.html> and <http://www.microsoft.com/corpinfo/java/java.htm>) and the Department of Justice's action against Microsoft (see <http://law.miningco.com/library/weekly/bljackson.htm>).

applicability for managing networks it is an open race between WBEM, JMAPI and other commercial management platforms.

During our work with this thesis we have discovered that the specifications change very fast. An example is the scheduled dates for the next release:

- WBEM Q2-98
- JMAPI Q2-98
- CIM Q1-98.

The most important trend is the development of the Common Information Model, mostly due to the massive support from the big actors and especially that Microsoft uses it to model Win32 platforms in the WBEM agents provided with Windows 98 and Windows NT 5.0. Another interesting aspect is that several vendors such as Hewlett-Packard and IBM have not decided which technology to support; they will probably support the winning team. Our feeling is that this is more a political decision rather than a technical.

Suggestions to Axis

As mentioned above WBEM is a fast way to migrate existing applications to a modern environment. The fact that Microsoft will use WBEM in their future management applications and provide WBEM agents in Windows 98 and Windows NT 5.0 (Microsoft [980101]) points toward implementing something WBEM compliant in Axis products.

But if Axis wants to build their own management application we suggest using JMAPI. Sun's system provides a rich flora of user interface classes, the database is independent from the implementation, and an independent JDBC compliant database can be used. Further more, JMAPI is popular within the Java community which can be of great help in the implementation phase. Axis puts a lot of work into making their products to function in heterogeneous environment, JMAPI is a good opportunity to make the management tools work with different platforms as well.

It is important to remember that neither JMAPI nor WBEM are stable enough to be used to implement a real product. Further more, when the final releases of these products are available the technical specifications probably will have changed a lot.

A The Internet drafts

This summarizes Stalling [96] section A5.

An organization known, as the Internet Architecture Board (IAB) is responsible for the development and publication of Internet standards, which are published in a series of documents called Request for Comments (RFCs).

The IAB is the coordinating committee for Internet design, engineering, and management. Areas covered include the operation of the Internet itself and the standardization of protocols used by end systems on the Internet for interoperability. The IAB has two principal subsidiary task forces:

- Internet Engineering Task Force (IETF)
- Internet Research Task Force (IRTF)

Working groups carries out the actual work of these task forces. Membership in a working group is voluntary; any interested party may participate.

The IETF is responsible for publishing the RFCs. The RFCs are the working notes of the Internet research and development.

The final decision of which RFCs become Internet standards is made by the IAB, on the recommendation of the IETF. To become a standard, a specification must meet the following criteria:

- It must be stable and well understood.
- It must be technically competent.
- It must have multiple, independent, and interoperable implementations with operational experience.
- It must enjoy significant public support.
- It must be recognisably useful in some or all parts of the Internet.

Figure A-1 illustrates the series of steps, called the *standard track*, that a specification goes through to become a standard. The steps involve increasing amounts of scrutiny and testing. At each step, the IETF must make a recommendation for advancement of the protocol, and the IAB must ratify it.

The white boxes in the diagram represent temporary states, which should be occupied for the minimum practical time. However, a document must remain a proposed standard for at least four months to allow time for review and comment. The gray boxes represent long-term states that may be occupied for years.

A protocol or other specification that is not considered ready for standardization may be published as an experimental RFC. After further work, the specification may be resubmitted. If the specification is generally stable, has resolved known design choices, is believed to be well understood, has received significant community review, and appears to enjoy enough community interest to be considered valuable, then the RFC will be designated a proposed standard.

For a specification to be advanced to draft standard status there must be at least two independent and interoperable implementations from which adequate operational experience has been obtained. After such experience has been obtained, a specification may be elevated to a standard. At this point, the specification is assigned as a STD number as well as an RFC number. Finally, when a protocol becomes obsolete, it is assigned to the historic state.

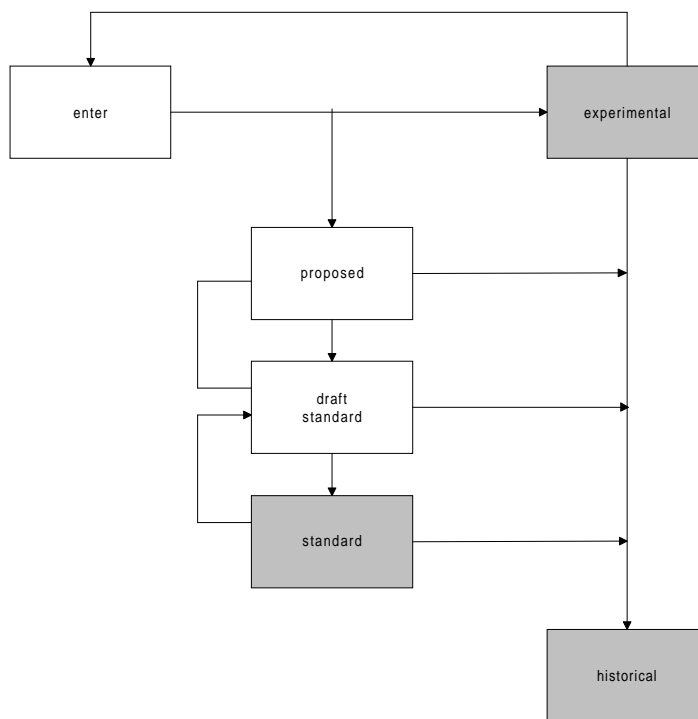


Figure A-1 Standard Track Diagram

B The OSI Model

This section comes from Ersmarker-Nillson [980205].

The OSI model was developed by the International Organization for Standardization as a model for computer communications architecture, and as a framework for developing protocol standards. Table B-1 briefly defines the functions performed at each layer. The intent of the OSI model is that protocols be developed to perform the functions of each layer.

Table B-1 The OSI Layers

1. Physical	Concerned with transmission of unstructured bit stream over physical medium; deals with the mechanical, electrical, functional, and procedural characteristics to access the physical medium
2. Data link	Provides for the reliable transfer of information across the physical link; sends blocks of data (frames) with the necessary synchronization, error control, and flow control
3. Network	Provides upper layers with independence from the data transmission and switching technologies used to connect systems; responsible for establishing, maintaining, and terminating connections
4. Transport	Provides reliable, transparent transfer of data between end-points; provides end-to-end error recovery and flow control
5. Session	Provides the control structure for communication between applications; establishes, manages, and terminates connections (sessions) between co-operating applications
6. Presentation	Provides independence to the application process from differences in data representation (syntax)
7. Application	Provides access to the OSI environment for users and also provides distributed information services

Layer 7 Application Layer
Layer 6 Presentation Layer
Layer 5 Session Layer
Layer 4 Transport Layer
Layer 3 Network Layer
Layer 2 Data Link Layer
Layer 1 Physical Link

Figure B-1 Open Systems Interconnection model

C TCP/IP compared to OSI

This appendix shows a brief architectural comparison between OSI and TCP/IP. For further information see Steven [9603].

Application	Process
Presentation	
Session	
Transport	Host-to-host
Network	Internet
Data link	Network access
Physical	

Figure C-1 OSI compared to TCP/IP.

D Organizations

The different organizations mentioned in this thesis are presented in this appendix. The presentations are taken from respective home pages.

D.1 Object Management Group

(this section comes from www.omg.org)

Object Management Group (OMG) was founded in May 1989 by eight companies: 3Com Corporation, American Airlines, Canon, Inc., Data General, Hewlett-Packard, Philips Telecommunications N.V., Sun Microsystems and Unisys Corporation. In October 1989, OMG began independent operations as a non-profit corporation. Through the OMG's commitment to developing technically excellent, commercially viable and vendor independent specifications for the software industry, the consortium now includes over 800 members. As OMG moves forward in establishing CORBA as the "Middleware that's Everywhere" through its worldwide standard specifications: CORBA/IIOP, Object Services, Internet Facilities and Domain Interface specifications. OMG is headquartered in Framingham, Massachusetts, USA, with international marketing partners in the UK, Germany, Japan, India and Australia.

OMG was formed to create a component-based software marketplace by hastening the introduction of standardized object software. The organization's charter includes the establishment of industry guidelines and detailed object management specifications to provide a common framework for application development. Conformance to these specifications will make it possible to develop a heterogeneous computing environment across all major hardware platforms and operating systems. Implementations of OMG specifications can be found on over 50 operating systems across the world today. OMG's series of specifications detail the necessary standard interfaces for Distributed Object Computing. Its widely popular Internet protocol IIOP (Internet Inter-ORB Protocol) is being used as the infrastructure for technology companies like Netscape, Oracle, Sun, IBM and hundreds of others. These specifications are used worldwide to develop and deploy distributed applications for Manufacturing, Finance, Telecoms, Electronic Commerce, Realtime systems and Health Care.

OMG defines object management as software development that models the real world through representation of "objects." These objects are the encapsulation of the attributes, relationships and methods of software identifiable program components. A key benefit of an object-oriented system is its ability to expand in functionality by extending existing components and adding new objects to the system. Object management results in faster application development, easier maintenance, enormous scalability and reusable software.

The acceptance and use of object-oriented software is widespread and growing. Virtually every major provider and user of computer systems in the world is either using or planning to implement object-oriented tools and applications. Within the next three to five years, revenue from the sale of object-oriented software is projected to exceed three billion dollars. Find out more via <http://www.omg.org>.

D.2 Internet Engineering Task Force

(This section comes from www.ietf.org)

The Internet Engineering Task Force (IETF) is a loosely self-organized group of people who make technical and other contributions to the engineering and

evolution of the Internet and its technologies. It is the principal body engaged in the development of new Internet standard specifications. Its mission includes:

- Identifying, and proposing solutions to, pressing operational and technical problems in the Internet;
- Specifying the development or usage of protocols and the near-term architecture to solve such technical problems for the Internet;
- Making recommendations to the Internet Engineering Steering Group (IESG) regarding the standardisation of protocols and protocol usage in the Internet;
- Facilitating technology transfer from the Internet Research Task Force (IRTF) to the wider Internet community; and
- Providing a forum for the exchange of information within the Internet community between vendors, users, researchers, agency contractors and network managers.

The IETF meeting is not a conference, although there are technical presentations. The IETF is not a traditional standards organization, although many specifications are produced that become standards. The IETF is made up of volunteers who meet three times a year to fulfill the IETF mission.

There is no membership in the IETF. Anyone may register for and attend any meeting. The closest thing there is to being an IETF member is being on the IETF or working group mailing lists (see <http://www.ietf.org/tao.html>). This is where the best information about current IETF activities and focus can be found.

D.3 Desktop Management Task Force

(this section comes from www.dmtf.org)

The Desktop Management Task Force (DMTF) is the industry consortium chartered with development, support and maintenance of management standards for PC systems and products, including DMI (Desktop Management Interface), the most-widely used management standard today. Founded in 1992 by a group of PC industry leaders, the DMTF brings together more than 200 key technology and support-industry providers to create the tools and infrastructure for enabling a more cost-effective, less crisis-driven approach to PC management.

Led by Compaq, Dell, Digital, Hewlett-Packard, IBM, Intel, Microsoft, NEC, Novell, SCO, SunSoft, and Symantec, the DMTF adheres to a collaborative, working-committee approach that involves DMTF members in all aspects of specification development and refinement. Focused on meeting the management needs of today with solutions that are flexible enough to meet the needs of tomorrow, the DMTF also works closely with related industry organizations such as The Open Group and the NMF (Network Management Forum) to ensure that DMTF solutions thoroughly address customer issues and overall management needs.

Easing the Management Burden, Reducing Costs

The goal of the DMTF is to ease the burden of PC management, consequently reducing direct and indirect costs associated with staffing support and productivity losses. To achieve this, the DMTF is chartered with establishing a standard for describing and accessing key support and management information about PC systems and components. The group also promotes the adoption of the standard throughout the industry and works within the industry to address non-technology issues - such as tracking cost of ownership - which are part of the broader PC management challenge.

D.4 The Open Group

(this section comes from www.opengroup.org)

THE OPEN GROUP is a vendor-neutral, international consortium of more than 200 members, including leaders in government, academia, worldwide finance, health care, commerce and telecommunications, which have combined IT budgets in excess of \$55 billion annually.

As the industry's flagship specifications organization for more than a decade, it provides a vendor-neutral forum for buyers and suppliers of technology to build the critical linkages between the \$3 trillion worth of heritage systems currently installed and the new operating environments and applications emerging today, including the Internet and World Wide Web.

THE OPEN GROUP's breakthrough IT DialTone™ initiative will ensure that the Internet remains open by collecting a core set of specifications, products and technologies to create a common level of global security and reliability for the Internet.

To that end, THE OPEN GROUP membership is working with standards bodies and its members to address the areas of core information exchange, transaction processing, application services, location services and security.

THE OPEN GROUP's charter, which provides for research and development of non-proprietary technologies and specifications, enables collaborative development among leading high technology companies worldwide.

THE OPEN GROUP's unique testing and branding process guarantees compatibility of products developed against THE OPEN GROUP specifications.

Formed in February 1996 by merging X/Open Company Ltd. and the Open Software Foundation (OSF), THE OPEN GROUP's nine sponsors are Digital Equipment Corporation, Hewlett-Packard Company, Siemens-Nixdorf Informationssysteme AG, Fujitsu Limited, Hitachi Limited, International Business Machines Corporation, NCR Corporation, Novell, Inc. and Sun Microsystems, Inc.

Headquartered in Cambridge, MA, THE OPEN GROUP can be reached on the World Wide Web at www.opengroup.org.

E UML Notation

(This appendix is taken from DMTF [980303])

The CIM meta-schema notation is based directly on the notation used in Unified Modeling Language (UML). There are distinct symbols for all of the major constructs in the schema, with the exception of qualifiers (as opposed to properties, which are directly represented in the diagrams).

In UML, a class is represented by a rectangle. The class name either stands alone in the rectangle or is in the uppermost segment of the rectangle. If present, the segment below the segment containing the name contains the properties of the class. If present, a third region indicates the presence of methods.

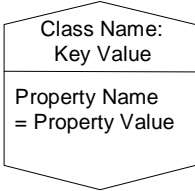
A line decorated with a triangle indicates an inheritance relationship; the lower rectangle represents a subtype of the upper rectangle. The triangle points to the superclass.

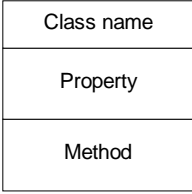
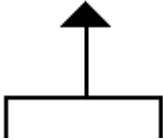
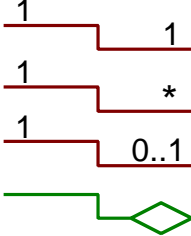
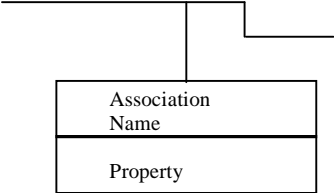
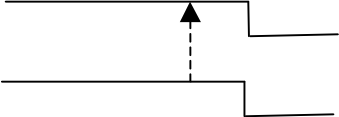
Other solid lines represent relationships. The cardinality of the references on either side of the relationship is indicated by a decoration on either end. The following character combinations are commonly used:

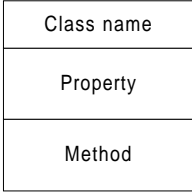
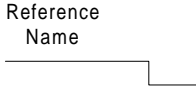
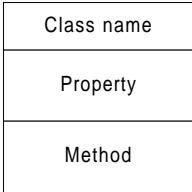
- “1” indicates a single-valued, required reference
- “0..1” indicates an optional single-valued reference
- “*” indicates an optional many-valued reference (as does “0..*”)
 - “1..*” indicates a required many-valued reference

A line connected to a rectangle by a dotted line represents a subclass relationship between two associations.

The diagramming notation and its interpretation are summarized in this table:

META ELEMENT	INTERPRETATION	DIAGRAMMING NOTATION
Object		
Primitive type	Text to the right of the colon in the center portion of the class icon	

META ELEMENT	INTERPRETATION	DIAGRAMMING NOTATION
Class		
Subclass		
Association	<p>1:1</p> <p>1:Many</p> <p>1:zero or 1</p> <p>Aggregation</p>	
Association with properties	<p>Link-class with the link-class having the same name as the association, and using normal conventions for representing properties and methods.</p>	
Association with subclass	<p>A dashed line running from the sub association to the super class.</p>	

META ELEMENT	INTERPRETATION	DIAGRAMMING NOTATION
Property	Middle section of the class icon is a list of the properties of the class.	
Reference	One end of the association line labeled with the name of the reference.	
Method	Lower section of the class icon is a list of the methods of the class.	
Overriding	No direct equivalent. Note: Use of the same name does not imply overriding.	
Indication	Message trace diagram in which vertical bars represent objects and horizontal lines represent messages.	
Trigger	State transition diagrams.	
Qualifier	No direct equivalent.	

F Remote Procedure Call

(this section comes from Jacobsson[9803])

F.1 Overview

RPC (Remote Procedure Call) is a technique for managing connections between client and server in a way that is familiar to most programmers. The idea of the RPC model is to make the complex interaction between the client and server across the network look like regular procedure calls. To the programmer, a remote procedure call looks just like a local one, but in reality it is executed on a server connected to the client by a network. A side effect is, of course, that executing a remote call is much slower than executing a local one.

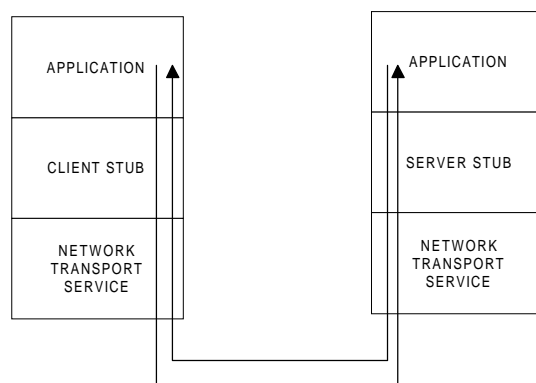


Figure F-1 Remote Procedure Call

The client and the server are two separate processes possibly running on different hosts. In the client program, all calls to a remote procedure are first sent to a dummy procedure called *the client stub*. The stub takes the procedure's in-parameters and packages them in a network message (this is called marshaling) and forwards it to the server. When the message reaches the server it first goes to a server stub which unpacks the parameters and sends them to the implementation of the remote procedure. The return values from the procedure are then sent back to the client in the same way. A problem appears when you want to make RPCs that send parameters by reference and not only by value (naturally, the server can't access memory allocated by the client). Microsoft's RPC uses one possible solution. It lets you send a pointer as a parameter. The marshaling is done of the data the pointer is pointing to, and when the modified data is returned to the client stub it knows where to "put it back" in the client's memory¹. Although this technique solves some problems it still isn't as flexible as using real memory. For instance, it would not be possible to use with a linked list in an efficient way.

The stubs can either be created by the programmer, or generated automatically. All RPCs feature some kind of compiler that generates the stubs from the definitions of the server functions. The definitions are typically done in an interface, written in a special *Interface Definition Language (IDL)*.

Most versions of RPC are synchronous – the client that makes the RPC is blocked and can do nothing until the reply from the server has arrived. A common way to make asynchronous RPC is to use threads.

Different implementations of RPC may offer different *call semantics*. Depending on what reliability the transport service offers (connection-oriented or datagram, for instance) you can't always be sure that the call arrives as it should. It is possible that a call is lost on the way, or delayed for so long that the

client makes a faulty re-transmission of the call. The semantics establish what you can expect from an RPC call. Existing semantics are:

- **Exactly once** – The call is executed once. No more and no less. This semantic is the most difficult to implement and requires lots of checking, and is not supported by either of the RPCs studied.
- **At most once** – The call is either executed zero or one time. This is done by giving every call a sequence number and not accepting duplicate numbers.
- **At least once** – The call is executed, but it's not stated how many times. This is done by retranslating the call until you receive a confirmation that it has been completed.

Currently there are two major standards for RPC – Sun's ONC RPC, and OSF's DCE RPC. These two are incompatible with each other, but protocols are compatible across platforms – for instance, two programs using different kinds of DCE implementations can make calls to each other.

F.2 ONC RPC

This RPC used to be known as SUN RPC, but is now called ONC RPC. ONC stands for Open Network Computing. ONC RPC is a very well spread RPC for UNIX systemsⁱⁱ, and also available for Windows.

ONC PC gives no guarantees whatsoever about reliability in the specification. It relies completely on the underlying protocol. For instance, if TCP/IP is being used at-most-once-semantic are guaranteed, but not if you use UDP. It is left to programmer to deal with the problems that can occur if a message is lost, or a server crashes (in practice there are available libraries that take care these problems).

There are also simply security features in ONC RPC. It has built-in authentication, and offers the option for developers to extend it with more powerful methods.

ONC RPC is described in RFC 1831ⁱⁱⁱ.

F.3 DCE RPC

The DCE RPC is part of the Distributed Computing Environment (DCE), a complete middleware environment from the Open Group (who used to be called Open Software Foundation – hence the "OSF"). DCE provides all technologies needed to build a complete distributed system; RPC, Threads, Naming-, Time-, Distributed file- and Security Services.

The DCE RPC is more powerful and robust than the ONC RPC. Instead of relying on the transport protocol, like ONC RPC does, it has a RPC protocol that guarantees at-most-once-semantics. Another powerful feature is that the DCE RPC is integrated with the other parts of DCE, like Naming- and Security services. This makes it possible to use authentication and encryption, to dynamically locate servers at run-time, and to use threads to make the RPCs asynchronous.

The DCE Security Service is based on MIT's Kerberos^{iv} that offers both authentication and encryption. Release 1.2.2 of DCE supports Kerberos v5^v.

DCE is currently evolving to become more object-oriented, but it is still not a complete distributed object system. Earlier it only supported C, but version 1.2.2 also supports C++. The IDL now supports object-oriented ideas like inheritance^{vi}.

Note that the complete DCE is a commercial product, although you can find public domain versions of just RPC^{vii}.

F.4 MS RPC

The Microsoft RPC is a RPC included with Windows NT and Windows 95. It is supported in The Visual Studio development environment, and can be used for

applications written in C and C++. It is DCS compatible, but it only follows the protocol for the DCE remote procedure call, and does not include any of the other services offered by the DCE environment. They are replaced by Microsoft's own services instead.

MS RPC includes two levels of security services. Included is run-time library for authentication, and you can also use Windows NT Transport Security that works on a lower level.

ⁱ Microsoft RPC Programmer's Guide and Reference, Microsoft Corp. 1997, Supplied on CD-ROM with Visual Studio 97.

ⁱⁱ J. Barkley, NISTIR 5277 - Comparing Remote Procedure Calls, System and Software Technology Division, October 1993,
<http://nemo.ncsl.nist.gov/nistir/5277/>, January 25, 1998

ⁱⁱⁱ R. Srinivasan, Request For Comments: 1831 – RPC: Remote Procedure Call Protocol Specification Version 2 Network Working Group, September 1993,
<http://www.ludat.lth.se/text/rfc/rfc1510.txt>, February 4, 1998

^{iv} J. Kohl, C. Neuman Request For Comments:1510 – The Kerberos Network Authentication Service (V5), Network Working Group, September 1993,
<http://www.ludat.lth.se/text/rfc/rfc1510.txt>, January 25, 1998

^v A. Umar, Object-Oriented Client/server Internet Environments, ISBN 0-13-375544-4, Prentice Hall PTR, 1997

^{vi} OSF DCE 1.2.2 New Features, The Open Group,1996,
<http://www.camb.opengroup.org/dce/info/papers/tog-dce-ds-1296.htm>, January 25, 1998

^{vii} DCE Frequently Asked Questions,
<http://www.opengroup.org/tch/dce/info/faq-mauney.html>, January 25, 1998

G Abbrivations

API	Application Programming Interface
AVM	Admin View Module
AWT	Abstract Window Toolkit
CIM	Common Information Model
CIMOM	CIM Object Manager
CMIP	Common Management Information Protocol
COM	Common Object Model
CS	Computer System
CORBA	Common Object Request Broker Architecture
DCE	Distributed Computer Environment
DCOM	Distributed COM
DME	Distributed Management Environment
DMI	Desktop Management Interface
DMTF	Desktop Management Task Force
DNS	Domain Name Service
GUI	Graphical User Interface
HMMP	Hyper Media Management Protocol
HMMS	Hyper Media Management Schema
HTTP	Hyper Text Transfer Protocol
ICMP	Internet Control Message Protocol
IDL	Interface Definition Language
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol version 6
IPX/SPX	Internetwork/Sequent Packet Exchange
IRTF	Internet Research Task Force
JDBC	Java DataBase Connectivity
JMAPI	Java Management Application Programmers Interface
LAN	Local Area Network
LMS	License Management Service
MIB	Management Information Base
MIDL	Microsoft IDL
MOF	Managed Object Format
MVS	Multiple Virtual Systems
ODBC	Open DataBase Connectivity
OMG	Object Management Group
OMT	Object Modeling Technique
ORPC	Object RPC
OSF	Open Software Foundation
OSI	Open Systems Interconnection
PDF	Portable Document Format
PDU	Protocol Data Units
RMI	Remote Method Invocation
RMON	Remote Network Monitoring
RPC	Remote Procedure Call
SNA	Systems Network Architecture
SNMP	Simple Network Management Protocol
SOHO	Small Office Home Office
TCP	Transmission Control Protocol
TMN	Telecommunications Management Network
UML	Unified Modeling Language
WAN	Wide Area Network
WBEM	Web Based Enterprise Management

H References

- Byte [9704] Byte, 9704, John Montgomery, "For CORBA and DCOM it's time to get practical", <http://www.byte.com/art/9704/sec8/art1.htm>
- Byte [9708] Byte, 9704, Dick Pountain and John Montgomery, "Web Components", p 56-68
- Byte [9710] Byte, 9710, Nancy Nicolaisen, "Embedded diagnostic hardware and new standards simplify the monitoring of system components.", <http://www.byte.com/art/9710/sec6/art13.htm>
- DMTF [960326] Desktop Management Task Force, Desktop Management Interface Specification ver. 2.00
<http://www.dmtf.org/tech/specs.html>
- DMTF [971222:1] Desktop Management Task Force, Common Information Model Common Schema Core Version 0.1 (Draft) (<ftp.dmtf.org/cim>)
- DMTF[971222:2] Desktop Management Task Force, Common Information for Application Management (<ftp.dmtf.org/cim>)
- DMTF[971226] Desktop Management Task Force, Common Information for Application Management (<ftp.dmtf.org/cim>)
- DMTF[980303] Desktop Management Task Force, Common Information Model ver. 2.0 (<ftp.dmtf.org/cim>)
- Heilbronner-Wies[9708] Heilbronner, Stephen and Wies, Stephen. Managing PC Networks. IEEE Communication Magazine, October 1997.
- IBM [9604] IBM Corporation, "SystemView Agent for Win32 User's Guide", Second Edition (April 1996)
- Jacobsson[9803] Per Jacobsson, Comparing Middleware for Client/Server Systems Integrated with the World Wide Web, Department of Computer Science, Lund University, March 1998
- JMAPI[9705] Sun Microsystems, Inc. Java Management Programmer's Guide. http://www.javasoft.com:81/products/JavaManagement/documents/prog_guide/prog_guide.pdf
- JMAPI[971013] Sun Microsystems, Inc. Java Management API, <http://www.javasoft.com:81/products/JavaManagement/index.html>
- Lendenmann [9701] Rolf Lendenmann, Jennifer Nelsson, Carlos Patino Lara, Janet Selby, "Understanding Tivoli's TME 3.0 and TME 10", IBM international support organization
- Microsoft [971031] Web Based Enterprise Management Software Development Toolkit Ver 2.0 Beta

- Microsoft [9801] documentation.
(<http://wbem.freerange.com>)
Microsoft Corporation, Network Working Group, Internet Draft. Distributed Component Object Model Protocol – DCOM/1.0, January 1998
<http://www.microsoft.com/oledev/olecom/draft-brown-dcom-v1-spec-02.txt>
- Microsoft [980101] Windows Management Instrumentation Overview
(http://www.microsoft.com/management/wmi_field_briefing_v1_0.htm)
- Rogerson [97] Rogerson, Dale, “Inside COM, Microsoft’s Component Object Model”, Microsoft Press, 1997
- Seemann[97] Seemann, Michael. SNMP-Simple Network Management Protocol, Nätvärlden nummer 4, maj 1997.
- SNMPv2[960306] SNMP research, Backgrounder on the SNMPv2 Standardization Process
(<http://www.snmp.com/pressrel/v2-background.html>)
- Stallings[93] Stallings, William, “SNMP, SNMPv2, and CMIP” Don Mills: Addison-Wesley, 1993.
- Stallings[96] Stallings, William. “SNMP, SNMPv2, and RMON, Practical Network Management”, second edition. Don Mills: Addison-Wesley, 1996.
- Stevens [9603] Stevens W. Richard, “TCP/IP illustrated Volume 1, The Protocols”, March 1996.
- The Open Group Interface Definition Language, DCE|RPC, The Open Group
- Tanenbaum [96] Andrew S. Tanenbaum, “Computer Networks, #rd Edition”, Prentice-Hall, 1996
- Tyler Tyler. A Guide to SNMP and CMIP.
(<http://www.inforamp.net/~kjvallil/t/snmp.html>)
- WBEM [971022] “The Web-Based Enterprise Management Initiative” Overview
<http://wbem.freerange.com>
- www.axis.com [1] Axis new Print Server is tailor-made for PC Networks
(http://www.axis.com/press/uk/axis_150.htm)
accessed 980305

I Index

A

Admin View Module, 24
Agent Object Factory, 26
Appliance, 23, 26
Application management, 33
AVM, 24
AVM Base, 24
AVM Help, 25
AVM Integration, 25

C

CIM, 30
CIM Object Manage, 21
CIMOM, 21
Cluster, 33
CMIP, 9
COM, 27, 29
Common Information Model, 30
Common Management Information Protocol, 9
Common model, 31
Common Object Model, 29
Common Object Request Broker Architecture, 27
CORBA, 8, 27
Core model, 31

D

DCE RPC, 51
DCOM, 29
Desktop Management Interface, 16
Desktop Management Task Force, 45
Distributed COM, 29
Distributed Common Object Model, 27
DME, 7
DMI, 16
DMTF, 45

E

Events, 34
Extension schemas, 31

H

HMMP, 21
HTTP access layer, 22
HTTP server, 25
Hyper Media Management Protocol, 21

I

IDL, 31
IESG, 45
IETF, 44
Interface Definition Language, 31
Interface Specification Language, 28

Internet Engineering Steering Group, 45
Internet Engineering Task Force, 44
Internet Research Task Force, 45
IP, 6
IPX, 6
IRTF, 45

J

Java Code, 26
Java DataBase Connectivity, 22
Java Management API, 22
Java Management Application Programming Interface, 22
JDBC, 22
JMAPI, 22
JMAPI Elements, 23

K

Kerberos, 51

L

LME, 7

M

Managed Data Interfaces, 25
Managed Object Format, 31
Managed Object Interface, 25
Managed Object Interfaces, 25
Managed Object Server, 25
Management Information Base, 12
Management Information Format, 17
MIB-II, 13
Microsoft Interface Definition Language, 29
MIDL, 29
MOF, 31
MOS, 25

N

Native Methods, 26
NetBEUI, 6

O

Object Management Group, 44
Object Manager Group, 28
Object Pinging, 29
Object RPC, 29
OMG, 28, 44
OMG IDL, 28
ONC RPC, 51
Open Network Computing, 51
Open Software Foundation, 46
ORPC, 29
OSF, 7, 46
OSI, 8

P

PDU, 10

R

Remote Method Invocation, 27
Remote Network Monitoring, 15
Remote Procedure Call, 50
Remote Procedure Calls, 27
Results, 34
RMI, 27
RMON, 15
RPC, 27, 50

S

Simple Network Management Protocol, 10
SNA, 6
SNMP, 10
SNMPv2, 14
SNMPv3, 15
SPX, 6

SUN RPC, 51
Systems management, 33

T

TCP, 6
THE OPEN GROUP, 46
TMN, 8
trap, 11
Trap-directed Polling, 11

U

UML, 31
Unified Modeling Language, 31
Unitary Computer System, 33

W

WBEM, 8
Web Based Enterprise Management, 19
VMS, 6