Java Dynamic Management™ Kit White Paper

Dynamic Management for the Service Age

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
1 (800) 786.7638
1.512.434.1511

Contents

Java Dynamic Management™ Kit

This White Paper describes some of the technology behind Java Dynamic Management™ Kit: Sun Microsystems' new paradigm for dynamic management.

Java Dynamic Management Kit is the foundation for building and distributing dynamic network management intelligence into applications, networks, and devices.

Java Dynamic Management Kit technology provides a unique set of features that radically changes the way companies solve their real-life management problems. By leveraging existing standards like SNMP, and combining them with new technologies based on Web and Java™ technology, it goes beyond the limitations of current management systems to effectively address the new requirements of the service age.

A New Paradigm

Corporations and service providers in all industry segments are facing new challenges in management of service-driven environments. In these environments, managed resources (which can be applications, devices, services or network policies) appear, move and disappear across the network, as they are created, installed, activated and used at a pace never experienced before.

Such challenges can hardly be met by current, second generation management technologies. These are static in nature and can therefore only work with resources that are known in advance by all the elements of the management system. A static management system is incapable of managing today's new services since they need to be added faster than the system's change mechanisms can be updated.

Java technology provides a dynamic alternative, but we would gain little from rewriting existing static management systems in the Java programming language.

Further Than Standards

Sticking to standards is essential. But it doesn't give you any competitive advantage in itself. The next step forward in network management will come, not from improved standards, but from management technology that, while being compatible with standards, is flexible enough to manage anything, and to cope instantly with change without much human intervention. Such technology will store continually management evolving intelligence within the network itself. It will incorporate standard technologies, and build upon them.

By leveraging existing management standards like SNMP and combining them with new technologies based on the Web and on the Java platform, Java Dynamic Management Kit allows you to step toward the future now. Using Java Dynamic Management Kit, you can maintain compatibility with existing network infrastructures while you evolve beyond the limitations of current management systems towards new management solutions that are better adapted to .com infrastructures.

Java Technology Standard for Management

Java Dynamic Management Kit implements the public specification for Java Management Extensions. This defines how to manage Java technology-enabled resources, as well as defining a core set of services for managing these objects, and the means to interface with existing management solutions. Applications, services or devices that follow the specification are instantly manageable through Java Dynamic Management Kit agents.

Java technology has much to offer to the developers of system, network and service management solutions. Java technology is dynamic, flexible, and portable: these unique features make it an ideal foundation for building third-generation management solutions. Such solutions are dynamic in nature and can therefore meet the requirements of service driven management over heterogeneous networks and platforms.

Java Dynamic Management Kit: Universal Agent Toolkit

Java Dynamic Management Kit is a Java technology-based solution for building and distributing management intelligence into network devices. It allows rapid development of autonomous Java agents for system, network, configuration and service management.

Java Dynamic Management Kit makes management dynamic, opening the door to new types of lightweight, flexible management applications which can be created, deployed, enhanced or deleted in real time. By design, Java Dynamic Management Kit is suitable for adapting legacy systems, implementing new management solutions and preparing those of the future, while remaining open to applications across all industries.

Managed JavaBeans™ Components

The Java Dynamic Management Kit toolkit includes a library of core management services, in the form of Java technology-based components, called MBeans (Managed Beans). These are mini applications or parts of applications, which can be used as they are or combined with each other, to create management services for agents.

MBeans instrument the resources to be managed. Resources can be devices, computation time, network control, user applications, or virtually any object you want to handle through a management application.

MBeans follow certain design patterns for exposing attributes and operations so that any Java Dynamic Management Kit agent can recognize and manage them. Because the design patterns for MBeans are so simple, existing Java technology-based resources and services can become manageable at little cost. Even legacy code can become manageable through a suitable MBeans interface.

Java Dynamic Management Kit MBean technology gives:

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

Autonomous, Java Technology-Based Agents

Java Dynamic Management Kit allows management services to be incorporated directly into agents, so that they can perform management tasks autonomously. This allows management intelligence to be distributed throughout the network, rather than being concentrated in manager devices. It also means that agents can actually perform management tasks themselves, rather than just sending information to higher level managers.

The benefits include:

- Reduced network management traffic
- Fewer alarms escalated to administrators
- Quicker response to events
- Reduced administration costs

The management framework in an agent, called the MBean server, is the link between a manager and the managed resources. The MBean server dispatches management requests to MBeans and forwards events back to managers. MBeans can slot in and out of the MBean server, just like hardware elements in a rack.

The MBean server is fully compatible with the PersonalJava™ platform, as are most of the management services. This means that Java Dynamic Management applications can connect to a wide range of consumer devices and future products.

Dynamic Push/Pull Service Distribution

Software distribution is also integrated into Java Dynamic Management Kit, enabling rapid propagation of new services or software over the network.

Management services, in the form of MBeans, can be dynamically distributed to the network. They can be slotted in and out of the Java Dynamic Management Kit agent's MBean server, to add, change or delete services, just as hardware components are slotted in and out of a rack.

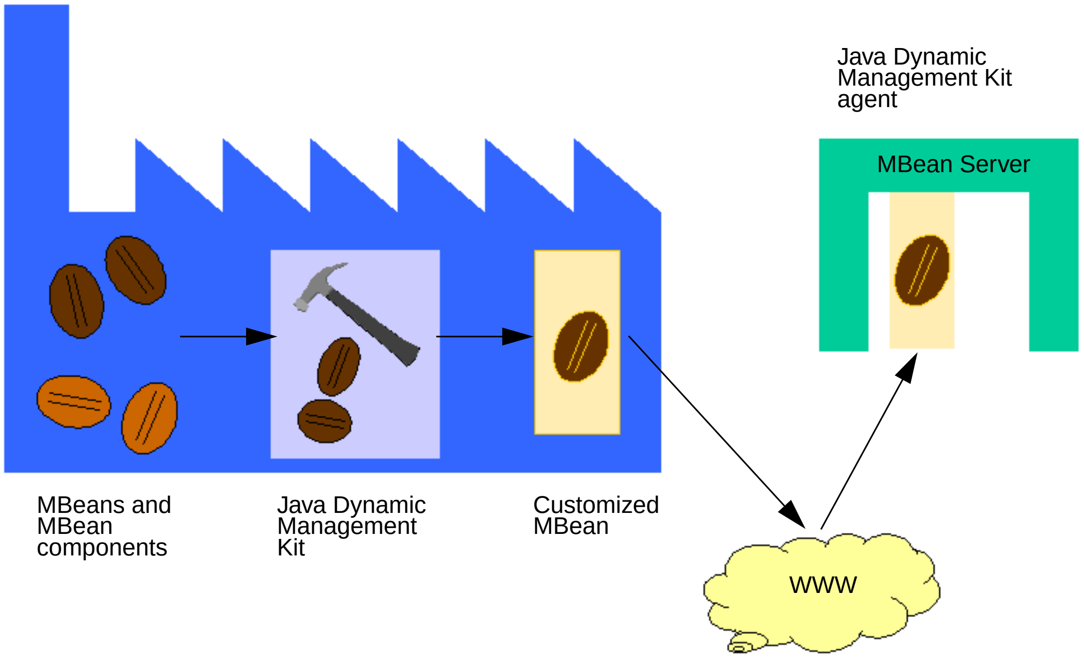
Java Dynamic Management Kit agents can download management services from the management web server when they access it, for example, at boot time. So they can pull new management services as soon as these become available. Updated management services can also be pushed to the agents. This allows new management services to be implemented and started at any time.

The net result is that with Java Dynamic Management Kit, it is no longer necessary to know in advance what will need to be configured, monitored and managed in the future.

The Java Dynamic Management Kit Development Process

To develop an agent using Java Dynamic Management Kit:

- You select MBean components that implement generic management services, from those included in the Toolkit, and customize them in order to generate MBeans for the services you require. Or you create your own MBeans, adapted to your environment and instrumenting your own resources.


Java Dynamic Management Kit agent
MBean Server
MBeans and MBean components
Java Dynamic Management Kit
Customized MBean
WWW

- You can then either:
  - create a complete agent incorporating the management services, install it on a device and initialize it
  - or you can decide to create an empty agent framework which you fill with management services on a real-time basis, using network distribution.
- If you have chosen the second packaging option, then you upload management services in the form of MBeans, to the agent over the intranet/Internet, using the push/pull software distribution mechanisms included in Java Dynamic Management Kit. The new or updated services slot into the agent's MBean server and begin to function instantly.

The Resulting Java Dynamic Management Kit Agent

The resulting agent is composed of the following elements, running in a Java Virtual Machine:

- MBean server: this is a software backplane into which the services defined by the MBeans can be plugged.
- Generic management services in the form of MBeans.
- MBeans created by the developer implementing the specific management services for the device.
- Device specific access method created by the developer, using native function calls (based on the Java Native Interface specification).

Java Dynamic Management Kit Components

Java Dynamic Management Kit provides the following components:

- A dynamic management architecture, which provides a set of Web-based mechanisms for automatic propagation of management services across the network to the agents.
- An agent framework, which includes a library of reusable core agent services in the form of MBeans. These include: object repository, dynamic class loading, dynamic native library loading, relationship, basic notification, filtering and monitoring.
- Protocol adaptors, implemented as JavaBeans components, which provide transparent communication services to the MBeans. The available protocols are RMI, HTTP and SNMP.
- Service creation tools, which include:
  - A proxy generator, which helps developers to create their own specific Java management applications, by generating the remote client front end for a given MBean.
  - A Java SNMP MIB Compiler. This takes an SNMP MIB as input, and outputs a MBean which enables the Java Dynamic Management Kit agent to be managed by an SNMP manager.
- A set of APIs.

Benefits of Java Dynamic Management Kit for Developers

The benefits Java Dynamic Management Kit provides for agent developers include the following:

- Java Dynamic Management Kit provides a *universal Java agent toolkit,* which provides the tools and services needed by developers working on agents for network, system, application, or service management.
- Java Dynamic Management Kit tools and services enable *very rapid agent development,* through the use and recombination of Java components which provide basic management services such as filtering, event notification, and so on.
- By providing generic management services, Java Dynamic Management Kit *frees developers* to concentrate on developing value-added device or application-specific services rather than on (re)creating generic services.
- *JavaBeans applications can be managed by a Java Dynamic Management Kit agent directly,* without needing any instrumentation configuration files, CGI or HTTP support.
- The use of MBeans to develop management services *facilitates code reuse*, limits the code required to implement new services, and makes maintaining agents easier.
- Use of Java technology makes the development platform independent of the target platform, since *the resulting agent can be implemented on any platform* which supports a Java Virtual Machine.
- Use of protocol adaptors enables agents, services and resource instrumentation to be developed without knowledge of the protocol used for communicating with the manager.

Key Features of Java Dynamic Management Kit

The key features of Java Dynamic Management Kit include the following.

Autonomous Agents

Using Java Dynamic Management Kit to build a certain amount of intelligence directly into your agents allows them to perform management tasks directly, without the intervention of a manager device.

For example, you have a large installation of thousands of networked desktop devices and you want to monitor their disk space. Using traditional agent technology, when the device disk is 80% full, the device agent sends an alarm to the manager, which pages the network administrator, who performs actions by hand on a management console.

This kind of technology implies management traffic on the network (sending alarms, receiving action commands), and involves manual intervention by a network administrator over a management console.

Java Dynamic Management Kit agents eliminate all of that by acting autonomously. A Java Dynamic Management Kit agent in the same situation could actually go and look on the device disk and perform housekeeping operations, such as deleting all .back files more than 6 weeks old.

Agent to Agent Communication

As well as being able to take action autonomously, a Java Dynamic Management Kit agent can communicate with another Java Dynamic Management Kit agent, in order to solve the issue without management intervention.

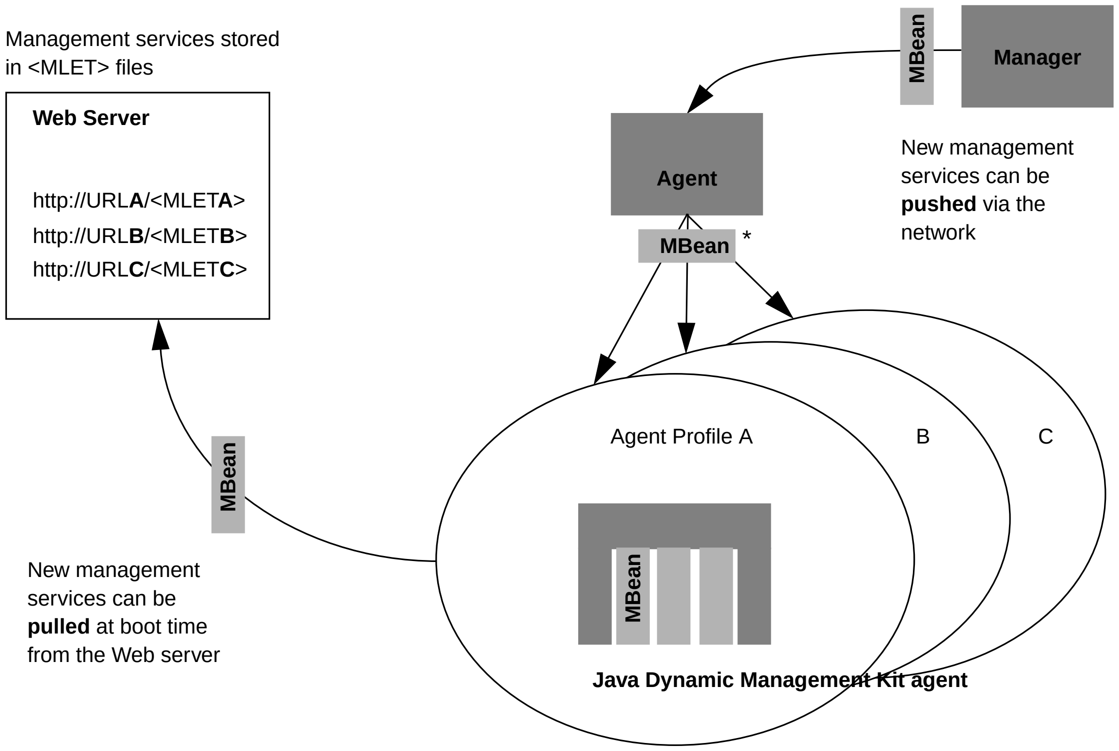
For example, the Java Dynamic Management Kit agent could communicate with the agent on the local file server, and back up PC files to the server before deleting them on the PC. This means the management happens horizontally, rather than using the old vertical, hierarchical model.

This type of structure reduces the management traffic load on the network, means that low level problems get dealt with without generation of alarm conditions, makes for quicker responses to critical conditions, and starts the evolution to zero administration networks which manage themselves without human intervention.

For example, in the case of a cluster configuration providing a high availability solution, agent to agent communication removes the need for a centralized manager as a single point of contact in case of failure. When any node of the cluster has a problem, the agent communicates directly with the agents in the other nodes, rather than to a centralized manager, which would then have to route the message back down to the other nodes of the cluster.

Push / Pull Technology

The graphic below shows how Java technology pushes management intelligence using Web technology methods.



Management services stored in <MLET> files
Web Server
http://URLA/<MLETA>
http://URLB/<MLETB>
http://URLC/<MLETC>
MBean
Manager
Agent
MBean *
New management services can be pushed via the network
Agent Profile A
B
C
MBean
MBean
New management services can be pulled at boot time from the Web server
Java Dynamic Management Kit agent

The management services are built as MBeans. These are compressed in a .JAR file which can be downloaded (either pushed or pulled) over the network.

The agent may boot from a specified URL at connection with the management server. The html page corresponding to the URL contains one or several <MLET> (management applet) tags. Each <MLET> tag refers to one or several .JAR files. It also includes extra code to specify details of the management services concerned, and may define icons, sounds etc.

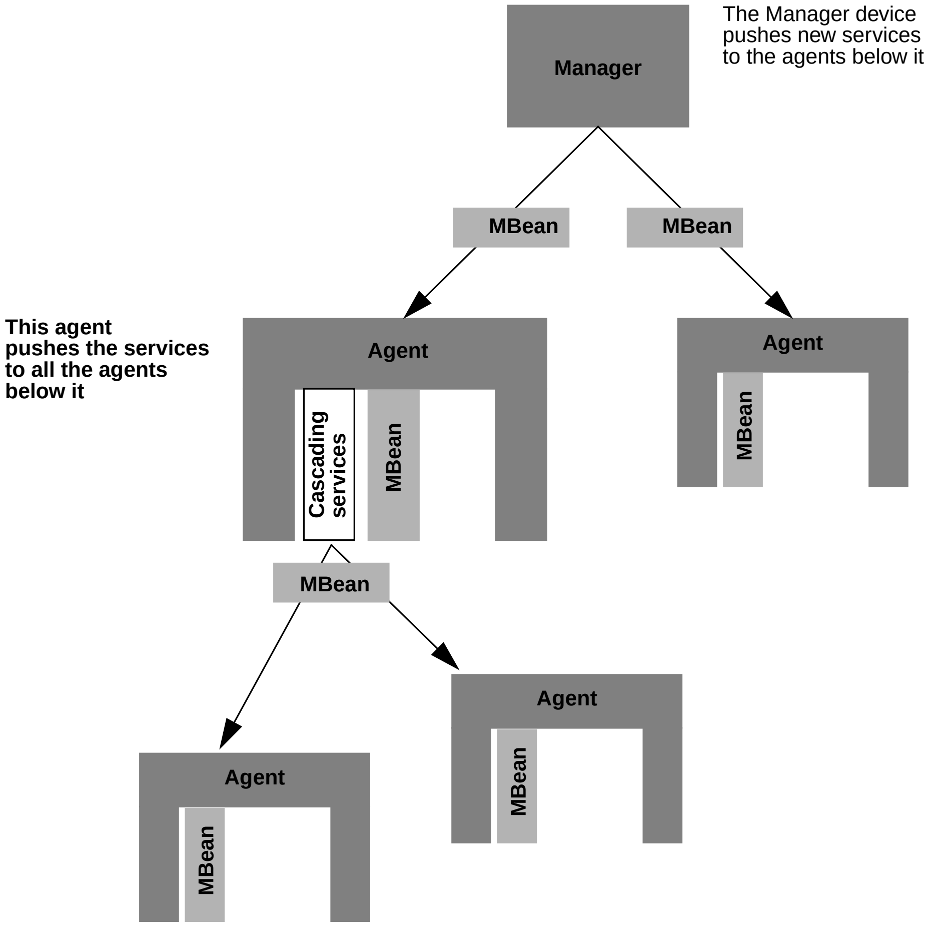
The .JAR files get pushed to the agent and work straight away, in exactly the same way as an animated Javatechnology-based applet for a web page.

Different clients can be made to boot from different URLs, which specify different agent profiles (different <MLET>s). This allows administration of large installations of heterogeneous devices with minimum effort, since the different agents automatically receive the management services appropriate to them, when they boot from the server.

Changes to the management agents get downloaded automatically, as soon as they are implemented. All the agents are running the right version, since they can be updated if necessary whenever they boot to the server, or in response to a request from a management application.

Cascading

Cascading services make it easier to distribute management services in very large installations, with minimal administration effort and cost. Using cascading, agents distribute services to the agents below them on the hierarchy. Java Dynamic Management Kit includes a library of cascading services.


The Manager device pushes new services to the agents below it
Manager
MBean
MBean
This agent pushes the services to all the agents below it
Agent
Cascading services
MBean
Agent
MBean
MBean
Agent
MBean
Agent
MBean

Cascading Services

Cascading services allow you to establish cascading hierarchies in which each certain agents manage a group of lower level agents.

Cascading services make distribution of software or services easier to manage, by enabling certain agents to push management services to the agents below them. In the example shown here, the Manager pushes Java agent services such as Event Filtering and Logging to certain agents, in which cascading services are installed. These agents then push this intelligence further down the agent hierarchy, to the agents below.

Integration with Existing Management Solutions

Java Dynamic Management Kit's additional management protocol APIs put the breakthrough technology of Java Dynamic Management Kit within the reach of any existing management solution.

Java Dynamic Management Kit technology does not attempt to replace existing legacy systems. This is simply not an option for end-users who must consider the value of the investment they have made. Introducing new technology beside an existing management system is also not a viable option, as this would create two separate management entities unable to cooperate or communicate effectively.

Instead, Java Dynamic Management Kit offers a vertical integration, providing manager and agent services through the technologies already in place. It thus provides a means for the seamless introduction of the latest Java technologies into existing management systems.

Java Dynamic Management Kit includes an open interface that any management system vendor can leverage. Using this interface, a Java Dynamic Management Kit agent and its resources can present management information consistent with various management models, such as:

- SNMP
- CIM / WBEM
- CORBA
- TMN
- LDAP

The Java Dynamic Management Kit specification includes the definition of several management protocol APIs which allow Java Dynamic Management Kit managers to access agents in a legacy system and communicate with them through an existing protocol. For example, the SNMP manager API provides the services needed to write applications that manage SNMP agents or act as SNMP proxies. The definition of a WBEM client API allows you to write Java applications that access a CIM Object Manager.

SNMP Agents

You can use Java Dynamic Management Kit to develop dynamic SNMP agents. Using the Java Dynamic Management Kit MIB Compiler you can easily develop an SNMP agent, based on your own specific MIB. The resulting SNMP agent can initially be deployed like a traditional SNMP agent, but can be updated easily using the flexibility of Java Dynamic Management Kit.

When a Java Dynamic Management Kit agent is managed by an SNMP manager, its Java Dynamic Management Kit-specific capacities remain opaque to the SNMP manager. However, new applications can be developed easily using Java Dynamic Management Kit, to integrate the two worlds.

For example, when the Java Dynamic Management Kit agent's icon shows an alarm condition on the SNMP manager screen, because it has received an SNMP trap, you could click on the icon to open a Java application which would allow you to download a specific diagnostic module to the remote device. Such an application would enable you to benefit from the capacities of Java Dynamic Management Kit even within the more limited SNMP environment.

Scalability

Java Dynamic Management Kit agents are scalable. They can be deployed in any device that can run a Java Virtual Machine. These can range from a mobile phone to a high end server.

Furthermore, Java Dynamic Management Kit agents are *dynamically* scalable; they can load and unload services as required, which means that their footprints are never bigger than they need to be, and can be adjusted to new requirements instantly.

Java agents range in size from 200 Kilobytes, depending on the number of services they support.

Seamless Jini™ Connection

Jini™ connectivity technology provides an infrastructure for federating devices, delivering automatic discovery of services.

Java Dynamic Management Kit acts as a bridge between Jini technology-based communities and advanced management applications. Once Jini technology-enabled devices are plugged into the network and discovered, Java Dynamic Management Kit applications can then access or control the resource. This is called Sun Spontaneous Management™ technology.

Sun Spontaneous Management technology enables service providers to deliver both user-specific customization and universal access through heterogeneous networks. This is just one example of how the Java Dynamic Management Kit’s universal architecture can integrate new technologies as they develop, link them to legacy systems, and as a result deliver services faster and more efficiently to the customer.

Benefits of Java Dynamic Management Kit-Enabled Solutions

- Java Dynamic Management Kit enables “intelligent” management services to be integrated inside devices. Java Dynamic Management Kit “intelligent” agents can take pro-active actions without human intervention or administration console (pro-active actions can be based on device-dependent events or other agent behavior).
- Java Dynamic Management Kit reduces management traffic on the network, by enabling direct agent to agent communication. The conventional approach allows no horizontal communication, even where it makes sense. Instead, events get percolated up the network to manager applications, and actions trickle down. Java Dynamic Management Kit's agent to agent communication enables actions to be taken without intervention from a manager.
- Java Dynamic Management Kit agents can be remotely enhanced, upgraded or updated with new intelligent management services via Internet at any time. Intelligent management services can be distributed on a large scale with zero administration, either automatically received, or pushed. For example, this technology allows the upgrade of large installations of network devices, such as desktop PCs, with minimal human intervention or management operations.
- The Java Dynamic Management Kit dynamically extensible object model provides richer information content than that provided by traditional agents. SNMP MIBs are static. If you want to enhance them, you have to build an extension at development time. Afterwards it's too late; deployment of an upgrade at once the SNMP agents are in place would be too costly. But Java lets a management application look at whatever it needs to look at (CPU, memory, disk space...). It also allows you to change what you're looking at from one day to the next, extending or changing your management applications as you need, in real time.
- Java Dynamic Management Kit enables devices to be managed via Web Browsers, Java applications or existing SNMP managers.
- The scalability of Java agents makes them available for management of even the smallest devices. Furthermore, the fact that they are *dynamically* scalable means that their footprints are never bigger than they need to be, while they can always be extended as necessary.

- Java Dynamic Management Kit implements the public specification of the Java Management Extensions (JMX), which define the standard for managing Java technology-enabled resources.

Java Dynamic Management Kit Agents: Areas of Use

Since Java Dynamic Management Kit is a universal toolkit, it can be used to develop agents for a multitude of different functions.

Webtop Management

Java Dynamic Management Kit push technology, combined with a system of profiling, enables large heterogeneous installations of PCs, Unix systems, NetPCs and webtop devices to be managed with minimum human intervention. Each device contains a Java Dynamic Management Kit agent, which corresponds to a particular profile, and which downloads appropriate services accordingly.

Desktop management becomes easy, since the management application and the management services in the agents can be changed at any time. New services are simply pushed to the agents at boot time. Software upgrades become trivial. New profiles can be added at any time.

System Management

System management agents can be used to manage an entire installed base of devices, ranging from webphones, webtops and PCs, to high-end machines running the Solaris operating environment, Unix and Windows NT. System management and software distribution, using the push/pull capacities inherent in Java Dynamic Management Kit, becomes an automatic, zero-effort task.

Java Dynamic Management Kit technology takes system management towards the future. A Java Dynamic Management Kit agent can be integrated into any device that supports a Java Virtual Machine, which allows for easy and immediate integration of new devices in the future.

Integration of current SNMP management with the new facilities provided by Java Dynamic Management Kit preserves existing investment while allowing access to the benefits of Java Dynamic Management Kit. Java Dynamic Management Kit agents can monitor SNMP variables, and send traps to an SNMP manager. Their dynamic nature allows them to deploy new SNMP management services as these become available.

Management information concerning Java Dynamic Management Kit agents is accessible via any SNMP manager, Java platform management console or Web browser.

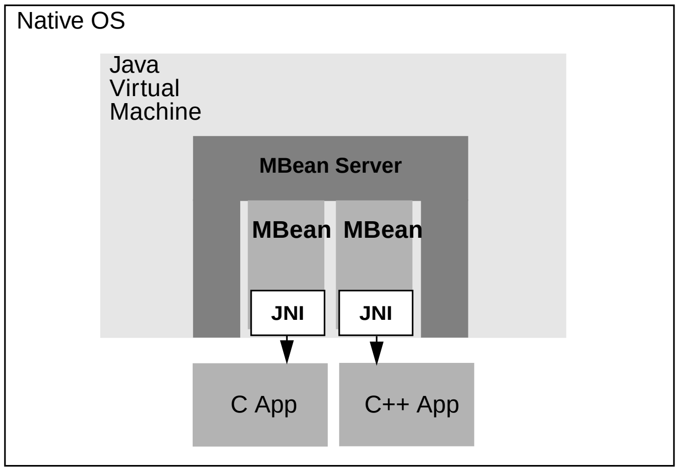
Application Management

Application configuration management agents provide application configuration and management tools. They make it possible to remotely reconfigure an application in real time, over the Web.

By combining the JavaBeans components provided with Java Dynamic Management Kit, with JavaBeans applications you write yourself, you can create new application configuration management agents quickly and easily.

C and C++ Applications

Applications written in C or C++ can be managed by a Java Dynamic Management Kit agent through a native interface (generally developed by the developer of the application). The application's native interface acts as the glue which allows the Java Native Interface to communicate with the application.



C and C++ applications can be managed via native interfaces

Device Management

Device management agents provide management of network devices such as bridges, routers, modems, PC cards, printers, network access servers and so on.

Any device which can run a Java virtual machine can be managed using Java Dynamic Management Kit. Obviously a device with a real time operating system needs to be supplied with an interface allowing the RTOS to interface with the Java Virtual Machine.

A New Paradigm

Java Dynamic Management Kit opens the door to the future. It enables a new type of dynamic management application, and represents a huge opportunity for ISPs and system integrators, making it easy for them to provide services which differentiate them from their competition.