



Operating System

Implementing Directory Enabled Networks Using Windows 2000 Technology

White Paper

Abstract

Directory enabled networks (DEN) integrate directory services with the network infrastructure in ways that can reduce the total cost of ownership of the network. This document provides a framework for building directory enabled networks integrated with Active Directory™, the directory service included in the Microsoft® Windows® 2000 operating system. Network equipment vendors, network management software developers, and service providers can use the information in this white paper when developing network applications specifically designed for a Windows 2000 environment.

© 2000 Microsoft Corporation. All rights reserved.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Microsoft, Active Directory, IntelliMirror, Jscript, Visual Basic, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The example companies, organizations, products, people and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

3/00

Contents

Introduction.....	1
DEN Basics	2
DEN Background	3
DEN Update	3
Directory Enabled Networking Overview	5
Physical Infrastructure Management	5
Inventory and Asset Tracking	5
ACL Management	5
Performance and Fault Management	6
Network Services Management	6
Quality of Service	6
Policy-Based Routing	7
Dial-in Remote Access and Virtual Private Networking	7
IP Address Management	8
Desktop Management	9
Directory Enabled Networking Architecture	10
Functional Components	10
Policy Console	11
Policy Repository	11
Policy Decision Point (Policy Server)	12
Policy Enforcement Point	13
Additional Considerations	14
Policy Transaction Protocols	14
Support for Legacy Devices	14
End Host Participation	14
Dynamic State Information	15
Missing LDAP Features	16
Variations in Architecture Implementation	16
Two-tiered Architecture	16
Three-tiered Architecture	17

Directory Enabled Networking with Active Directory	19
Policy Data Store and Active Directory	19
Policy Console and MMC	20
Policy Decision Point and Group Policy	21
Policy Enforcement Point	23
Programmatic Interfaces for Active Directory	24
Dealing with Replication Latency and Data Consistency	24
Roadmap to Active Directory Enabled Networking	25
Schema Extensions Design	25
Management Console Snap-Ins	26
Policy Templates Design (Optional)	27
Directory Population	27
Policy Enumeration and Evaluation	28
Policy Events/Triggers	29
Dynamic State Information (Optional)	30
Policy Proxy (Optional)	30
Windows Installer Integration	31
Summary	32
For More Information	32

Introduction

Directory enabled networks (DEN) refers to the industry initiative, sponsored by the Distributed Management Task Force (DMTF), to develop a standard information model for representing network elements and services in a directory that both stores the network state and exposes the network information. Vendors can use this information model to build interoperable network applications and services according to a consistent set of policies.

Through such integration of the directory service and the network, the directory service takes on a new role. It not only acts as a repository for information about users and computing resources (such as servers or printers), but also is extended to include information about network devices, services, and applications. More significant, the directory includes information about the relationships among all the elements in the directory. In this expanded view of directory services, users along with computing and network resources use the directory service to publish information about themselves and to discover other resources and obtain information about them. Once information about users, network elements, and services is available in a single location, it is possible to manage the network based on policies.

Directory enabled networks resonate with the enterprise market because they let network managers replace the device-by-device management model with a more holistic approach to managing network resources so they don't have to manually configure various network devices.

Enterprises also look to directory enabled networks to help them deploy quality of service (QoS) across networks to allocate resources, such as bandwidth, to applications. Network administrators use QoS to guarantee that critical applications receive high quality service, without being affected by other resource hungry applications such as streaming media. However, QoS implementation is often unwieldy, requiring manual configuration of hundreds of routers and switches, making the process error-prone and driving up the cost of network administration. Using directory enabled network management, QoS can be deployed from a central management console that creates policies in directories and automatically distributes configurations to network devices, operating systems, and applications. Furthermore, based on policy, critical, time-sensitive, or confidential information can be directed to special routes that are set aside to handle this type of traffic.

Overall, automatic configuration and troubleshooting devices can save expensive network engineering time, allowing IT managers to concentrate on adding greater value with other efforts.

Note: Directory enabled networks is the comprehensive term that includes all the technologies necessary to make directory-based control of networks a reality. The term is often used interchangeably with policy-based network management.

This document provides a framework for building directory enabled networks that integrate with Active Directory™, the directory service included in the

Microsoft® Windows® 2000 operating system. This framework can serve as a basis for network equipment vendors, network management software developers, and service providers to develop components specifically designed for distributed networking in a Windows 2000 environment. The purpose of this document is to provide the reader with the following information:

- Introduce the background and technology behind directory enabled networks.
- Identify the ways that directory enabled networks ease network administration in the areas of infrastructure management and network service provisioning.
- Identify the key components of a policy based networking architecture, how Windows 2000 can be used to enhance these components, and how to provide for any missing functionality.
- Present a roadmap for integrating with Active Directory and creating a directory enabled networking product strategy using the Windows 2000 platform.

This document is not intended to be a tutorial on DEN, the DMTF's Common Information Model (CIM), the Lightweight Directory Access Protocol (LDAP), Active Directory, or any of the related Windows 2000 technology components discussed in this document. Pointers to more detailed documentation are included in this document wherever appropriate.

DEN Basics

DEN refers to the industry initiative, sponsored by to develop standards in the Distributed Management Task Force (DMTF), to develop standards for improving the manageability of networks by using a directory services that apply a consistent set of policies linking users, applications, and network resources.

The DEN specification is part of the DMTF's Common Information Model (CIM) standard for enterprise management. CIM is a detailed information model developed by DMTF that describes the manageable objects, attributes, and relationships in a single system, network of systems, or enterprise. The domains of interest for CIM are system and network management. Management is loosely defined in the CIM context so the scope is very broad, encompassing hardware, operating systems, operations, application installation and configuration, security, identity and many other areas. CIM is expressed in Unified Modeling Language (UML) diagrams and in a formal language devised for the purpose called Managed Object Format (MOF).

The domains of interest for DEN are network device, user, and service/application management. The scope of DEN is narrow compared to that

of CIM, and there is a clear overlap of the CIM and DEN domains. DEN is therefore being developed as an extension to CIM; the DEN information model adds network devices and services to the CIM information model. The DEN model is also represented the same way: diagrammatically in UML and as a formal schema expressed in MOF.

The DEN schema is intended for implementation in directory services that support LDAP v3 as the access protocol. This presents some challenges because the CIM meta-model can express relationships that are difficult (and in some cases impossible) to express in the LDAP information meta-model. Thus, converting a model with a rich meta-model to a model with a more constrained meta-model requires compromises and will have varying degrees of success.

DEN Background

Microsoft and Cisco Systems originally conceived of DEN as a joint specification and development effort. DEN was introduced as an industry initiative at the Microsoft Professional Developers' Conference in October 1997 under the name "Network Use of the Directory Service." The DEN Ad Hoc Working Group (AHWG) was formed at the second DEN review meeting in February 1998 to complete the specification and submit it to the DMTF for development as a part of CIM. The specification was completed in August 1998 and submitted to the DMTF in September 1998.

The goals of the DEN initiative are to produce an information model and schema that describes all of the elements required to perform end-to-end configuration and management of network devices and services. As stated in the DEN specification, its primary goals are: to provide an extensible foundation for building network applications and services; to allow for the provisioning of end-to-end services on a per-user, per-application, and per-service basis; and to increase the integration of network management services.

After information about users, services, devices, and other network elements is available in a central location, it is possible to manage the network based on policies. The DEN specification defines policies as the resources a given consumer (user, application, or service) can use in the context of a given application or service.

DEN Update

The original DEN model submitted to the DMTF was somewhat simpler than the current DEN models because the original DEN plan was to develop a schema suitable for use in an LDAP directory, and the DEN meta-model recognized this. CIM has no such constraints—the goal of the CIM model is to be correct in the abstract. An implementation of DEN must deal with the constraints of an LDAP directory in mapping the CIM model to LDAP.

Development of CIM is ongoing with each successive release adding more elements to the model. The current version of CIM is 2.2 was released in June 1999. Version 2.3 is currently in review by DMTF members. Important additions anticipated in the CIM 2.3 release relate to policies, users, and security with an LDAP schema for the DEN specification.

Directory Enabled Networking Overview

Directory-enabled network management focuses on managing three areas in the IT environment: the physical infrastructure, network services, and the desktop.

Physical Infrastructure Management

In the physical infrastructure, the directory stores object definitions that contain the attributes of routers, switches, hubs, firewalls, remote access devices, and other network components. Using object classes and inheritance properties, these devices are more easily and consistently managed through the directory. Knowledge of the physical network structure can be useful in a variety of ways—asset tracking, access control list (ACL) management, and fault isolation. The first uses knowledge of physical location of devices, the second needs connectivity information, and the third employs both.

Inventory and Asset Tracking

Using a DEN-based directory, IT managers can maintain an inventory of network equipment and configuration. Keeping this network data, along with application and server data, in a centralized store allows for better network modeling and management. On a mundane level, knowing which users are using which network equipment and services within an organization is useful for IT managers. Network administrators would also appreciate the ability to store network hardware configuration information in a common repository rather than maintaining this information for each device separately.

Asset tracking applications need to match records of purchased items with the physical objects themselves. Typically, an asset number is assigned to an asset, recorded in a database of some sort, and a sticker with the number is attached to the actual object. The tracking number provides the link between the asset tracking software and the physical objects.

The schema provides a way to unify asset tracking with network management. The asset tracking number is stored as an attribute of the directory object representing the hardware device. Searches may be performed based on the asset tracking number to locate a particular piece of hardware. Once the hardware asset object has been found in the directory, references to containing objects may be followed. For example, a hierarchy may start at the chassis that contains the object, and go to the rack containing the chassis, to the closet where the rack is located, and so forth on up through floor to building to campus.

ACL Management

Router access control lists are used for a variety of purposes, including firewall applications where packets with spoofed IP addresses are rejected. Spoofing is a common type of attack where the source address of a packet originating outside an enterprise is forged to make it appear to originate from inside the

enterprise. You can prevent spoofing by putting an entry in the router ACL that connects the enterprise to the Internet so that packets with internal source addresses are rejected. (It is impossible for a valid packet arriving from the Internet to have an internal source address.)

It is sometimes desirable to perform spoof prevention among internal routers. When physical connection information is available in the directory, it is a simple matter to see what is connected to each router interface as it is being configured. This can help ensure that the correct subnets appear in the interface's ACL. The directory also contains other relevant information, such as which routers must use route authentication (to guarantee that updating their configuration is not malicious), and policies that govern the distribution and use of ACLs.

Performance and Fault Management

When communication faults occur, following the path of the physical connections between devices helps identify the point of failure. At various points it may be necessary to find the physical location of one of the devices or the endpoints of a physical connection so that physical inspection can occur or diagnostics can be run. In these cases, the user can change the focus from connectivity to location information, in order to identify the location as described in the asset tracking example above. This illustrates how the connectivity and location information can be used together to solve complex problems.

Network Services Management

Managing network services through directory enabled networking allows administrators to assume a network-centric view of services and provision them on an end-to-end basis as opposed to managing individual network elements. Several examples of DEN based network service management are discussed below.

Quality of Service

The driving factors behind deploying QoS in the network are as follows:

- Mission critical applications need QoS to ensure delivery that won't be impacted when misbehaving or bandwidth-intensive applications are using the network.
- Internet service providers charge for guaranteeing a minimum bandwidth (during periods when the network is congested).
- Real-time traffic generated by multimedia applications (like video-on-demand) or packetized voice (like VOIP) needs QoS to guarantee latency, jitter, and a specified level of packet loss.

QoS can prioritize application traffic and allocate resources across the network to

ensure the delivery of mission critical applications when the network is congested. Any mission critical applications are given the right of way, with e-mail and file transfers receiving the lowest priority. Prioritizing network traffic can be implemented with a number of mechanisms (such as priority queuing); the particular mechanism depends on the specific policy that is used to implement QoS, which is reflected in the appropriate policy object in the directory. A policy can also include per-user settings—such as the number of flows allowed, peak rate bandwidth that can be requested, allowable bandwidth reservations depending on time of day—that allow for delivery of personalized QoS service. In order to manage these QoS policies, an enterprise needs a solution that is integrated with their existing LDAP-based directory service. This allows the network administrator to edit the user's QoS attributes with the same user interface as they use to edit other user attributes, and eliminates the need to maintain multiple stores of user information.

Policy-Based Routing

Another use of QoS is in implementing policy-based routing. This can be used to select alternative paths based on the attributes of the packet, providing administrative control over routing. Again, this needs centralized administration and management, which in turn needs the directory, in order to be implemented. Policy-based routing has several benefits, among them:

- Provides the ability to differentiate on a user, application, operation, or process basis and direct traffic to higher-performance lines than other traffic.
- Permits servicing critical, time-sensitive, or confidential information by special routes that are not available to other traffic

Dial-in Remote Access and Virtual Private Networking

Let us assume that Contoso Pharmaceuticals, a fictional company, wishes to offer dial-in access to their domestic sales force, as well as virtual private network (VPN) access for contractors and employees traveling overseas. In order to consistently manage and account for their users, and for the use of their NAS devices and Layer 2 tunnel servers (PPTP/L2F/L2TP), Contoso Pharmaceuticals has chosen to adopt the RADIUS protocol. However, given the large number of employees and contractors that need to be managed, Contoso Pharmaceuticals desires a RADIUS solution that is integrated with their existing LDAP-based directory service. This allows the network administrator to edit the user's RADIUS attributes with the same user interface as they use to edit other user attributes, and eliminates the need to maintain multiple stores of user information.

As part of this service offering, Contoso Pharmaceuticals may wish to implement a number of access policies that need to be centrally managed through the directory. Some of their requirements are as follows:

-
- Sales force personnel require high speed dial-up access; however, access restrictions (in terms of application usage, files that can be accessed, etc.) may vary by location and time of day. The directory needs to manage and coordinate these policies.
 - Hardware resources may become over-subscribed. So, Contoso Pharmaceuticals may wish to restrict the use of ISDN ports to sales personnel only during the hours between 9 A.M. and 5 P.M., and permit the use of multi-link connections.
 - Contractors should only be given access to a selected group of computers with restricted logon privileges. Therefore, Contoso Pharmaceuticals may wish to apply a special filter to contractor traffic.
 - Certain individuals, such as roaming users from the finance department, often need access to highly confidential information over VPNs. Contoso Pharmaceuticals may require that these users authenticate using a smart card and use only 128-bit encryption so as to provide for extended security.
 - For security reasons, Contoso Pharmaceuticals may wish to restrict contractors and finance users to a single logon per session.

In certain cases, Contoso Pharmaceuticals may also wish to implement policies that depend on the type of port or network that the user is connecting to. For example, if the user is connecting using dial-up connection, then it may be appropriate to include tunnel attributes during the logon process to set up a tunnel for the user. However, if the user were already connected by a tunnel, this would not be necessary. Similarly, if Contoso Pharmaceuticals only has a limited number of ISDN ports available, it may be desirable to manipulate the characteristics of the port to accommodate resource contention. Administration of all these requirements can be managed by appropriate policies defined in the directory, which would then be evaluated by the RADIUS servers and the resultant actions enforced by the NAS devices.

IP Address Management

IP address management is a crucial network service for intelligent networks that essentially consists of the following tasks:

- Configuring network nodes with an IP address, subnet mask, nearest-router address, and the preferred DNS server.
- Monitoring usage of addresses to ensure efficient use.
- Configuring DNS servers with each node's name and IP address mapping.
- Automating all these tasks.

The essential components of address management are the DNS service, DHCP service, and the DNS dynamic update protocol that provides the link

between DHCP and DNS. IP address allocation across multiple DHCP servers needs to be coordinated to avoid duplicate address assignments.

Using a directory, DNS and DHCP servers can receive their configurations directly from the directory service. They can access and store both configuration and operational data in the directory for sharing and coordination across multiple servers. Further, to improve reliability, failure of one DHCP server can cause a failover to a second DHCP server that configures itself from the directory.

Directory integration also provides a foundation for policy-based management of network services on a per-user basis. It allows network managers to define policies in terms of users (as opposed to device addresses or IP addresses) and relies on the directory to provide address mapping to match the two types of information.

Desktop Management

Enterprise networks include large numbers of computers on user desktops, often used by roaming users, connected to various server systems, often geographically separated. Network administrators want the power and flexibility of the distributed system, and the simplicity of administration of a centralized system. In addition to the increasing need to lower the cost of ownership for PC networks, there is also an increasing need to lower the cost of administering desktops. Much of the cost of administering the desktop environment is due to user configuration errors, which increase the number of Help desk incidents, as well as wasting employee time. Secondly, application and file deployment is a huge cost to any organization. Administrators need to mandate desktop and application settings and to efficiently install applications, so that users no longer have the need or ability to alter their system configuration. The directory stores workstation configuration, user preferences, application settings, and policies governing application deployment. The Group Policy infrastructure for desktop policy enforcement included in Windows 2000 uses the Active Directory service to bring together policy and desktop management. Integrating with the directory allows for user location independence and centralized management.

Directory Enabled Networking Architecture

Directory enabled networking allows for policy-based management of networks, using directories as the underlying repository of policy information. The systems for policy-based management serve as a link between networks and the business requirements of network users. For example, by setting policies that distinguish between traffic from different users and different applications, network managers can control the allocation of network resources to mission-critical traffic. In general, DEN's role in binding users and applications to network services using information stored in the directory is key to policy-based management of networks.

Using network QoS as an example, today the industry standard for provisioning end-to-end QoS is to configure each network device along the path through a telnet connection into each device. This device-by-device management is a huge barrier to implementation. Policy-based management can significantly reduce the complexity, because a QoS policy is defined once, at a management console, and then automatically distributed across geographically dispersed network devices.

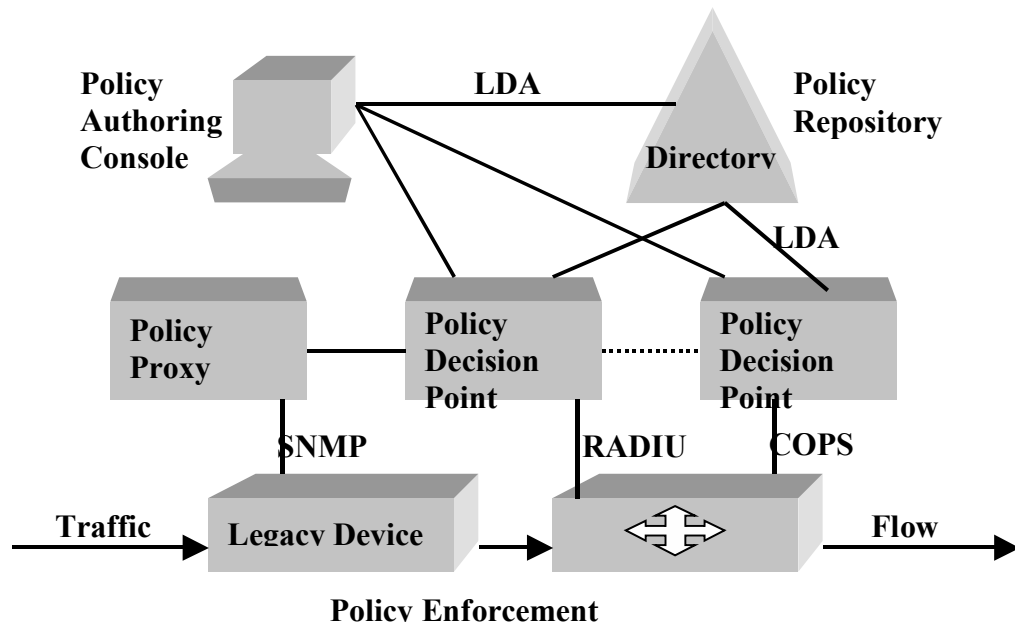
Functional Components

Policy-based management involves the use of administratively prescribed rules that specify actions in response to defined criteria. All implementations of policy-based management architecture must contain four key requirements:

- An interface to define and update policy rules.
- A directory repository to provide storage for the rules.
- Decision point(s) to assess the conditional criteria of rules.
- Enforcement point(s) to execute the actions of a rule when the conditions are met.

Directory enabled networks contain these four components that correspond to the requirements above:

- Policy console used to author and edit policies.
- Policy repository for storing policies and data about network elements.
- Policy decision points (PDPs) for evaluating and disseminating policies.
- Policy enforcement points (PEPs) responsible for enforcing the policies



Policy Console

Policy management consoles provide network administrators with an interface to author and edit policies that are then deposited into the directory using the LDAP protocol. Optionally, the policy console may also communicate with policy servers to notify them of changes made to policies in the directory. With the console, it's mostly a matter of choosing between a graphical or text interface. The policy console provides high-level abstraction of rules or templates to help an administrator create policies. The policies, defined as sets of abstract rules, are network device independent. The user interface presented by the console is driven by the policy schema used to store policy data in the directory. Since the policy console is the first component to which a new or changed policy is presented, it should additionally perform the function of validating policies by checking consistency of the constituent policy rules and verifying that the rules are deterministic before placing them into the policy repository.

Policy Repository

The directory service plays a special role in directory-enabled networking. The directory serves as the central location for storing policies, profiles, user information, network configuration data, and IP infrastructure data such as network addresses and name server information. Policy information includes a deterministic set of policy rules, structured as conditions and actions, for managing resources in the context of a specific policy discipline (for example, remote dial-up access, QoS, security). The policy rules are typically network

device independent. They are produced by the policy management tools (like the policy console below) and consumed by the policy decision points (PDPs). User information typically includes not only user profiles but also authentication and authorization (access rights) data. The format of data stored in the directory is formalized according to schemas that are derived from standard schemas published as part of the DEN specification.

The DEN specification defines LDAP v3 as the core protocol for accessing information from the directory. Hence, the directory used to implement the data store must have built-in support for the LDAP v3 protocol. Furthermore, since a sophisticated policy system will have network components accessing the directory from geographically dispersed locations, the directory must be distributed and highly available in character.

Directory service scalability is another technical issue that gates the deployment of directory-enabled network services. For large enterprises and service providers, directories need to accommodate millions of entries. Thus, a given directory service must scale to handle both a large number of entries and to perform transactions against those entries in a timely fashion. In addition, the accompanying directory replication and synchronization methodology must also be capable of accommodating the scaling requirements of very large customers operating across multiple time zones.

Policy Decision Point (Policy Server)

The policy decision point (PDP) generally takes the form of policy servers that are responsible for gathering all relevant information, making a decision based on the administrator's predefined set of policies, and then communicating that decision to the relevant network node(s) by a policy transaction protocol. For example, the policy server is responsible for telling switches and routers along a traffic path how to handle different types of traffic in order to meet a specified QoS level. Those devices, in turn, use internal queuing mechanisms to enforce policy and mark traffic so that their counterparts on the network do the same.

Policies express business rules. Commonly used policies in distributed systems define how resources in the network are accessed, configured, and used. For example, when a user logs on, policies determine what applications they see on their desktop. When a service (for example, Microsoft SQL Server™) starts on a computer, policies determine how many users can connect to the service. When an employee moves to another group, policies determine which documents and other services the employee can access. Because a single policy can be applied to many users, computers, or services, the administrator manages a small number of policies, rather than a large number of computers.

Policy selection is the mechanism that determines what policies should be applied to a given operation. The number of policies in the distributed system can be large and diverse. For example, there might be a set of policies governing network bandwidth reservations, another set that determines access

to documents and services, a set controlling router configurations, a set for setting up secure Internet connections, and so on. Each of these policies applies to particular subjects and objects, in the context of particular operations. Evaluating all policies for all operations is expensive in terms of time and very inefficient—a given policy usually applies to a fairly narrow set of circumstances and is meaningless outside of that set. For example, user desktop policies never apply to router configurations.

The PDPs access policies from directory-based policy schemas using the LDAP protocol to make policy decisions. The PDP evaluates the policy rules inherent in policy definitions and generates appropriate instructions for PEPs for which it provides the higher-level abstraction translation service. These policy discipline-specific and, perhaps, device-specific instructions may, for example, specify the traffic shaping parameters for a QoS policy or the address filters for a firewall policy.

Conflict detection and resolution is an important aspect of a policy-based management architecture. A policy conflict occurs when two or more actions are expected to execute at the same time but are inconsistent with each other. For example, unless identified with appropriate precedence rules, a policy rule specifying, “all engineers get bronze service” is in conflict with another rule defined as “the lead engineer gets gold service.” The resulting actions may be linked to more than one policy. The conflict may be detected at the time the policy is entered into the policy repository or when the policy decision is made at the PDP. The latter might have to resolve conflicts between multiple policies originating from uncoordinated sources performing policy updates.

PDPs may also perform the additional functions of logging events and monitoring network resource usage and state to determine if the applied policies produce the desired effect.

Policy Enforcement Point

The network node, which may be a router, switch, remote access server, or even an end-host, is the policy enforcement point—that is, the location where the appropriate policies are applied to the network traffic. These devices/nodes have the ultimate control over which traffic is allocated resources.

The network nodes (devices, hosts) serving as PEPs request policy-based decisions from one or more PDPs, cache policy decisions for future use, and enforce policy by processing incoming or outgoing packets according to the policy-based decision applicable to that specific network traffic flow. The PEPs may also relay events back to PDPs for the purpose of network monitoring as well as roll-up of accounting and resource consumption information.

Additional Considerations

It is important to realize that the DEN structure, while necessary for building intelligent networks and delivering policy-based networking, is not sufficient by itself. Other related issues merit consideration and are described below.

Policy Transaction Protocols

In addition to the directory access protocol (LDAP) used to access policy and other data from the directory, other protocols are also needed between PDPs and PEPs. Using such protocols, called policy transaction protocols, a PEP can request policy decisions from a PDP. The same protocols are used to distribute policies from policy servers (PDP) to network devices (PEP) and keep them in synch. Many vendors reuse existing protocols like CLI (Command Line Interface) or SNMP (Simple Network Management Protocol) to do this job. This legacy approach can be problematic. Although SNMP is good for performing simple functions (*get/set*), it is unreliable, not secure, and not fast enough to do bulk data transfers. Using LDAP as the policy transaction protocol also exhibits similar drawbacks because it was designed as a simple query mechanism for accessing X.500 directories. More efficient policy transaction protocols, such as Common Open Policy Service (COPS) and DIAMETER, are under development by the Internet Engineering Task Force (IETF). The COPS protocol was developed specifically for QoS and is tuned for speed.

Network devices being built today vary in their support for policy. Very few existing devices support the protocols that distribute policies. LDAP-enabled devices are starting to emerge from several networking vendors. COPS-capable devices have also been announced.

Support for Legacy Devices

There is a need for mechanisms that support legacy network devices in the policy-based management framework. These mechanisms can permit corporate network managers to add policy to networks built with legacy devices such as aging routers and LAN switches that do not have the memory or processing power to run the newer policy protocols. These legacy devices, however, can be managed through protocols like SNMP.

A policy proxy is an entity that can enable support for legacy network devices. A policy proxy is essentially a policy server that serves as a proxy for legacy devices from the standpoint of PEP and PDP communications to effect policy decisions and distribution of policies through a policy transaction protocol. In essence, the policy proxy translates policy actions handed down from PDPs and communicates them to the legacy devices by CLI or SNMP.

End Host Participation

In order for QoS policies to be applied effectively to network flows through a device like a router or switch, network traffic must be properly classified either

by the end host originating the traffic or by an edge device at the LAN/WAN boundary. Extending the reach of policy to the end nodes (or desktops) enables applications on the end nodes to actively participate in the end-to-end delivery of service by shaping and controlling traffic even before it gets onto the network thus delivering much more granular control over network traffic. Furthermore, classification of application traffic often should be done at the host for two reasons:

- If the end-to-end traffic is encrypted, then the edge device cannot effectively identify and classify it.
- Most policy-enabled switches and routers employ two items to identify traffic: an end-user's IP/media access control address and an application's layer-4 TCP/User Datagram Protocol (UDP) port number. This works fine with applications that have static well-known port numbers such as FTP. But many applications, such as H.323 video, use dynamic port numbers making identification of such traffic at the edge device more difficult if not impossible.

End-to-end delivery of QoS depends on end host participation. Without enabling the applications on the end nodes, policies can only be enforced passively because a network device like a router or switch has to wait until the traffic reaches it in order to deal with it. A similar consideration also holds true for IP security services that rely on the two end nodes participating in the negotiation of appropriate protocol parameter to be used in delivering end-to-end security.

Dynamic State Information

Although the directory will act as the root of a range of information, it may not necessarily be the single repository for this information. In particular, there is a large amount of non-static, or volatile information that must be maintained, including information about the state of network links, the flows active through a router, and the data rate for each flow. This type of information is normally maintained in some kind of real-time data store other than a directory. Such information may need to be accessed by the policy servers to make a decision based on policies that are predicated on dynamic network state in addition to static data stored in the directory. For example, if user Jane in Accounting wants to access R3 on the SAP server, she must reserve 1.5Mbps of bandwidth going through a particular set of switches and routers, which gives her access to a particular server. To deliver consistent levels of service, the network must be capable of maintaining detailed information on the state of Jane's connection and the devices that enable it.

DEN is not designed to change the directory to accommodate transient data needs. Rather, it is designed to serve as the repository of slowly changing data, and it is up to the overall data and information models to integrate the directory

with other data stores such as memory caches, file systems, or real-time databases where necessary.

Missing LDAP Features

The current LDAP standards are not sufficient to support all the requirements of an efficient policy-based network architecture. Two deficiencies have particularly been identified as hindrances:

- **Change notification**—The ability to notify entities accessing data from the directory when data changes and how the data changed.
- **Transactional integrity**—The ability of the directory to ensure that a set of related operations is completed as a set or not at all.

Variations in Architecture Implementation

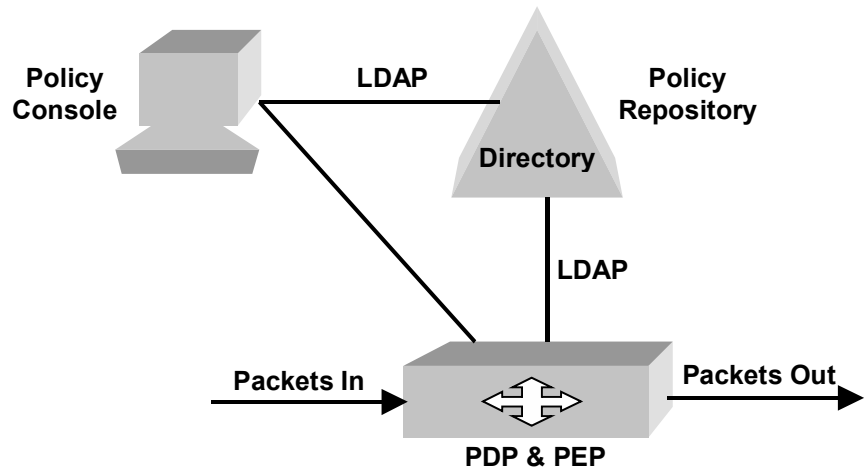
Vendors may differ in how they implement the various components of the policy architecture, in particular the PDP and PEP. There are two options with respect to how the PDP and PEP functions are located relative to each other. First, the policy server (PDP) functionality may be embedded inside a switch or router, hence co-located with the PEP functionality. This results in a two-tiered architecture (Figure below). Second, the PDP may be deployed as a standalone policy server, typically running on a Windows 2000-based server or a Unix-based server, that distributes policy decisions to the switches and routers. This results in a three-tiered policy architecture. Where the policy server is positioned can affect scalability, cost, and performance thus necessitating close attention from network architects.

Two-tiered Architecture

A two-tiered architecture implementation, illustrated below, requires smarter network devices that have an embedded policy server (or agent) communicating directly with the directory service using LDAP. Consequently, these devices also need more memory and processing power, which drives up cost. On the other hand, since the switch/router does not have to go out to ask for policy, there isn't any significant slowdown in the processing of network traffic, possibly resulting in better traffic-handling performance. It should be noted, however, that a substantial number of entities in the form of switches/routers directly interface with the directory for policy, and exert a significant directory load. Furthermore, this structure doesn't provide a global view of the network.

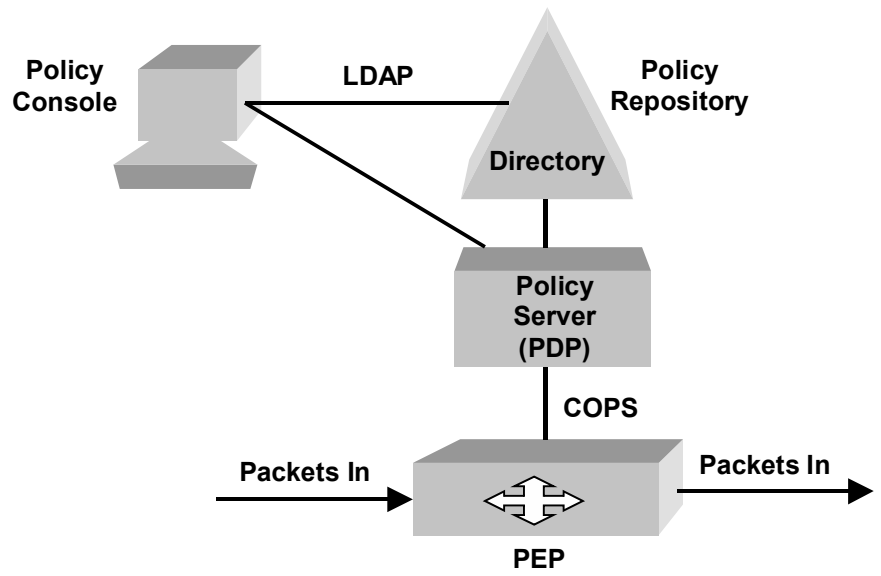
The two-tiered architecture is highly suitable for populating the directory with individual device information for inventory and asset tracking as well as for network topology discovery. Instead of having a separate application process crawling the network to discover devices, the individual devices can each publish pertinent information about themselves and their knowledge of the

network topology into the directory as a prescribed schema at pre-selected junctures (such as right after booting up). This is a more efficient process and less demanding on the network than a crawling discovery method.



Three-tiered Architecture

In a three-tiered architecture implementation, network devices require less intelligence because they are not required to interpret and evaluate policies to make decisions. Instead, a device is handed down a set of policy actions possibly in the form of configuration parameters (like traffic classifiers, queuing parameters, IP filters) through a policy transaction protocol (like COPS) to be enforced at the device. Several network nodes or devices can be coordinated by a single policy server. Since the number of entities interacting with the directory is fewer, the directory load exerted is far less. An illustration of a three-tiered architecture is contained in the figure below.



Directory Enabled Networking with Active Directory

Active Directory is a central component of the Windows 2000 operating system. Active Directory is a scalable directory service, based on Internet-standard technologies, and fully integrated with the operating system. In addition to providing comprehensive directory services to Windows applications, Active Directory is designed to be a consolidation point for isolating, migrating, centrally managing, and reducing the number of directories that companies have. This makes Active Directory the ideal long-term foundation for corporate information sharing and common management of network resources, including applications, network operating systems, and directory-enabled devices.

This section outlines the Active Directory features and technologies that form the essential ingredients of a directory enabled networking strategy using Windows 2000 technology.

Policy Data Store and Active Directory

Active Directory plays many roles, from being the backbone of distributed security in Windows 2000 to providing a framework for publishing network services. It provides a central service for administrators to organize network resources—to manage users, computers, applications, and services; and to secure intranet and Internet network access. In a policy-based networking architecture, Active Directory additionally serves as the policy store where policies are defined and bound to objects or aggregates of objects.

The following features make Active Directory an effective platform for directory enabled network development:

Open Standards Support—Active Directory is built on standards-based protocols wherever possible such as:

- DNS as the locator service.
- LDAPv3 as the query and update protocol.
- Kerberos as the default security protocol for logon and authentication.

Active Directory natively implements the LDAPv3 protocol, schema, and semantics. The support for open standards makes it possible to use a wide variety of applications with Active Directory.

Extensibility—The schema, which contains definitions for every object class that can exist in the directory service, is extensible. This allows administrators and software developers to tailor the directory to their needs. The schema is held in Active Directory just as other objects are and can be manipulated using the same access methods and tools that are used to access other directory objects.

Security—Active Directory provides the infrastructure for a variety of security capabilities. Using mutual authentication, not only do clients authenticate themselves to servers, but clients can also verify the identity of the server (for

example, LDAP clients can verify the authenticity of an Active Directory server) before transferring sensitive data. Using public key security support, users can log on using smart cards instead of passwords. Active Directory also supports central management of access control to network services and resources, by specifying permissions in access control lists.

Simplified and Flexible Administration—Active Directory lets administrators apply object- and attribute-level access control. This lets administrators delegate specific administrative privileges and tasks to individual users and groups to make better use of system administration resources and reduce the number of users who need domain-wide control.

High Availability—Traditional directories with single master replication offer high availability for query operations, but not update operations. Active Directory supports multi-master replication, which offers high availability of both query and update operations.

Scalability—Active Directory uses the Internet's Domain Name System (DNS) as a locator mechanism, and stores information in domains, which are partitions that allow the directory to be distributed over a large network. Active Directory's database technology has been tested for hosting millions of objects in the directory. Experts and Compaq Computer have developed and tested the world's largest LDAP directory on a single server with over 85 million entries in one Active Directory. Their test results show that customers can use Active Directory for both managing their Windows 2000 networks and for their most demanding electronic commerce applications. This combination of DNS locator, partitioning, and scalable storage ensures that the directory will scale gracefully as an organization grows.

Policy Console and MMC

Microsoft Management Console (MMC) is an extensible user interface that hosts administrative tools called snap-ins that are responsible for performing management tasks. MMC makes extensive use of COM technology. Both Microsoft and independent software vendors (ISVs) can extend the console by writing MMC snap-ins. The MMC hosting environment can be used to create policy consoles that serve as policy authoring stations in the policy-based network.

MMC interacts with snap-ins using several MMC-defined interfaces under the COM standard. Snap-ins are implemented as in-process components supporting one or more of these interfaces and register themselves appropriately in the MMC registry area. MMC gains access to your snap-in by obtaining a pointer to one of its interfaces. With this pointer, MMC can use your snap-in without understanding its implementation and can depend on it to behave in a consistent manner. MMC interfaces allow snap-ins to share a common hosting environment and provide for cross-application integration. By using this approach, both Microsoft and ISVs can write administrative

applications that are hosted by MMC. This is also true for developing management tools to run in MMC and writing applications to be managed by MMC administrative tools.

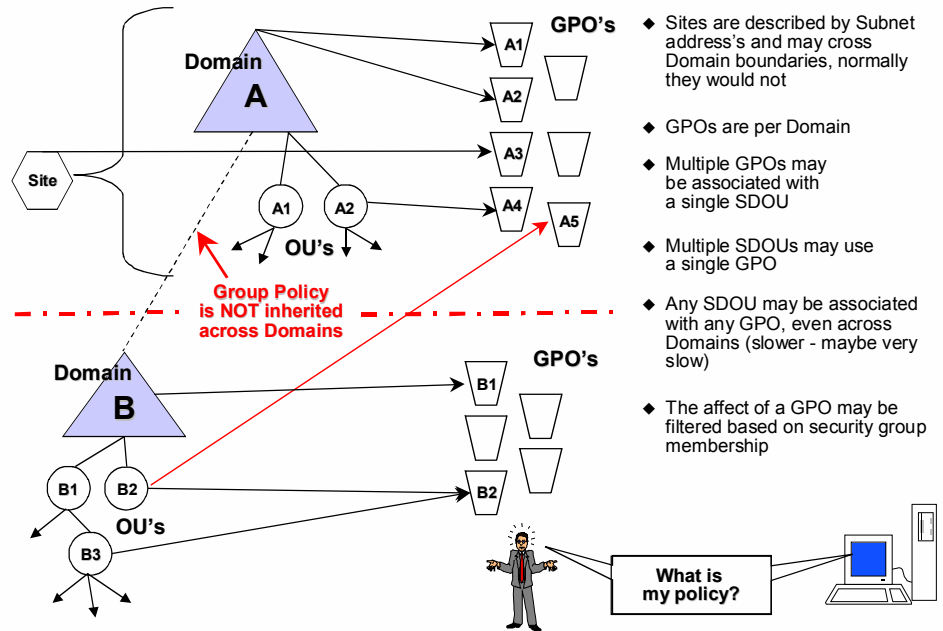
The Windows 2000 Active Directory can specify various user interface (UI) elements on a per-class basis. These elements are property pages, context menus, icons, creation wizards, and localized class and attribute names. This UI information is stored in an Active Directory object called a *display specifier*. Each display specifier object is stored in a container that corresponds to each locale supported by Windows 2000.

The administrative UI for Active Directory is presented through the Administrative Tools menu, which includes the following snap-ins: Active Directory Users and Computers, Active Directory Sites and Services, Active Directory Domains and Trusts, Active Directory Schema Manager, and the Group Policy snap-in. Depending on the extent and nature of the Active Directory schema extensions that you make to define policies and other objects, you should either extend the UI of an existing snap-in using display specifiers or create a new MMC snap-in.

Policy Decision Point and Group Policy

Windows 2000 provides an extensible policy framework called Group Policy that allows administrators to centrally deploy policy-based management of users' desktops. The tool for defining and manipulating policy is an MMC snap-in, called the Group Policy snap-in, which stores policy settings in Group Policy objects (GPOs). The GPOs are in turn associated with the Active Directory containers: sites, domains, and OUs. Multiple containers can be associated with the same GPO, and a single container can have more than one associated GPO. The type of policy settings that can be specified includes not only registry-based policy, but also includes security policies, software installation and maintenance options, script options, and folder redirection options.

The diagram below illustrates the relationship between Group Policy and Active Directory. The Group Policy infrastructure includes a policy store, a policy engine that runs as part of the computer startup and user logon (WinLogon), and an API for services to invoke the policy selection process on demand (GetGPOList).



By default, Group Policy affects all computers and users in a selected Active Directory container. The default order of precedence follows the hierarchical nature of Active Directory: sites are first, then domains, and then each OU. If you have more than one GPO associated with an Active Directory container, a GPO that is higher in the list has higher precedence. Use the GPO context menu to view and modify information about the GPO. You can also filter the effects of Group Policy based on users' or computers' membership in a Windows 2000 Security Group. To filter Group Policy, you use the **Security** tab on a Group Policy Object's **Properties** page to specify discretionary access control list (DACL) permissions. To delegate the use of the Group Policy snap-in tool, you use DACL permissions.

The Group Policy snap-in includes several snap-in extensions. A Group Policy snap-in extension may extend either or both of the User or Computer Configuration nodes in either the Windows Settings node or the Software Settings node. Most snap-ins extend both of these nodes, but frequently with different options. The following is a list and brief description of the Group Policy snap-in extensions that are included in Windows 2000:

- **Administrative Templates**—Includes registry-based policy settings, which you use to mandate registry settings that govern the behavior and appearance of the desktop, including the operating system components and applications. The Administrative Templates snap-in extension also includes functionality for managing Disk Quotas and Remote Installation options.

-
- **Security Settings**—You use the Security Settings extension to define security configuration for computers within a GPO. You can define local computer, domain, and network security settings.
 - **Software Installation**—You use the Software Installation extension to centrally manage software distribution in your organization. You can install, assign, publish, update, repair, and remove software for groups of users and computers.
 - **Scripts**—You can use scripts to automate computer startup and shutdown, and user logon and logoff. For these purposes, you can use Windows Scripting Host to include Visual Basic®, Scripting Edition (VBScript), and Jscript® type scripts.
 - **Folder Redirection**—Allows you to redirect special folders to the network.

Policy Enforcement Point

A policy enforcement point is a discrete location in the network where the policies selected at a decision point are enforced. Enforcement points and decision points can be co-located, but this need not be the case. Typically, enforcement points are in network components, such as routers, switches, network access servers (NAS), private branch exchange (PBX), Voice over IP (VOIP) gateway/gatekeeper nodes and similar devices/servers.

A network component that also has the decision point (policy server) function embedded in it needs to interact with Active Directory using secure LDAP connections for policy transactions. A network component that only serves as the enforcement point also needs to interact with Active Directory securely to publish/update device data about itself. In either case, there is a need to have secure LDAP client functionality in the network equipment. This can be enabled in one of three ways:

- Host the network component on Windows 2000-based computer so that it not only gets secure LDAP client runtime, but also has access to the rich features and functionality of the platform.
- Run the Windows 2000 Embedded operating system, when it is available, as the control operating system on blade in the network component, where it is possible, to again obtain the same benefits as in the point above while maintaining a small footprint and optimized efficiency.
- Microsoft is considering making available a secure LDAP client for Active Directory, based on the LDAP v3 and Kerberos v5 protocols, in source code format to allow vendors to quickly add secure LDAP client functionality to network components. Starting with this source code version of the secure LDAP client would help speed up time to market significantly.

Programmatic Interfaces for Active Directory

Active Directory offers application developers two levels of programmatic access: the LDAP C APIs and the Active Directory Service Interfaces (ADSI). ADSI is a set of COM programming interfaces that make it easy for customers and ISVs to create applications that register with, access, and manage multiple directory services with a single set of well-defined interfaces.

Dealing with Replication Latency and Data Consistency

The replication model used in Active Directory is called multimaster loose consistency with convergence. This model lets you update the directory at any domain controller (DC). The replication system propagates changes made at any domain controller to all other domain controllers. The replication partners are not guaranteed to be consistent with each other at any particular point in time (loose consistency), since changes can be applied to any domain controller at any time (multimaster). If the system is allowed to reach a steady state, in which no new updates are occurring and all previous updates have been completely replicated, all domain controllers are guaranteed to converge on the same set of values (convergence).

This replication behavior is consistent and predictable. Given a set of changes to a given replica the outcome can be predicted—the changes will be propagated to all other replicas. Devising a reliable general model for predicting when the changes will be applied at all other replication partners, or at a particular domain controller, is impossible, because the future state of the distributed system as a whole cannot be known. This is called non-deterministic latency, and applications that use the directory must understand and allow for it. The best way to accommodate replication latency is to design applications that minimize the effects. The ideal directory-enabled application is insensitive to version skew, does not depend on relationships among multiple objects, and has no intra- or inter-object consistency requirements.

Applications and services that fit this profile need not be concerned with replication latency. Other applications must be designed with replication latency in mind. The key to success in designing such an application is awareness of the replication process. Steps taken at design time to reduce inter-object dependencies and minimize partial update windows will pay large dividends at run time. In general, the simpler and less inter-related objects are the more stable and reliable the implementation that depends on them will be.

Roadmap to Active Directory Enabled Networking

The road to a successful implementation of policy-based networking integrated with Active Directory consists of the following steps of development work.

1. Design the necessary directory schema extensions and extend Active Directory schema.
2. Design and implement new MMC snap-ins or extensions to existing MMC snap-ins to contain policy and administrative information.
3. Design administrative templates for policy settings (optional).
4. Plan how the directory is to be populated with device and configuration data.
5. Plan how policy will be enumerated and evaluated for a given entity, such as a router or service, and then delivered to that entity.
6. Provide for detecting changes to policies in the directory and doing change notification to entities that are affected.
7. Evaluate the best way to expose dynamic data and network state information without affecting directory performance (optional).
8. Plan how to support legacy devices already deployed in a network infrastructure for policy-based management (optional).
9. Integrate with Windows Installer, when appropriate, to allow central management of application and service installation and configuration, as well as uninstall.

Following is a description of each of these steps along with pointers to further documentation and development materials available for the Windows 2000 operating system.

Schema Extensions Design

The schema contains a definition of each object class, and each object class's attributes, that can be stored in a directory. The schema is stored in Active Directory. Schema definitions are themselves also stored as objects—Class Schema objects and Attribute Schema objects. This lets Active Directory manage class and attribute objects in the same way that it manages other directory objects. Applications that create or modify Active Directory objects use the schema to determine what attributes the object must or might have, and what those attributes can look like in terms of data structures and syntax constraints.

Extend the schema only if no existing object class meets your needs. Extending the schema is an advanced procedure and schema changes are global and replicated to every domain controller in the enterprise. Consider your needs carefully and assess the scope of the schema extension that is necessary. Further, bear in mind the implications imposed by replication latency and multi-object consistency while designing a schema.

You can extend the schema in the following ways:

-
- Derive a new subclass from an existing Active Directory base class—the subclass has all of the attributes of the superclass and any additional attributes you specify. This is the preferred mechanism for adding attributes to an existing class.
 - Derive a new subclass using the evolving LDAP schema based on the CIM/DEN specification.
 - Create an entirely new class (derived from the special class TOP) with the attributes you want if no existing class meets your needs.

Once you have decided that schema extension is required, do the following:

- Name the new attributes and classes.
- Obtain schema object IDs (OIDs) for new attributes and classes.
- Use display specifiers to integrate new attributes and classes with the user interface, if necessary.
- Create a program or an LDIF script to install your schema extensions when creating new attributes and classes, or adding attributes to existing classes.

During development and testing phases, use a class and attribute versioning scheme to avoid reinstalling the directory service to test new models. In a multiple domain controller environment, use a version specific suffix during the development cycle, for example, `cn=MSFT-DS-Attribute-One-001`. If you need to change the syntax of an attribute, for example, then you can create a new attribute named `cn=MSFT-DS-Attribute-One-002`. Older versions as well as new versions of the application will work. The final version of the application would use `cn=MSFT-DS-Attribute-One` (without the version suffix). Further, in a single DC environment, you can use the DCPROMO tool (or the Configure Your Server wizard) to first demote and then promote the DC, which removes the schema extensions.

Management Console Snap-Ins

If your application must provide other types of policy settings, or you would like to provide a richer user interface for registry-based policy settings, create your own snap-in extension for the Group Policy snap-in.

Options for designing and implementing MMC snap-ins to manage policy and other newly defined directory objects are the following:

- Extend one or more Active Directory base class snap-in, if appropriate. If there is already a Windows 2000 service that meets your needs, you should extend the UI for the corresponding snap-in.
- Create object creation wizard extensions (that is, add creation wizard pages) for the new attributes and classes being defined.

-
- Create and register a new MMC snap-in.

Policy Templates Design (Optional)

If you have an application or service that will use the local registry of the system where it is hosted to store policy settings, then design administrative template files for the registry-based policy settings.

Options to design the policy templates for registry-based policy settings are the following:

- Derive from an existing Windows 2000 Group Policy template that most closely matches what you want to do.
- Create a new template with the policy settings you want when no existing template meets your needs.

Directory Population

An important issue for the effective use of directory enabled networking is how to populate the directory with necessary policies and the relevant information about network devices (like routers, switches, PBX, Voice over IP devices). From a usability standpoint, an important consideration for network administrators is how to automate adding information about new devices to the directory when they are added to the network infrastructure. Once a device is represented in the directory, it is centrally available for asset tracking, configuration management, and service provisioning.

The policies must be administratively defined through the Group policy snap-in. The preferred method of populating device data in the directory is to have the devices self-publish this information themselves. Options to enable the directory to be populated with relevant device data are the following:

- Host the network component on Windows 2000, where possible, to take advantage of the inherent ADSI and LDAP support. Examples of network components where this would be feasible are PBX devices and Voice-over-IP gateway/gatekeeper components. The devices can self-publish data in Active Directory as per predefined directory schemas.
- When the product is available, use the Windows 2000 Embedded operating system (now in development by Microsoft) as the control OS on blade in the network device (router, switch) where the device architecture allows it. The device can publish data about itself in Active Directory as per predefined directory schemas.
- Implement and integrate a secure LDAP client in the device operating system, starting from an LDAP client in source code form, such as the one Microsoft is considering making available. Using the LDAP client functionality, the device can self-publish data about itself into Active

Directory as per predefined directory schemas.

- Provide for scripted addition of device data to Active Directory from management stations.

In the cases above, the LDAP client binds to a configured container in Active Directory and adds or updates the device information there either at startup or when a script is run.

Policy Enumeration and Evaluation

A policy-based management system not only needs a way to create and store policies (this is done with the Group Policy snap-in or a separate policy console MMC snap-in), but also a way to retrieve and apply policies. The latter function is normally done directly by the client device in a two-tiered policy architecture, or by a policy server that enumerates and retrieves policies from Active Directory and distributes it to the client devices in a three-tiered architecture.

In two-tiered policy architecture, you can use the Group Policy infrastructure provided in Windows 2000 to configure policy settings for your network devices and applications. In a three-tiered policy architecture, you must build a policy server that retrieves policy from Active Directory and provides the policy engine functionality.

The following options can be used to implement policy enumeration, evaluation, and distribution:

- Host the network device or service on Windows 2000, where possible, to take advantage of the Group Policy client infrastructure support. The network device or service can obtain its policy settings downloaded to it from the policy engine in the Group Policy infrastructure. When necessary, the *GetGPOList* API can be used to retrieve applicable policy on demand.
- When it becomes available, use the Windows 2000 Embedded operating system as the control operating system on the blade in the network device (router, switch). This option allows you to use the Group Policy client infrastructure support to allow the device to retrieve policy from Active Directory in a secure fashion. When necessary, the *GetGPOList* API can be used to retrieve applicable policy on demand.
- Implement and integrate a secure LDAP client in the device operating system starting from an LDAP client in source code form, such as the one Microsoft is considering making available. The device can interact with Active Directory through secure LDAP connections to retrieve applicable policy and apply it.
- When the Group Policy framework does not meet the policy needs, implement a policy server with full PDP functionality using Active Directory as the policy store.

In the cases above, the LDAP client (either on the device or in the policy server) binds to a configured policy container in Active Directory, retrieves policy information, and enumerates applicable policy for the network component before applying it.

Policy Events/Triggers

A mechanism is needed to notify consumers of changes to policy in Active Directory so that affected entities can reread policies and enforce the new policies. In the Windows 2000 Group Policy infrastructure, policy is applied when the computer starts up and when the user logs on. When a user turns on the computer, the system applies computer policy. When a user logs on interactively, the system loads the user profile and then applies user policy. Policy can be optionally reapplied on a periodic basis. By default, policy is reapplied every 1.5 hours. The Group Policy snap-in can be used to reset this interval. Policy can also be reapplied on demand by using the *RefreshPolicy* function that refreshes the current policy settings.

Active Directory also provides a mechanism for a client application to register with a domain controller to receive directory change notifications. To do this, the client specifies the LDAP change notification control LDAP_SERVER_NOTIFICATION_OID in an asynchronous LDAP search operation. Once a client has registered a notification request, it continues to receive notifications until the connection is broken. This mechanism can be used by policy servers to receive notification of changes in the directory to policies that they control and monitor, retrieve the changed policies, and send resulting actions to PEPs for enforcement.

The following options can be used to implement policy event triggers:

- Host the network component or service on Windows 2000 to take advantage of the inherent Group Policy client support. The network component, in this case, can use the startup/logon policy application and the interval timer to refresh policy settings.
- Using a secure LDAP client implemented in the device operating system, starting from an LDAP client in source code form, such as the one Microsoft is considering making available, the network component uses the container hierarchy in Active Directory to determine the effective policy at startup and at configured intervals thereafter.
- Same as the second item above but done by the policy server as opposed to the device in a three-tiered policy architecture.

In all cases above, the policy consumer (either the device or the policy server) can use the LDAP change notification control mechanism supported by Active Directory for immediate notification of policy changes in the requested search scope.

Dynamic State Information (Optional)

A policy-based networking system needs to maintain state information about an entire network transaction to fully realize the potential of the intelligent network. For example, a RADIUS server, when authenticating and granting access to a user requesting dial-up remote access, might want to consult a dynamic state server keeping remote access session information in order to limit the number of simultaneous log on sessions that a single user can have running. It might also maintain accounting records for the session in the same dynamic data store. Other examples of dynamic network state information arise from keeping data on QoS sessions and DHCP leases. Obviously, such a state server requires a scalable infrastructure that can support a large number of updates and queries per second.

Since directories in general are designed to store relatively static data—data that at least changes slower than the replication latency so the changes propagate to all replication partner—they cannot adequately accommodate this type of state information. Consequently, you will have to look for other real-time data stores to maintain volatile state information. The challenge lies in evaluating the best approach to expose this type of data in a seamless fashion as you do with other data accessed from Active Directory.

Active Directory offers one feature that can address this requirement in a limited fashion. While defining attributes in the Active Directory schema (by way of *attributeSchema* objects), the attribute can be flagged as non-replicated with the connotation that changes to the attribute do not get propagated outside the domain controller where the change is initiated. This is accomplished by turning on the `FLAG_ATTR_NOT_REPLICATED` flag in the *systemFlag* attribute of the schema definition for the attribute. By eliminating the need to replicate changes and hence the latency associated with it, the domain controller hosting the objects with the non-replicated attributes can effectively be turned into a dynamic state server that serves out volatile data. However, such a setup still suffers from having a single point of failure.

Options to implement and integrate a dynamic data store with Active Directory to serve volatile state information are the following:

- Make use of non-replicated attribute definitions in Active Directory schema to host volatile data on a dedicated Active Directory domain controller.
- Evaluate and integrate other external store to serve the volatile state information.

Policy Proxy (Optional)

As discussed in the [“Support for Legacy Devices”](#) section, a policy proxy is an entity that can enable support for legacy network devices by serving as a proxy for legacy devices from the standpoint of PEP and PDP style communications to effect policy decisions and distribution of policies through a policy transaction

protocol. In essence, the policy proxy translates policy actions handed down from PDPs and communicates them to the legacy devices using a command line interface (CLI) or SNMP.

The following options can be used to either implement or eliminate the policy proxy functionality:

- Upgrade the network component or service host with LDAP client support so it can participate directly in the policy transaction protocol with the policy server or Active Directory.
- Upgrade the network component or service host with support for the Windows 2000 Embedded operating system (when available) to serve as the control operating system on blade, which accomplishes same result as in the item above.
- Build a policy proxy server that translates policy actions into SNMP *Get* and *Set* operations.

Windows Installer Integration

Windows Installer centrally manages application installation configuration as well as application uninstall. Using this service enables robust installation and self-repairing applications and services; reliable and complete uninstall, including correct handling of shared components; and use of IntelliMirror® technologies for policy-based deployment, update, and uninstall over a network.

You can use the following options to integrate install/uninstall of your software with the Windows Installer service:

- Derive from sample code using Windows Installer service APIs in the Platform SDK.
- Build your own install and configuration routine.

Summary

The benefits of directory enabled networking, including simpler QoS, configuration, and security administration are compelling to network managers. As directories take on expanding roles within the enterprise, they will form an integral part of the network services infrastructure and become more important to customers. DEN offers the prospect of more flexible and powerful network management applications that can become interoperable.

There are a variety of core services in Windows 2000 Active Directory that can be used to both extend existing network services as well as offer new, enhanced services. Although this paper is not intended to be the complete guide for what these services are, how to directory-enable them, and how to deploy them, it is intended to provide a greater awareness of the technology components in Windows 2000 that are relevant to a directory-enabled networking strategy. Further, the roadmap provided for Active Directory enabled networking is intended to be a rough guide of the steps necessary to build on the extensibility features of the platform. The roadmap, with its pointers to appropriate documentation, is an attempt to narrow down the scope of documentation search and help you get off to a quick start with the necessary information. The online Help in Windows 2000 Server also provides a wealth of information on the infrastructure services. Active Directory provides significant levels of extensibility. Making it work for your needs is often only a quick development effort away.

For More Information

For the latest information on Windows 2000 Server, check out our Web site at <http://www.microsoft.com/windows2000> and the Windows 2000/NT Forum at <http://computingcentral.msn.com/topics/windowsnt>.

Additional Web Site Resources

Windows Platform SDK

<http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/psdk/portals/mainport.htm>

Exploring Windows 2000 Management Services

<http://www.microsoft.com/WINDOWS2000/guide/server/features/managementsvcs.asp>

Active Directory Technical Documents

<http://www.microsoft.com/windows2000/library/technologies/activedirectory/default.asp>

Active Directory Services Interfaces

<http://www.microsoft.com/windows2000/library/howitworks/activedirectory/adsilinks.asp>