Understanding Message Routing in Microsoft® Exchange 2000 Server

PT104 **1.0**

From the Exchange 2000 Server Pre-emPt Series

Paul Bowden
Program Manager - Europe
Microsoft Exchange Server Product Unit

Pt
PLATINUM

Microsoft
Technical
Readiness
GROUP

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

Contents

Acknowledgments

Many thanks to the following people for their time and enthusiasm. No white paper is produced without many people going out of their way to assist.

Exchange Product Group

David Lemson

Steve Townsend

Bruce McMillan

Steve Abrams

Chris Boonham

Pretish Abraham

Asit Kini

Geeman Yip

Derik Stenerson

David Madison

Others

Paul Appleby (SRG Engineer, European Regional Support Center)

Laura Payne (Exchange Program Manager, Service Readiness)

Everyone on PlatChat

Philip Beville, Richard Minty, and the rest of the Exchange Global Engineering team at Credit Suisse First Boston

Document Vision and Scope

The purpose of this document is to provide a better understanding of how message routing and transport works in Microsoft® Exchange 2000 Server.

After reading this document, you should understand how to design and deploy routing groups, how to link routing groups, and how link state information is replicated across an Exchange organization. In addition, this white paper discusses how you can customize message transport using your own code.

This information should be part of your learning cycle for Exchange 2000 Server. To gain a full understanding of the other technologies Exchange uses, read the other documents in this series:

- PT100 – Windows 2000 and Exchange 2000 Server Terminology Primer
- PT101 – Deploying Active Directory Connector
- PT102 – Exchange 2000 Server Directory Access and Integration with Windows 2000
- PT103 – Exchange 2000 Server Co-existence and Upgrades
- PT105 - Understanding Exchange 2000 Server Storage Technology
- PT106 - Deploying Exchange 2000 Server Real-time Collaboration Services
- PT107 - Collaboration with Exchange 2000 Server

Note The information in this document is based on Microsoft Windows® 2000 Release Candidate 2 and Microsoft Exchange 2000 Server Beta 3.

Introduction

Exchange Server Message Transport

Any messaging system that must be fast, scalable, and reliable depends on a strong underlying transport and routing engine. This core component is fundamental to the operation of an enterprise messaging system with users who may be distributed around the world. Without an intelligent transport, messaging servers function independently of one another.

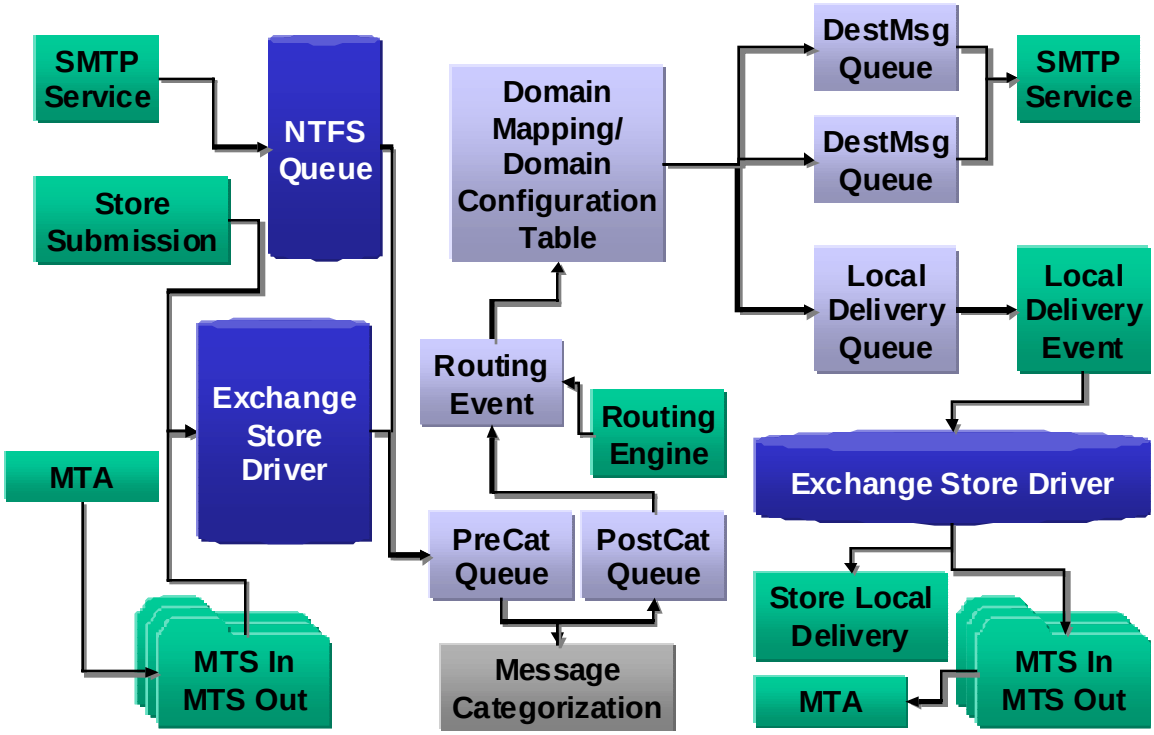
Earlier versions of Exchange incorporated a rich message transfer agent (MTA) built on the X.400 standard. Exchange site boundaries defined the message routing topology in which information was routed with a single hop through remote procedure call (RPC) communications. Between sites, a number of connectors could be deployed to enable messaging, ranging from Site Connectors that used RPC to X.400 connectors and Dynamic Remote Access Service (RAS) connectors.

Exchange 2000 Server builds on the rich heritage of the Exchange MTA but uses a full-featured Simple Mail Transfer Protocol (SMTP) transport for all native communications. The X.400 MTA is still present in this new version of Exchange, and has been improved to support additional functionality, such as Request for Comments (RFC) 2156 MIME Internet X.400 Enhanced Relay interoperability. However, in many circumstances, the X.400 MTA is used to connect Exchange with other external X.400 systems rather than for native message transfer between Exchange 2000 servers.

Using SMTP as the native communication method between Exchange servers opens up a number of new opportunities and eliminates some of the deployment issues of earlier transport implementations. For example, companies with a distributed user base normally designed their Exchange site models based on available network bandwidth instead of designing them for convenience of administration. This is because all Exchange servers within a site use RPC to communicate with one another, and low-bandwidth and high-latency networks are inefficient (and sometimes unworkable) for the synchronous nature of RPC. Because Exchange no longer uses RPC to transfer messages, you can devise a more flexible routing scheme. Additionally, the site concept is now split into administrative groups and routing groups, which allows you more flexibility in your administration and routing models.

In addition to using SMTP as the native transport protocol, Exchange 2000 Server features a new routing calculation engine, which allows the most efficient routing of messages based on the current conditions within the network. Although earlier versions of Exchange provided backtrack and failover messaging, Exchange rerouted messages based on network and server conditions at the point of routing. Exchange 2000 Server uses *link state* information to transfer information about the condition of network and server resources to each messaging server in the organization so that the best routing decision can be made at source.

The entire transport architecture, not only the protocol, has changed in Exchange 2000 Server. The following diagram shows the new Exchange transport architecture.



Administrative and Routing Groups

The site concept in earlier versions of Exchange defined three boundaries: single-hop routing, a collective administration unit, and a namespace hierarchy. To provide a more flexible deployment and administration structure, all three boundaries are separate in Exchange 2000 Server.

- A routing group defines single-hop routing.
- An administrative group defines collective administration.
- The namespace hierarchy exists in Active Directory in the form of a domain.

An administrator normally works with administrative groups and routing groups on a regular basis. Logically, *routing groups* (collections of servers in which servers send messages to one another directly) are located beneath *administrative groups* (collections of Active Directory objects that are grouped together for the purpose of permission management), so a relationship exists between these two entities. Within an administrative group (a group of Exchange servers that you administer as a logical collection), you can configure the servers so that some of them route messages to one another directly and others forward messages to a bridgehead server. In Exchange Server 5.5, the routing and administration models were tied together.

The configuration naming context within Active Directory stores all administrative group, routing group, connector, and cost information. Because of this, all routing information is available to the entire Exchange organization because the naming context is fully replicated (read and write) among all domain controllers within the Active Directory forest. However, although every Exchange server can automatically detect all other servers within the organization, you must still define the connections between servers. This allows for flexible routing.

Mixed-Mode vs. Native-Mode Organizations

An Exchange organization can run in one of two modes: mixed mode or native mode. The mode concept is similar to that of Active Directory domains in which mixed mode allows full compatibility with earlier versions of Exchange, and native mode means that all servers have been upgraded to Exchange 2000 Server. However, there is no relationship between the mode of Active Directory domains and the mode of the Exchange organization; they are independent of one another.

When an Exchange organization is in mixed mode, all administrative groups are mapped directly to Exchange 5.*x* sites; for example, if an Exchange 5.*x* deployment consists of 50 sites, when you install Exchange 2000 Server, 50 administrative groups are created in Active Directory. For compatibility reasons, you cannot move Exchange servers between administrative groups. Additionally, each administrative group has one routing group only, the members of which are the Exchange servers you install in the administrative group. You can create new routing groups within the administration group, which creates a *sub-site* effect. Additionally, Exchange 2000 servers in the same administrative group can route messages to each other using the fast SMTP transport; they are not bound by the bandwidth constraints of RPC. An Exchange 2000 server installed in an Exchange 5.*x* site becomes the Routing Information Daemon for that site. Although Exchange 2000 servers do not use a Gateway Address Resolution Table (GWART) file for their own routing, one generated and replicated using Active Directory Connector (ADC). ADC is a Microsoft® Windows® 2000 service that synchronizes the Exchange 5.5 directory with Windows 2000 Active Directory, allowing you to administer directories from Active Directory or the Exchange 5.5 directory service. Exchange 5.*x* servers can route messages to connectors installed on a computer running Exchange 2000 Server and vice versa. This is an important capability, because you can use these new connectors without upgrading your entire organization. You can also continue to use third-party connectors (such as Fax) that cannot run on Exchange 2000 Server provided an Exchange 5.5 server still exists in the organization. PROFS and SNADS connectors (used to connect Exchange with host-based mail systems) are not included with Exchange 2000.

After you have upgraded all servers in your Exchange organization to Exchange 2000 Server, you can switch the organization to native mode. This allows greater flexibility with routing groups. Multiple routing groups, with servers from other administration groups can be defined within a single administrative group, allowing you to refine point-to-point message routing for your organization. You can also create a single administrative group to hold all routing groups in the organization, while still allowing other administrative groups to handle day-to-day administration of the other components on the server. One of the significant architectural changes between the site model in earlier versions of Exchange and the new groups in Exchange 2000 Server is that you can move objects between the groups using a drag-and-drop operation, providing immediate flexibility when you make changes to the underlying infrastructure.

The following diagram provides an example how an Exchange 2000 organization can be configured; administrators in different geographic regions manage the servers, and a separate team of administrators manage routing.

```
+-- North America Management Team
        +-- Routing Groups
        +-- Servers
                +-- BOSTON-PF01
                +-- LA-PF01
                +-- NEWYORK-MBX01
                +-- PHILADEL-MBX01
                +-- PORTLAND-MBX01
                +-- SANJOSE-PF01
                +-- SEATTLE-MBX01

+-- Worldwide Routing Management Team
        +-- Routing Groups
                +-- Asia
                        +-- Members
                                -- SINGAPORE-PF01
                                -- TAIWAN-MBX01
                                -- TOKYO-MBX01
                        +-- Connectors
                                -- Backup link to USA West Coast
                                -- RGC to Northern Europe

                +-- Northern Europe
                        +-- Members
                                -- FRANKFURT-PF01
                                -- LONDON-MBX01
                                -- PARIS-MBX01
                        +-- Connectors
                                -- RGC to Asia
                                -- RGC to USA East Coast

                +-- USA East Coast
                        +-- Members
                                -- BOSTON-PF01
                                -- NEWYORK-MBX01
                                -- PHILADEL-MBX01
                        +-- Connectors
                                -- RGC to Northern Europe
                                -- RGC to USA West Coast

                +-- USA West Coast
                        +-- Members
                                -- LA-PF01
                                -- PORTLAND-MBX01
                                -- SANJOSE-PF01
                                -- SEATTLE-MBX01
                        +-- Connectors
                                -- Backup link to Asia
                                -- RGC to USA East Coast
        +-- Servers
```

Routing in Exchange 2000 Server

The new routing transport and architecture in Exchange 2000 Server may prompt a new set of questions for administrators who are deploying Exchange 2000 Server:

- Do I design my routing groups the same way that I designed Exchange sites in the past?
- Can I create hundreds of routing groups without impacting the rest of the Exchange organization or infrastructure?
- Will my routing group design have an impact on any other components in Exchange?
- What options do I have for connecting routing groups?
- What protocol is used when Exchange 2000 servers exist in the same site as Exchange 5.*x* servers?

- Is SMTP data going to be larger than messages sent with Exchange Server 5.5?
- Is the link state data going to flood my infrastructure with status messages if my network is continually changing?

An Exchange 2000 Server deployment requires you to approach design and planning differently than you would with earlier versions of Exchange. This document addresses all of these questions.

SMTP Transport Core

Windows 2000 IMS Base Transport

Windows 2000 Server includes a native Simple Mail Transfer Protocol (SMTP) component called Internet Mail Service (IMS). IMS is designed as a 'no-frills' implementation of SMTP and is included as part of the operating system to allow Windows 2000 and other products to use a known, available transport. For example, Directory Service (NTDS) replication can be achieved over SMTP, and Microsoft Office® 2000 can use SMTP for document notifications. Technically, the new Windows 2000 IMS is part of Microsoft Internet Information Services (IIS) 5.0. This new version of SMTP isn't completely new; its history lies in IIS 4.0 and the Microsoft Commercial Internet System (MCIS). SMTP runs as part of the Inetinfo.exe process.

Although you have minimal control over the base IMS, IMS supports many of the ESMTP standards available including VRFY, EXPN, DSN, ETRN, SIZE, TLS, AUTH, 8BITMIME, CHUNKING, ENHANCEDSTATUSCODES, PIPELINING, and RECEIPT. The SMTP engine also supports the scripting of sinks through Collaboration Data Objects (CDO) 2.0, basic expansion functionality, and virtual servers. Messages can be submitted through three mechanisms:

- SMTP protocol through port 25
- Drop-off directory for formatted files (controllable through ACLs)
- CDO 2.0 (CDOSYS.DLL)

Specifically, the base IMS does not support ListServer capabilities, the ability to route messages based on topology link status, or any other advanced queue management mechanisms.

Exchange 2000 Server SMTP Extensions

When you install Exchange on a computer running Windows 2000 Server, the base IMS is extended—not replaced—through the use of transport and protocol event sinks. The following list includes some of the ways that installing Exchange adds functionality to support the enterprise messaging environment:

- Command verbs to SMTP to support link state information (X-LINK2STATE)
- Advanced queuing engine
- Enhanced message categorization agent
- Exchange IFS store driver to allow pickup and drop off from Information Store instead of NTFS

Virtual Servers

Every Exchange 2000 server can host multiple SMTP virtual servers. Each virtual server has its own configuration information, such as bound IP addresses, port number, and authentication settings. By default, only one virtual server is present on each server, which checks for incoming connections on port 25 of all IP addresses. Using System Manager, you can tune and configure the default virtual server to support the configuration you want; for example, you can place a restriction on the virtual server so that it blocks anonymous access or performs reverse Domain Name System (DNS) lookups.

You can create additional virtual servers on the same physical server. In general, you do this to provide separate options for different messaging services and not for scalability purposes, because each virtual server is multi-threaded. Scenarios in which you would create multiple virtual servers include:

- *Application data.* CDO applications may have to be able to send SMTP messages without being restricted by reverse DNS lookup or recipient limits. You can create a separate virtual server on a different listening port to handle this traffic. You control access to each virtual server through permissions; therefore, only specific hosts can route messages to a particular virtual server.

- *Firewall or multiple domain hosting.* By using multiple virtual servers on the same computer, you can tightly control relaying to meet the requirements of internal and external networks.
- *Flexible authentication.* You can configure each virtual server to use different authentication mechanisms (plain text, NTLM protocol, Kerberos version 5).

No matter how many virtual servers exist on a single server, they must belong to a single routing group. It is not possible for a single physical server to span multiple routing groups.

Virtual Server Settings

Each virtual server you create has parameters that you can set. This section lists the major parameters and provides examples of situations in which you should change the defaults.

Listening for Incoming Connections

By default, a virtual server accepts SMTP connections on port 25 and listens on each network interface. When multiple virtual servers are present on the same computer, each virtual server must have a different listening port or bound IP address. For security purposes, you may want to block SMTP connections on certain networks to which the server is directly connected; this is especially true for Internet connections.

Connection Settings

Each virtual server accepts an unlimited number of inbound SMTP connections and creates as many outbound connections as required (these are limited by the resources of the computer). In scenarios in which the Exchange 2000 server performs tasks besides message routing, you may want to prevent the computer from becoming overwhelmed by SMTP connections, and in some circumstances, you may want to apply a limitation for security reasons. You can set the number of inbound and outbound connections and their session time-outs (default of 600 minutes) independently.

You can also configure the virtual server to connect with a port number other than port 25 for making outbound connections.

Logging

Messages passing through the virtual server can be logged a number of ways, including:

- No logging
- IIS Log File Format
- National Center for Supercomputing Applications (NCSA) Common Log File Format
- ODBC Logging
- World Wide Web Consortium (W3C) Extended Log File Format

Access Control

You can secure access to the SMTP port in several ways when a stringent policy for SMTP message transfer is necessary. First, you can require authentication before a message transfer session can be established; second, you can associate a certificate with the virtual server and create a secure channel; and third, you can allow only specific computers to connect to the SMTP port.

SMTP Relaying

By default, messages can be relayed through a virtual server; however, this opens up the possibility of users forging messages from other users. To avoid this, the capability of the virtual server to relay can be dictated by the connecting SMTP client; for example, you can set the virtual server to disable relaying unless the incoming message is from a well-known host (specified by IP address, group IP address, or domain name).

Message Limits

The resources of a virtual server are precious. You can put various message restrictions in place to protect the server from becoming overwhelmed. The following list includes default configurations:

- Message size is limited to 2,048 KB.
- SMTP session size is limited to 10,240 KB.
- Number of messages for each connection is limited to 20.
- Number of recipients for each message is limited to 100.

You can adjust these parameters to achieve a balance between functionality, flexibility, and performance. For some companies, limiting the size of a message to 2 MB is too restrictive, so this limit can be raised. Alternatively, a limit of 100 recipients in the message header might produce excess network traffic and you may need to increase it.

When there are more recipients in the header of a message than the number you configure on the virtual server, multiple messages are generated; for example, if a message is addressed to 150 recipients, the routing engine transfers two identical messages: one for the first 100 recipients and another for the remaining 50 recipients. The split is invisible to recipients because it occurs only on the Request for Comments (RFC) 821 transmission. The RFC 822 envelope, which is what the user sees when opening the message, is intact with all 150 recipients showing in the header.

When a virtual server handles a constant stream of messages between the same servers, you can increase performance by 5 to 10 percent by not limiting the number of messages in a single connection.

Message Delivery

Most of the time, each virtual server attempts to deliver a message as soon as the message arrives in the queue; however, when there is a temporary problem with the next-hop server, or if a communications failure occurs on the network, the virtual server takes appropriate action, such as queuing the message for subsequent retries or rerouting the message.

If a message is in the queue longer than the period of time set on the server (the default is 12 hours), the sender is notified that the message has not been successfully delivered. If the message has still not cleared from the queue after two days, a non-delivery report (NDR) is generated and sent to the sender.

Reverse DNS Lookup

Because SMTP is so simple, some users may use a mechanism to forge messages from other users. To prevent this, you can configure a virtual server to perform a reverse DNS lookup on the sender of the message. If the submitting SMTP client does not belong to the DNS domain that matches SMTP domain name specified in the **Mail From** field, the virtual server rejects the message. Unfortunately, reverse lookups severely impact the performance of message transfer and prohibit messages from being relayed through multiple hops.

Note that reverse DNS lookups provide a partial solution to the problem of forgery. If users need to verify that the sender identified in the message actually sent the message, you should use digital certificates instead.

Metabase

All Internet services, such as Web, SMTP , File Transfer Protocol (FTP), and Network News Transfer Protocol (NNTP), store their configurations in and retrieve their configurations from a metabase. A metabase is a small database similar to the registry in Windows NT or Windows 2000, which is optimized for this type of data. Although it is possible that Active Directory could store this data , IIS 5.0 must be able to work independently, without the need for you to install Active Directory. In an enterprise environment, to alter the configuration of Exchange 2000 Server SMTP, you must modify the metabase on the local computer directly. This is not feasible in the Exchange environment because a remote administrator may need to make the change. To solve this problem, System Manager stores all

configuration data in Active Directory; however, a component of each Exchange 2000 server called Directory Service to metabase update agent replicates any configuration data that has changed to the local metabase. This is a one-way replication mechanism.

The metabase can be queried with the MetaEdit 2.0 tool available in the *Windows 2000 Resource Kit*.

Advanced Queuing Engine

The Advanced Queuing engine is central to the Exchange 2000 Server SMTP transport. All messages submitted to the Exchange server (including local messages) must pass through the Advanced Queuing engine; for example, when a local user sends a message, the store process notifies the engine that a message must be routed. The engine is passed a file handle relating to the message so it can parse the header. From here, the engine may hand the message to the categorizer, custom event sinks, and router to get the message on its way. However, after each of these components process the message, the component always passes the message back to the Advanced Queuing engine, which acts as an information controller.

There are two fundamental differences in the way that messages are handled in earlier versions of Exchange and in Exchange 2000 Server. First, in Exchange 2000 Server all messages are sent to the transport, even when the sender and recipient are located on the same server. This allows custom event sinks to operate even in a single-server environment. Second, the Advanced Queuing engine reads the message data directly out of Information Store through a file handle, which enhances performance. In earlier versions of Exchange, the message transfer agent (MTA) process physically copies the data out of the store and then transports it. You can observe this change yourself; messages sent from an Outlook client tend to sit in the Outbox slightly longer than with earlier versions of Exchange.

After the path of a message is determined, the message goes into a queue for final delivery. You can query these queues using System Manager. Queue types are as follows:

- *Domain queues* provide a view, by physical domain, of all messages that are destined for the same SMTP domain.
- *Link queues* provide a view, by logical link, of all messages that have the same next-hop in the routing infrastructure.

Categorizer

The message categorization agent performs lookups and checks limits and restrictions in Active Directory. Additionally, the categorizer handles group expansion. Windows 2000 includes a basic message categorization agent called CAT.DLL. Installing Exchange 2000 Server upgrades this component to a categorization agent that can read Exchange-specific attributes (such as HomeMDB) in Active Directory. This new functionality is provided by PHATCAT.DLL.

On receiving the message (by means of an IMSG interface) from the Advanced Queuing engine, the message categorizer in Exchange 2000 Server performs several steps:

1. It resolves the envelope sender and searches for that address in **proxyAddresses** attributes in the directory service to resolve the address.
2. It resolves the envelope recipient list and searches for each address in **proxyAddresses** attributes in the directory service to resolve the address.
3. If the envelope recipient list includes distribution groups (the Windows 2000 equivalent of distribution lists), the message categorizer expands the recipient list to include those members if expansion is allowed on this server.

   For distribution list expansion, Exchange Server 5.*x* uses the property **Home-MTA** for defining distribution list expansion servers. Exchange 2000 Server uses the property **Home-VSI** (Home-Virtual Server Instance).
4. If any recipient cannot be resolved, the message categorizer marks that recipient as unknown.
5. It applies any limits for each sender and for each recipient and marks recipients appropriately.

6. It bifurcates the message if multiple copies of the message must be created because different recipients have different properties set.

   An example of this is when a users sends a message with a read receipt request to a hidden group and an additional individual in the **To** field. Because the membership of a hidden group should be confidential, read receipts should not be generated. In this case, two copies of the message are created: one for the single recipient and one for the hidden group, because the read receipt request must be stripped from the message destined for the hidden group.

   Another example in which bifurcation takes place is when a message is destined for internal Exchange users and external Internet users. A message destined for an Internet user must be in a different format than a message destined for an Exchange user. Bifurcation occurs on the source server to which the client submits the message.

7. It marks each recipient as either *Gateway*, meaning that the recipient can be reached through the MTA, or *Local*, meaning that the recipient is on a local store.

After categorization, the message goes into a queue for each destination domain inside the Advanced Queuing engine.

Router

The router portion of the transport determines the best next-hop for a message. The router uses information about connectors, costs, and link states to determine this.

Store Driver

The base Windows 2000 Internet Mail Service (IMS) uses the file system as its basic repository for messages. This is normally referred to as the Mailroot directory, and has folders such as Queue, Badmail, Drop, and Pickup. To allow complete integration between the SMTP transport and the Exchange server, Exchange 2000 Server installs a *store driver,* which allows the transport to directly read and write files out of Exchange Information Store instead of NTFS. The Mailroot directory still exists, although it is relocated to the \Exchsrvr directory when you install Exchange.

Transport and Protocol Event Sinks

You can extend the SMTP transport and protocol by using *event sinks*, which are pieces of code that activate on a predefined trigger, such as receiving a new message. Essentially, you can use any programming language that is compatible with COM to write an event sink. After you write the event sink, you register it with the transport through bindings, which are stored in the metabase. There are many events for which you can register event sinks. Some are available for Collaboration Data Objects (CDO) programmers (such as **ISMTPOnArrival**) through Microsoft Visual Basic®, Visual Basic Scripting Edition (VBScript), and C++, whereas others are more low-level (such as customizing messages when they go into or come out of the categorizer) and can be accessed only through C++ or Microsoft Active Template Library (ATL) interfaces.

Essentially, there are two types of event sinks in the area of the transport:

- *Transport events.* These are generally used to pass an incoming or outgoing message through a custom process before storing or relaying the message; for example, the event can alter the structure of the message, such as adding additional information such as disclaimers, or it can pass the message to a compression agent before submitting the data to the wire.
- *Protocol events.* These are used to extend the existing SMTP command verbs for custom applications or to capture command verb events; for example, a size restriction event can monitor the BDAT verb to ensure that large messages are not attempted for transfer. Some internal components of Exchange are written as protocol event sinks. For example, the link state protocol is a protocol sink that adds extra command verbs (such as X-LINK2STATE) to the core SMTP command language.

Note Exchange 2000 does not include a message compression agent.

Event sinks can be useful for the following purposes:

- Adding a disclaimer to outgoing messages
- Adding a warning for messages coming in from the Internet
- A basic file virus scanner
- Compression agent for attachments
- Rewriting e-mail domains on outbound messages
- Customized message journaling or message log
- Mailing list service
- Customized auto-replies
- Spam prevention
- Overriding the internal routing decision (sinks written in C++ only)

Implementing a CDO Transport Event Sink

Because events sinks are such a powerful feature of transports, it's beneficial to consider some examples of how you can write and register an event sink . The following process describes how to use Visual Basic to write and register a CDO event sink that checks the file name of attachments in messages and then registers it with the transport:

1. On the Exchange 2000 server, launch Visual Basic 6.0 and create a new Microsoft ActiveX® DLL.
2. Set fundamental properties, such as project name and class name.
3. Add the following type library references to the project:
    - CDO for Exchange 2000—provides message OnArrival interface
    - Server Extension Objects COM Library—allows caching of your DLL
    - ActiveX Data Objects 2.5—gets stream properties of the message
    - Other support libraries you need for your code—for example, Microsoft Scripting Runtime, to interact with the local filing system
4. Write your code; for example:

```
'This event sink checks the names of attachments in the message and
if it finds one called "WORM.ZIP" it aborts message delivery.

Implements IEventIsCacheable

Implements ISMTPOnArrival


Private Sub IEventIsCacheable_IsCacheable()

 ' just return S_OK which means do nothing!

 ' Object will be cached for subsequent use.

End Sub


Sub ISMTPOnArrival_OnArrival(ByVal iMsg As CDO.Message, EventStatus
As CdoEventStatus)

Dim BodyParts As CDO.IBodyParts

Dim BodyPart As CDO.IBodyPart
```

```
Dim flds As ADODB.Fields


EventStatus = cdoRunNextSink     ‘ normally run next sink
Set BodyParts = iMsg.Attachments   ‘ get attachments
If BodyParts.Count > 0 Then      ‘ check we have some
For Each BodyPart In BodyParts
If UCase(BodyPart.FileName) = “WORM.ZIP” Then
Set flds = iMsg.EnvelopeFields
flds(“http://schemas.microsoft.com/cdo/smtpenvelope/messagestatus”)
= cdoStatAbortDelivery
flds.update
EventStatus = cdoSkipRemainingSinks ‘ Skip other sinks
End If
Next
End If
End Sub
```

5. Compile your code as a DLL.
6. Register your DLL using REGSVR32 *DLLNAME*.DLL
7. Register your sink with the SMTP virtual server. You can do this using one of the following methods:
   - Write code
   - Use the Smtpreg.vbs script supplied with Exchange 2000 Server
   - Use the Transport Event Sink Registration Wizard on the Microsoft Platform SDK

   As part of the registration, you need to specify a priority between 0 and 32,767 (a lower number equals a higher priority) and a rule (which you can leave blank).
8. Test your event sink by sending a message. For this particular event sink, you may want to use Outlook Express to simulate a message coming in from the Internet with an attachment called Worm.zip.

Performance Considerations for Event Sinks

Of the three major programming languages you can use to create your sink, a DLL written in C++ executes the fastest. Following closely behind is a DLL written with Visual Basic, and behind that is a script written with VBScipt or JScript. When possible, avoid writing event sinks using a scripting language because they don't support early bindings and are not cacheable. Additionally, consider your resources to maintain the code as your needs change. Although an event sink written with Visual Basic may be slower than on written with C++, if many people on your team know Visual Basic it may be the best option. Above all, you need to thoroughly stress test your event sink to ensure that it performs at the level that you require. Additionally, you need to ensure that your event sink runs well and contains no errors. All event sinks are synchronous, which means that if errors exist, event sinks can severely degrade performance or in a worst case scenario, stop messaging in your Exchange organization.

For more information on transport event sinks, see the Platform SDK CD-ROM, which is part of Microsoft Developer Network (MSDN™).

SMTP and the Client

The client of choice for Exchange 2000 Server is Microsoft Outlook, which uses MAPI as a transport mechanism. Clients should not send or receive messages differently simply because the native message transfer protocol is now SMTP. All rich-text information is sent in Transport-Neutral Encapsulation Format (TNEF) and is preserved from end to end.

In SMTP, Delivery Status Notifications (DSNs) report the delivery status of a message. The Exchange server can translate the standard responses into messages that a MAPI client can read, as follows:

- FAILURE becomes a non-delivery report (NDR)
- SUCCESS becomes a delivery receipt

Delivery Options for each Domain

Under the Global Settings node in System Manager, you can define configuration properties for each external domain that the organization connects to, as follows:

- Send messages as Multipurpose Internet Mail Extensions (MIME) or uuencode
- Provide message body as plain text or HTML
- Select MIME and non-MIME character sets
- Send rich-text format
- Select word wrap
- Define allowed types of messages: out of office, automatic replies, automatic forward
- Preserve sender's display name

In earlier versions of Exchange, these settings were based on an individual IMS, which made management across multiple servers difficult because these settings had to be duplicated.

Global Message Delivery Options

Under the Global Settings node, you can define company-wide message size limits (incoming and outgoing), recipient limits, and options to prohibit junk e-mail.

Designing Routing Groups

Introduction

An Exchange routing group defines a collection of Exchange 2000 servers that communicate directly with each other through mesh-like connectivity. Each server may have one or more SMTP virtual servers and possibly an X.400 MTA and stack defined on it.

Unlike the site concept in earlier versions of Exchange, you can create and remove routing groups as you need to, and you can dynamically alter service membership. In other words, you can easily change the entire routing architecture for an organization without reinstalling Exchange. As the underlying network infrastructure changes when new network links are created or upgraded, the Exchange routing network can also change to take maximum advantage of the change. The only prerequisite for this is that the Exchange organization must be in native mode.

The only transport protocol used between Exchange 2000 servers in the same routing group is SMTP. In a pure Exchange 2000 Server environment, remote procedure call (RPC) connectivity is not offered as an option. If you install an Exchange 2000 server in an existing Exchange 5.*x* site, RPC is used between the Exchange 2000 and servers running earlier versions of Exchange; however, even in the same routing group, Exchange 2000 servers communicate with each other using SMTP.

Planning Routing Group Boundaries

Administrators of Exchange 2000 Server tend to plan their routing groups the same way that they planned Exchange 5.*x* sites—by the availability and reliability of network bandwidth. This is a good approach to begin with, and in the majority of situations, your routing groups will be organized the same as earlier versions of Exchange sites during the coexistence phase. Perhaps one of the biggest factors for dividing older Exchange designs into sites was the issue of synchronous RPC connectivity and its implications for slow or unreliable networks. Exchange 2000 servers in the same routing group communicate with each other using SMTP, which is asynchronous and can work efficiently on all types of network bandwidths and latencies; therefore, large routing groups are less of a concern.

The most important factor to consider when you are determining routing group boundaries is the stability of the network connections between the servers within the routing group. In cases in which network links are prone to failure or occasional problems, you should divide the servers into separate routing groups.

In the majority of installations, SMTP is used to connect routing groups. Because this protocol is tolerant of many network conditions, you might wonder if there is any benefit in dividing Exchange servers into routing groups.

Same Routing Group

The following conditions must exist for servers to belong to the same routing group:

- Exchange servers must belong to the same Active Directory forest.
- Exchange servers must have permanent, direct SMTP connectivity to each other.
- All servers within the routing group should always be able to contact the routing group master, which this paper discusses later.

Multiple Routing Groups

The following are reasons to divide installations into multiple routing groups:

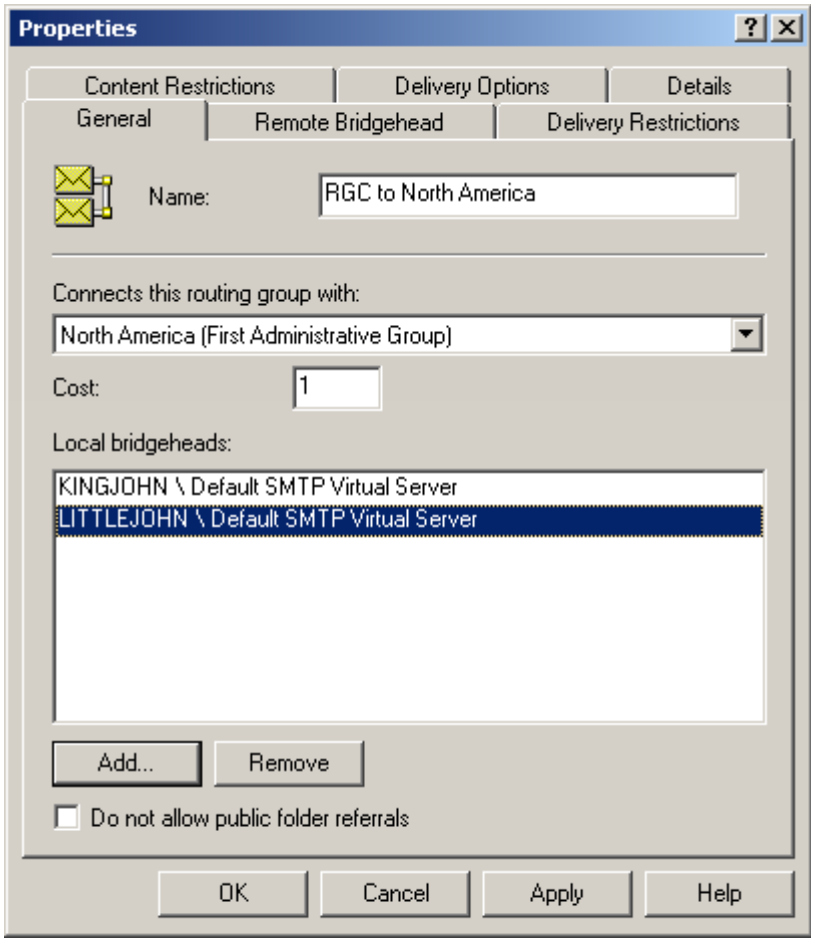
- The base prerequisites are not met.
- The underlying network is frequently unreliable.
- The messaging path must be altered from single-hop to multi-hop.
- If messages should be queued and sent on a schedule.

- For extremely low-bandwidth connections for which X.400 connectivity may be more appropriate.
- For client connections to public folders, because this is based on the routing group architecture.

Routing Groups and Public Folders

In Exchange Server 5.*x*, clients use the site boundary to determine the closest replica of a public folder. Exchange Server 5.5 introduced the concept of a sub-site that allows an extra level of granularity within a site. Across sites, you can assign affinities so that a local client can access a public folder beyond the site boundary.

In Exchange 2000 Server, clients access public folders by making a call to routing; therefore, public folders on servers that are in the same routing group in which the user's mailbox resides are preferred. When a user attempts to access a public folder that has replicas only in remote routing groups, the messaging connector cost determines the closest replica; this is the equivalent of an Exchange 5.5 affinity. Note that these affinities are automatically enabled (unlike Exchange 5.5), but there is a flag on the messaging connector to prevent public folder referrals across the link.



Routing Groups and Windows 2000 Sites

A Windows 2000 site is defined by a collection of resources or IP subnets that have good connectivity to one another. The design methodologies for routing groups are similar to this, and you might wonder why sites and routing groups aren't tied together. There are a number of reasons for this. First, customers state that distinct groups of administrators control enterprise messaging and the directory and network infrastructure. If a network administrator moves a server to a different IP subnet, the Exchange administrator does not necessarily want the server moved to a different routing group

because doing so could negatively effect routing performance and change public folder access. Additionally, Active Directory sites do not have the concept of costs with one another; therefore, generating efficient connections would be impossible.

The three key issues to keep in mind when designing routing groups are:

- Servers with slow (but resilient) links can be part of the same group.
- Routing groups are dynamic and you can change them at any time.
- Public folder access is determined by the routing group architecture.

Message Encryption

Another consideration to be aware of is the ability to securely transmit messages on the wire. Because Exchange 5.5 uses RPC for transferring messages within the site, and because RPC packets are always encrypted, the packets that make up a message on the wire are not readable. Because SMTP is used as the default transport in Exchange 2000 Server, packet-level security is no longer guaranteed. In perspective, although the packets are non-encrypted by default, the message bodies are MAPI-encoded (if sent by an Outlook client) and not easily readable. However, if your deployment requires packet-level encryption between servers, consider using IPSec in Windows 2000. For more information about IPSec, see the Windows 2000 Server documentation. Another approach is to deploy a Key Management (KM) server and certificates, which encrypt data at the message level. Even in Exchange 5.5, this is the preferred way of securing the content of messages. The advantage of using a KM server and certificates is that the entire message is secure from end to end. This is desirable; however, you'll need to do additional planning to deploy the KM server properly.

Large Organizations

For administrators who have deployed Exchange Server 5.*x* in large, geographically dispersed companies, site boundaries are extremely important. A small number of sites spanning slow links can cause RPC timeouts and MTA thread-blocking; too many sites drastically increases the number of directory replication messages and other background traffic.

Fortunately, the new Exchange 2000 Server directory and messaging architecture removes these concerns. There is no real limit to the number of routing groups that you can create, but you should try to keep the number to under 1,000 for administrative reasons. Remember that more routing groups mean larger link state databases and potentially more status data to replicate; however, this should not seriously affect the performance of the messaging infrastructure. As a guideline, each object in the link state database (routing group, connector, server) requires roughly 32 bytes of memory; therefore, an Exchange organization with 200 routing groups, 250 connectors, and 500 servers requires just over 32 KB of memory on each server to hold the link state database.

Mixed-Vintage Exchange

For maximum coexistence flexibility, you can install an Exchange 2000 server into an existing Exchange 5.*x* site. This allows a company with earlier versions of clients and servers to take advantage of new features in Exchange 2000 Server without having to upgrade the entire installation. Some companies may also see this as a desirable method for upgrading existing Exchange servers to Exchange 2000 Server. For more information, see the white paper, *PT103 – Exchange 2000 Coexistence and Upgrades*.

When an Exchange 5.*x* server must communicate with an Exchange 2000 server (and vice versa) in the same site, X.400 over RPC is used for the message transfer. If two Exchange 2000 servers are present in a mixed-vintage site, they communicate with each other using SMTP. This feature can drastically improve message throughput for those companies that have deployed hub sites spanning slow networks with Exchange 5.5. By upgrading the hub servers to Exchange 2000 Server, the existing spoke site and connector architecture can remain undisturbed while messages are streamed between the hub servers using the fast SMTP protocol rather than RPC.

Connecting Routing Groups

After you define routing group boundaries, you must connect the components of the groups using a connector. As with earlier versions of Exchange, a number of options are available.

Routing Group Connector

A Routing Group connector is the preferred connector for linking routing groups. It uses SMTP as the native transport mechanism and obtains its routing and next-hop information from the link state database. The following table outlines the attributes of Routing Group connectors.

Attribute	***Explanation***
Name	Name of the connector.
Local Bridgeheads	Defines which messaging services (or servers) in the local routing group act as bridgeheads for message transfer.
Remote Bridgeheads	Defines which messaging services (or servers) in the target routing group can be used for message transfer.
Connects this routing group with	Name of the target routing group (drop-down list).
Cost	Logical cost of the connection.
Content Restrictions - Priorities	Specifies whether High, Normal, or Low priority messages can traverse the connector.
Delivery Options - Scheduling	Specifies the active schedule for this connector. Oversize messages can be sent on a different schedule.
Delivery Restrictions - Everyone	Specifies whether messages from Everyone are by default **Accepted** or **Rejected**.
Delivery Restrictions - Accept	Lists the recipients that can always use the connector.
Delivery Restrictions - Reject	Lists the recipients that can never use the connector.

A Routing Group connector is unidirectional, so you must use two Routing Group connectors to form a bidirectional link between two routing groups. After a Routing Group connector is created, System Manager can automatically configure the adjacent Routing Group connector for you.

Although a Routing Group connector is described as using Simple Message Transfer Protocol (SMTP) to transfer messages, technically a Routing Group connector is transport protocol–independent. A Routing Group connector is the equivalent of a Site Connector in Exchange 5.5; therefore, during an upgrade of a 5.5 bridgehead server that supports Site Connectors, these connectors are upgraded to Routing Group connectors that communicate using remote procedure call (RPC).

Note In the Exchange 2000 Server Beta 3 release, messages are routed to servers over Routing Group connectors by their NetBIOS names. This will change to the fully-qualified domain name (FQDN) of the server in subsequent releases of Exchange 2000 Server.

SMTP Connector

An SMTP connector is analogous to the Internet Mail Service in earlier versions of Exchange. It can be deployed for use between Exchange and other SMTP-compatible messaging systems, such as UNIX sendmail or other SMTP hosts on the Internet. The major differences between a Routing Group connector and an SMTP connector are:

- The SMTP connector uses a smart host or mail exchanger (MX) records in DNS for next-hop routing (although they do relay link state information when configured between routing groups in the same Exchange organization).
- You can set custom authentication and encryption for the connection.
- You can deploy an SMTP connector between two independent Exchange organizations (that is, Active Directory forests).

The following table outlines the attributes of SMTP connectors.

Attribute	***Explanation***
Name	Connector name.
Local Bridgeheads	Defines which messaging services (or servers) in the local routing group act as bridgeheads for message transfer.
DNS/Smart Host	Specifies the name (IP address or DNS name) of the next-hop or notifies the server to use DNS and MX records for delivery.
Address Space	The SMTP domains that this connector can reach. Each entry has an associated cost. The scope of the connector can also be configured so the address spaces are restricted to the local routing group.
Connected Domains	Names of other routing groups that can be reached through this connection.
Content Restrictions - Priorities	Specifies whether High, Normal, or Low priority messages can traverse the connector.
Delivery Options - Scheduling	Specifies the active schedule for this connector. Oversize messages can be sent on a different schedule.
Delivery Restrictions - Everyone	Specifies whether messages from Everyone are by default Accepted or Rejected**.**
Delivery Restrictions - Accept	Lists the recipients that can always use the connector.
Delivery Restrictions - Reject	Lists the recipients that can never use the connector.
Advanced – ETRN/TURN options	Specifies whether this server should issue an ETRN or TURN command when connecting to remote servers.
Advanced – HELO or EHLO	Specifies whether a HELO command is sent instead of a EHLO command when making an outbound connection. This can be changed based upon the interoperability required with the SMTP host.
Advanced – Outbound Security	If an ELHO command is sent, this specifies the authentication required for outbound connections. Possible options are Anonymous, Basic authentication, or Windows security package**.**
Advanced – Security - TLS	Specifies whether Transport Layer Security (TLS) should be applied.
Advanced - ATRN	Lists the user accounts that are allowed to use the ATRN command.

X.400 Connector

The X.400 connector is provided for three reasons:

- Connectivity to other X.400 MTAs

- Connectivity to an X.400 service provider
- Connecting two routing groups

If you are familiar with the X.400 connector in earlier versions of Exchange, the support for X.400 in Exchange 2000 Server will be familiar to you. There have been some architectural changes, such as the switch to Lightweight Directory Access Protocol (LDAP) directory lookups instead of XDS, and RFC 2156 MIME Internet X.400 Enhanced Relay support has been implemented to allow full X.400 and SMTP interoperability. Note, however, that X.400 connectors over the TP4 protocol are no longer supported because Windows 2000 does not support TP4.

You can use the X.400 connector to link routing groups. This is desirable if only X.400 connectivity exists, or if the network link is unreliable or extremely short of bandwidth. The X.400 MTA allows graceful recovery of associations when there are transient problems on the network. However, the X.400 protocol carries an enormous amount of 'baggage' with it and enforces strict handshaking and acknowledgement rules. Because of the simplicity of SMTP, you'll find that for general message communications, SMTP is a faster transport mechanism.

When you use an X.400 connector between routing groups, link state information is passed between MTAs by sending a message blob before user interpersonal messages (IPMs) are transferred. However, unlike a routing group connector, you can define only a single host for the local and remote bridgeheads. This means that load balancing and bridgehead fault-tolerance can be achieved only if you have multiple X.400 connectors in place.

Non-Connected Networks

When an intermittent connection or no direct connectivity exists between routing groups, you still need to deploy a connector to enable messaging. A prime example is a situation in which asynchronous dial-up through a modem is the only form of connectivity between locations. Unlike earlier versions of Exchange, Exchange 2000 Server does not include a Dynamic RAS Connector. Instead, you can achieve more efficient routing by using one of the connectors Exchange supplies (Routing Group connector, SMTP connector, X.400 connector) over an on-demand connection that the operating system supplies. This takes advantage of the *Routing and Remote Access* components that Windows 2000 offers.

SMTP Performance and Message Payloads

Some people may be concerned about the performance of an SMTP transport over X.400 as implemented in earlier versions of Exchange. A quick test with Exchange Server 5.5 shows that in some circumstances message data transported in SMTP or Multipurpose Internet Mail Extensions (MIME) format can be larger than the same data transported over RPC or X.400 over TCP/IP.

Table 1.1 - A simple plain text message. Traffic includes binding and authentication of the MTAs

Connector	*Traffic Sent*	*Traffic Received*	*Total Frames*
RPC	7,484 bytes	3,889 bytes	60
X.400	4,087 bytes	751 bytes	19
Internet Mail Service	6,962 bytes	1,376 bytes	30

Table 1.2 - A simple plain text message. MTAs already bound

Connector	*Traffic Sent*	*Traffic Received*	*Total Frames*
RPC	4,744 bytes	1,434 bytes	24

X.400	3,653 bytes	319 bytes	13
Internet Mail Service	6,962 bytes	1,376 bytes	30

Table 1.3 - A message with a 304Kb attachment. MTAs already bound

Connector	*Traffic Sent*	*Traffic Received*	*Total Frames*
RPC	342,302 bytes	19,629 bytes	471
X.400	332,366 bytes	12,279 bytes	454
Internet Mail Service	453,164 bytes	13,258 bytes	523

The SMTP protocol that Exchange 2000 Server uses is more sophisticated than the protocols in Exchange 5.5. When configuring the Internet Mail Service between two Exchange 5.5 servers, all messages are sent in Base64 encoding. This means that 8-bit data must be converted to 7-bit data, which causes a payload overhead. With Exchange 2000 Server, when the SMTP client and server are both running Exchange 2000 Server, data is transferred in 8-bit format (verb: 8BITMIME); therefore, no size penalties are incurred. Tests have proven that Internet Mail Service in Exchange Server 5.5 can perform better than the X.400 connector by as much as 300 percent, mainly because both RPC and X.400 transports rely on stringent call setups, handshaking, and acknowledgements. It is true to say that SMTP doesn't carry as much overhead as these other protocols. With the additional performance improvements made in the Exchange 2000 Server SMTP stack, such as CHUNKING and PIPELINING (for more information, see Appendix A), this positive performance gap between SMTP and X.400 increases even further.

Routing Group Connection Arrangements

Now that you understand routing group boundaries and the type of connectors that you can use to connect routing groups, you can decide on a strategy for connecting multiple routing groups in an organization. The two most common methods are hub and spoke and mesh.

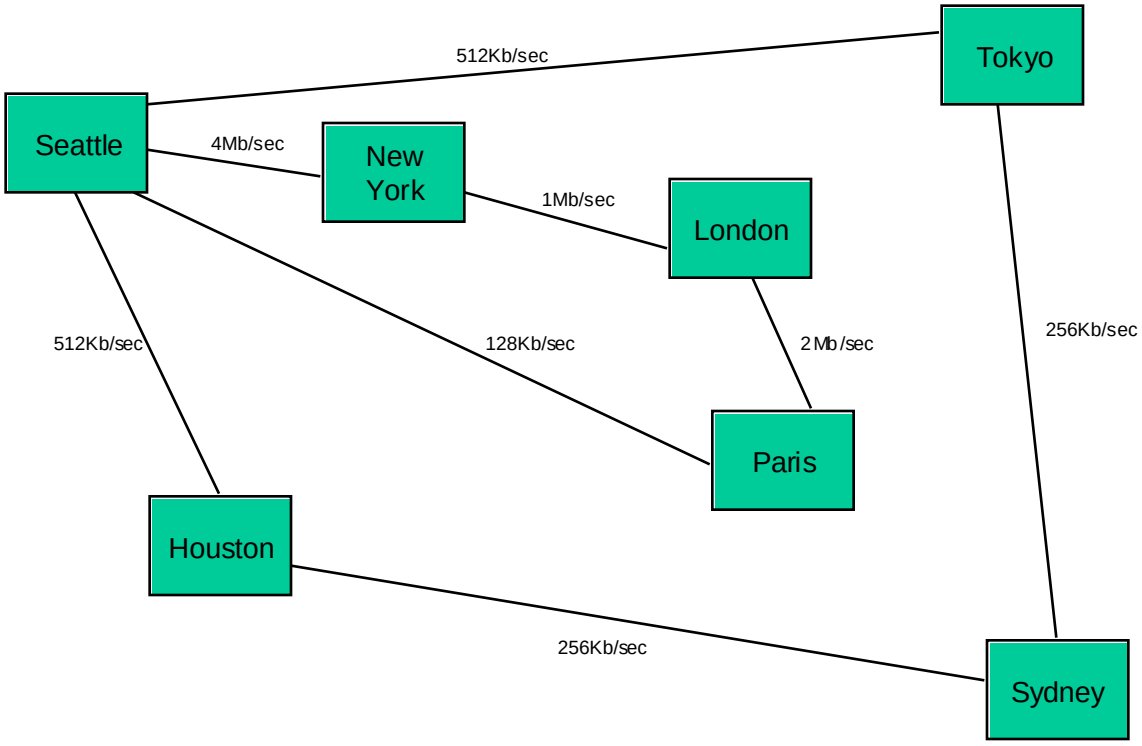
A careful examination of the network is necessary so that connectivity between routing groups can follow the underlying infrastructure as much as possible. In general, you do not want a point-to-point connection to travel over many network routers, because this incurs latency on the connection and can cause time-outs. Additionally, you shouldn't route messages to a hub routing group that's far away, if the destination is local to the source.

In earlier versions of Exchange, many site deployments were connected in a hub and spoke arrangement for scalability and to reduce directory replication latency. Additionally, many designers refrained from implementing shortcut or redundant routes between spokes because this created message *ping-pong* issues when network connectivity failed. Although you can configure a hub and spoke routing group arrangement in Exchange 2000 Server, many of the deployment issues found in earlier versions of Exchange have been eliminated, mainly because of the new the directory service architecture and the implementation of link state data.

Routing Group Deployment Scenario

The following example walks you through the design of routing group architecture, connectivity choice, and connection arrangements.

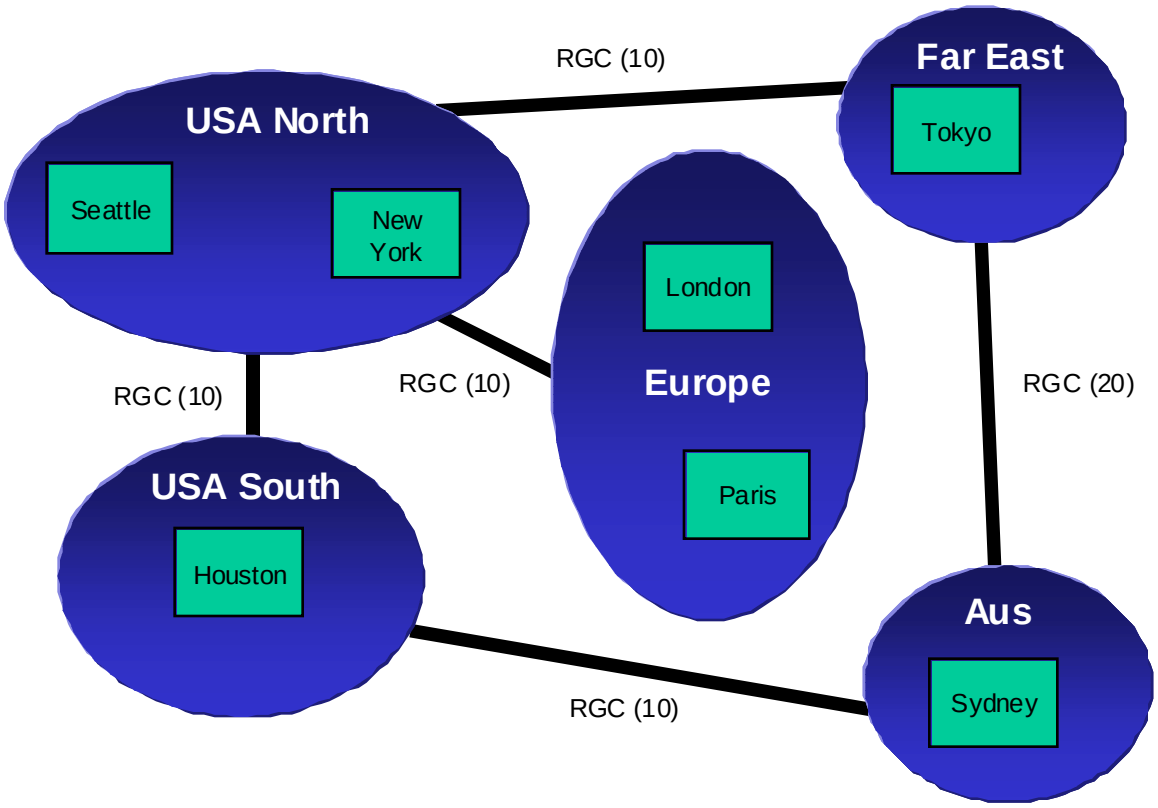
The following diagram shows the physical location for users in the company and the network links deployed.



Although there appears to be a high-bandwidth link between two locations, you also need to consider other traffic that may be present on the line when calculating the net bandwidth available. In addition, you need to ask the following questions:

- Are there certain groups of users that send large messages to one another on a regular basis?
- What's the average size of message that travels across the network?
- Which public folders do users access?

Many of the design philosophies you used when designing site architecture for Exchange 5.*x* can be reused for Exchange 2000 Server. After you perform the analysis and understand the business requirements, you can lay out the boundaries.



As you can see from the final strategy, you can group multiple locations together to form single routing groups, although you must understand the impact this has on client connections to public folders. In the previous illustration, although USA North and USA South belong to the same routing group, this can result in slow public folder access.

Routing Group connectors are used in this scenario because they offer the best functionality and resilience. Each Routing Group connector uses the concept of *source* and *target* bridgehead servers. To get the maximum efficiency from the network, the bridgeheads are configured so that all connections occur over a single network link; for example, the target bridgehead servers specified in the connector from USA South to USA North only include connector servers in Seattle, because no direct network connectivity exists between Houston and New York. Note that unlike the Site Connector in Exchange 5.5, multiple target bridgehead servers specified on a connector are not used in a cost-weighted mechanism, nor are they used on a round-robin basis. Essentially, when message transfer takes place, a local bridgehead server reads the list of target servers and chooses the first one on the list. If the first server is down, the bridgehead server uses the second server, and so on. Subsequent messages use the same algorithm.

You can implement different messaging costs to dictate the primary messaging path. In the previous scenario, the Far East and Aus routing groups have a higher cost link between them, so messages between London and Sydney go over the more reliable network link through Houston.

Under normal conditions in which multiple routes exist with the same cost, Exchange 2000 Server uses only one of those routes. Exchange uses the secondary route only if the first route fails. A similar design with Exchange Server 5.5 uses the same cost connections on a round-robin basis, which makes management difficult.

Link State Information

Message Routing Concepts

All enterprise messaging systems require a way of way of passing routing data to one another. With Exchange 2000 Server, when you place a connector between two routing groups, a cost is associated with the link. A cost is not expressed in monetary terms but determines preference when multiple routes are available; the lower the cost, the more preferable the route.

Exchange 2000 Server builds on the rich routing architecture in Exchange Server 5.5, which makes routes and costs in the organization available to any messaging server. Earlier versions of Exchange use the concept of a routing calculation server; in each Exchange site, one server collects the available routes and costs from the directory and forms a GWART. The GWART is propagated to all servers in the Exchange site so that they have information about the routing topology in the organization. However, this routing architecture cannot detect routing problems in the downstream network. If a network link or bridgehead server fails, no mechanism exists for communicating this information to the rest of the Exchange organization.

Introduction to Link State

Although Exchange 2000 Server uses routes and costs, a new link propagation protocol has been implemented, which is called the *Link State Algorithm*. This is based upon Dijkstra's algorithm from 1959 and has been used extensively on the Internet for many years in the form of Open Shortest Path First (OSPF) routers. The Link State Algorithm propagates the state of the messaging system in almost real time to all servers within the organization. This has the following advantages:

- Each Exchange server can make the best routing decision at the source instead of sending messages down a path in which a downstream link may be down.
- Message ping-pong between servers is eliminated, because each Exchange 2000 server has information about whether alternate or redundant links are up or down.
- It eliminates message looping problems.

This architecture is extensible, and it is possible that future versions of this architecture will be able to use the data stored in Active Directory through hardware such as network routers; this is known collectively as *Directory Enabled Networks*.

How Link State Works

Link state information is most effective when you configure multiple routing groups in the organization, particularly when redundant paths are available. Each routing group has a nominated master receives link state information from different sources. The master keeps track of this data and propagates it to the rest of the servers within the routing group. The master ensures that all servers in the routing group advertise the same information to the rest of the world; therefore, the routing group master increments the version number of the routing database for its group. The master is normally the first server to be installed in the routing group, but you can specify a different routing group master using System Manager. The routing group master is the RID.

There are differences in the way that link state data is propagated between routing groups and within the routing group. Between groups, new information is relayed through SMTP on port 25 and is sent between servers when a change takes place. Within a routing group, link state information is sent to and received from the master on TCP port 3044 (to be changed to port 691 in Release Candidate 1 and later). When a non-master server receives new link state information, the server immediately transfers it to the master server so that other servers can receive information about the routing change. There are only two states for any given link, up or down, so connection information, such as whether a link is active or in a retry state, is not propagated and is known only on the server involved in the message transfer.

All link state data is held in the memory of the Exchange 2000 servers; the data is not stored on a disk. If one of the servers is recycled (including the master), the running servers redistribute the information. The definitions of connectors and costs are not held in the link state database but are read from Active Directory on startup. The link state database refers to the connection by its globally unique identifier (GUID).

Link State Updates

When a bridgehead server detects a link is unavailable, it tags that connector as DOWN and propagates the data to the routing group master. The master immediately notifies the other servers in the routing group of this change and propagates this information to other routing groups. The following is an example of a network monitor trace of a link state update between a member and a master.

```
00000030          7B 30 30 30 30 30 30 35 31 7D    {00000051}
00000040 20 56 53 5F 43 4F 4E 4E 20 37 66 65 31 32 65 66 .VS_CONN.7fe12ef
00000050 65 65 38 66 31 36 35 34 64 62 63 32 37 31 37 35 ee8f1654dbc27175
00000060 38 35 35 64 37 66 37 64 31 20 63 33 38 30 61 62 855d7f7d1.c380ab
00000070 30 66 37 62 37 62 66 66 34 33 38 65 66 62 38 37 0f7b7bff438efb87
00000080 30 37 35 36 38 63 66 38 62 39 20 44 4F 57 4E 20 07568cf8b9.DOWN.
00000090 20
```

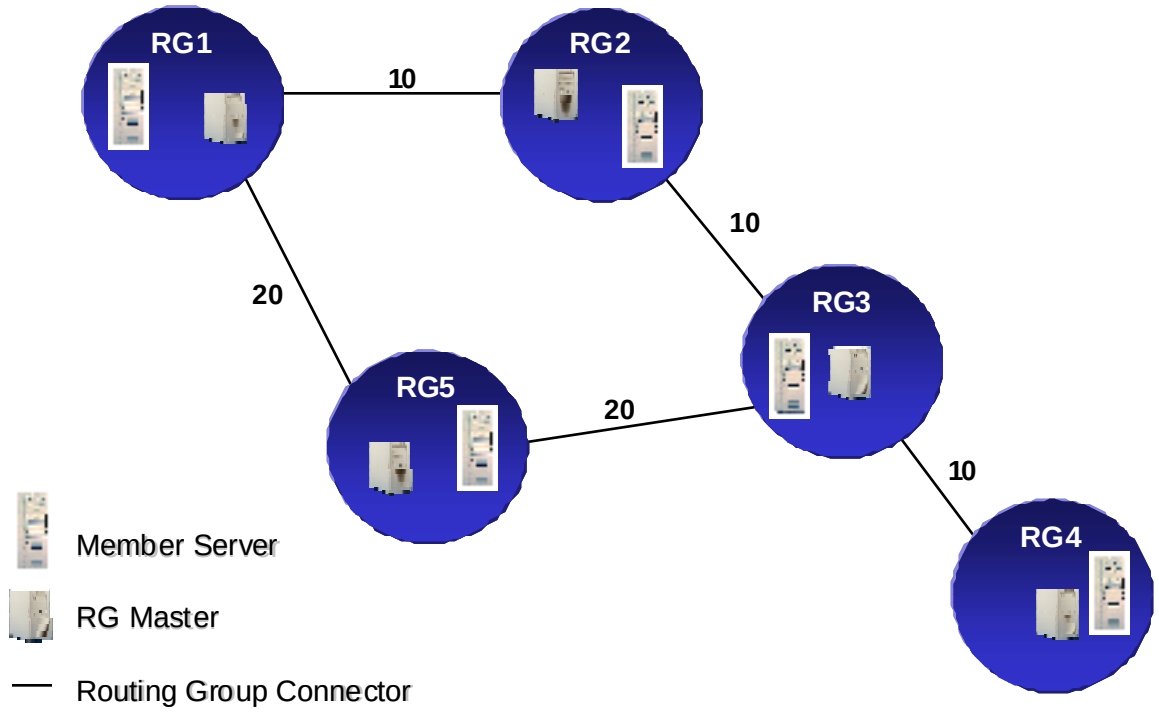
The information contained in this trace breaks down as follows:

- The number in parentheses refers to the number of bytes (in hexadecimals) remaining in this packet.
- The first hexadecimal number (7FE12….) relates to the GUID of the connector affected.
- The second hexadecimal number (C380A….) relates to the GUID of the virtual server that detected the change.
- The final word (DOWN) sets the status of the connector.

As stated earlier, data is transferred between routing groups over SMTP on port 25. The format of the data is roughly similar to that of the intra-routing group communications. However, if you are performing a network trace for link state data, you will notice the X-LINK2STATE command verb denotes the type of data, and the information is sent in chunks (FIRST CHUNK, SECOND CHUNK, LAST CHUNK).

Link State Walkthrough

The best way to gain an understanding of how link state works and what happens to messages in the event of a failure is to examine a scenario. Imagine the routing group configuration in the following diagram and assume that Routing Group connectors are deployed.



Normal Message Flow

As you can see from the cost of the connections, a message sent from RG1 to RG4 will hop by means of RG2 and RG3.

Single In-Line Failure

Assume that the network fails between RG2 and RG3. Unless messages are waiting to be transferred over this link, the failure is not detected immediately. If a user in RG1 sends a message to another user in RG4, the routing process take places as follows:

1. The bridgehead in RG1 sends the message to a target bridgehead in RG2.
2. Through a call to routing, RG2 attempts to open a SMTP connection to a target bridgehead in RG3.
3. If there are multiple target bridgeheads specified on the connector to RG3, the local bridgehead in RG2 attempts to open a connection in sequential order in case a single bridgehead fails.
4. If none of the bridgeheads in RG3 can be contacted, the connection goes into a GLITCH-RETRY state. The connection waits for 60 seconds and then retries the transfer.
5. If, after three reconnection attempts the link still does not operate, the connection is marked as DOWN and a call to routing is made to reroute the messages waiting in the queue.
6. The bridgehead in RG2 connects to port 3044 of the RG Master and sends a link DOWN notification.
7. The RG Master in RG2 immediately floods this data to all other Exchange 2000 servers in the routing group.
8. The bridgehead in RG2 calculates that an alternate route is available to RG4 by means of RG1, RG5, and RG3.
9. Before the messages are routed back through RG1, link DOWN information is sent to the bridgehead in RG1. The communication takes place by using the X-LINK2STATE command verb after issuing an EHLO command.

10. The bridgehead in RG1 immediately connects to the RG Master in RG1 through port 3044 and transfers the link DOWN information.
11. The RG Master in RG1 immediately floods this data to all other Exchange 2000 servers in the routing group.
12. Using the new link information, the bridgehead in RG1 calculates that the best route to RG4 is by means of RG5.
13. Before the message is routed to the bridgehead in RG5, the link state information is propagated to the bridgehead in RG5. There is a possibility that RG5 has already determined that the link is down if messages were already routed from RG3 to RG2.
14. RG5 continues the process, routing the messages by means of RG3 and on to the destination, RG4.

Returning a Link to UP Status

Although messages are now flowing through the alternate route, this link may be more costly, and the servers must be notified when the original link is available again. After a link is tagged as DOWN, the original bridgehead continues to retry the connection at 60-second intervals. Although no messages are awaiting transfer, the retry is simply an attempt to open port 25 on the destination server. Once a good connection is established, the bridgehead notifies the local RG master that the connection is available again.

Multiple In-Line Failure

Perhaps a more interesting scenario is one in which multiple link failures occur. With earlier versions of Exchange, the message ping-pongs between the links attempting to find an open connection. After 512 loops, an NDR is generated. However, because of link states, Exchange 2000 Server is more efficient; for example, take the previous scenario and assume that the network link between RG5 and RG3 also goes down.

1. The bridgehead in RG5 attempts to open the connection to a target bridgehead in RG3 and fails.
2. The connection enters the GLITCH-RETRY state and retries for three times at 60-second intervals.
3. If a connection still cannot be established, the link is tagged as DOWN and the RG master is notified (the routing group master in turn floods the state to the other servers within the routing group).
4. A call is made to routing on the bridgehead server, which calculates that all available routes to RG4 are down; therefore, the cost of the connection is implicitly INFINITE.
5. The messages remain in the queue; a further call to routing is made every 60 seconds to see if any links are available.
6. If a link becomes available, messages are rerouted as appropriate. If the messages are still in the queue after 48 hours, they are returned as NDRs to the senders in RG1.

Subsequent Messages

Another interesting facet of link state information is what happens to subsequent messages that are sent once a link is tagged as down. In the scenario with a single in-line failure, other messages addressed to users in RG4 are immediately routed through the alternate route because each server in the organization has information that the primary route is down. In the scenario with a multiple in-line failure, new messages submitted to RG1 stay in the queues on RG1. This is the most appropriate place to queue the message because either of the links could come back up first.

Routing Group Master Failure

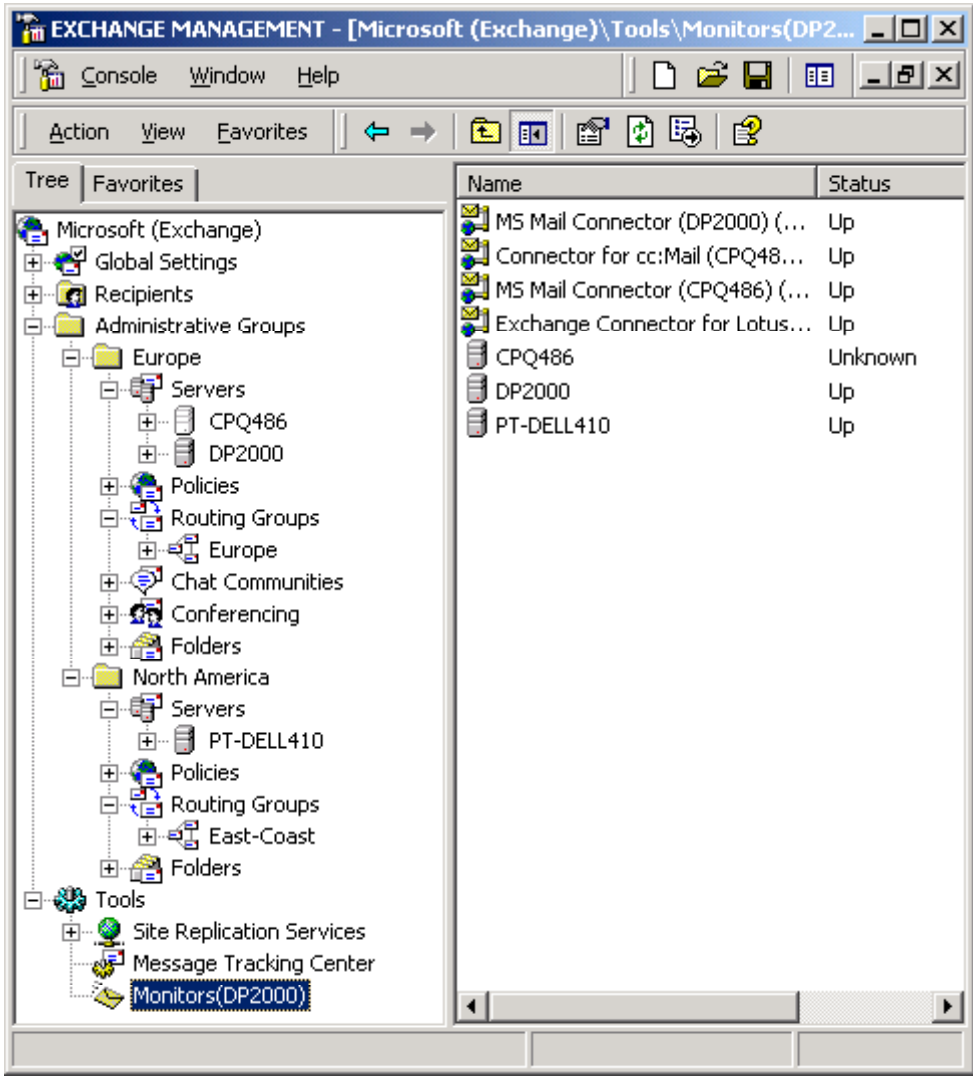
If a routing group master fails or is taken offline, bridgehead servers still accept and act on new link state data that they receive, ensuring that loop-free messaging is still in place. While the master is offline, there is a chance that a message may be sent over a link in which the downstream connection

is not functioning. This means that the routing is not optimal while the master is down; however, this configuration still does not allow message loops.

If a master is offline for more than a few minutes, you may want to intervene and select a new server to become the master. This is a very simple process, which you can accomplish using System Manager.

Viewing the Status of a Link

You can view the link state database in System Manager by navigating to Tools, Monitoring and Status, and then the Status node.



EDK Gateways and Connectors

Link state information is most effective when used for efficiently routing messages between Exchange 2000 servers. Although Exchange Development Kit (EDK) Gateways and Connectors such as the Lotus cc:Mail Connector, MS Mail Connector, Lotus Notes Connector, and Novell GroupWise Connector display their link state information, their status is always shown as UP.

Transport Management

Performance Monitoring

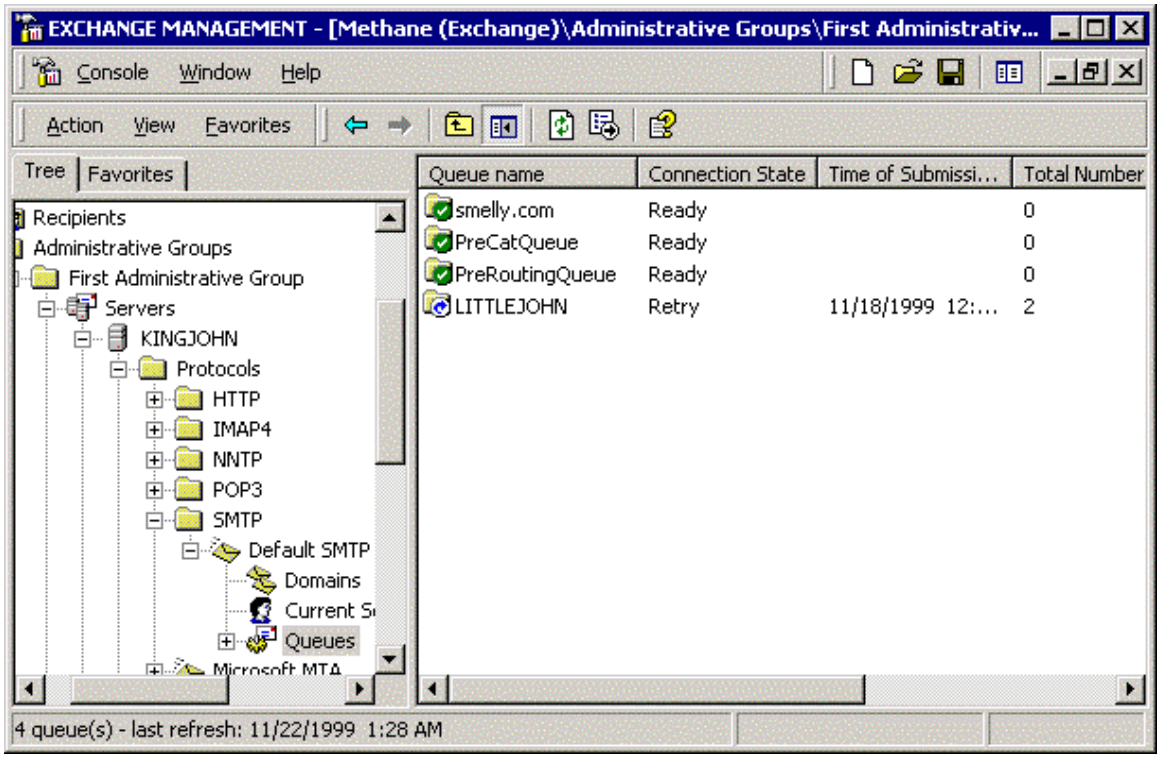
As in earlier versions of Exchange, the most effective way to calculate performance and identify transport issues is to watch the queue levels in Microsoft Windows NT Performance Monitor. This mechanism uses the least intrusive and most resource-efficient method for detecting problems.

Queue Viewing

Once you recognize that an issue exists with a queue, you may want to look into the virtual server to see the messages awaiting transfer. There may be a large message in the queue that is clogging the transport, or an application may be sending hundreds of messages in error.

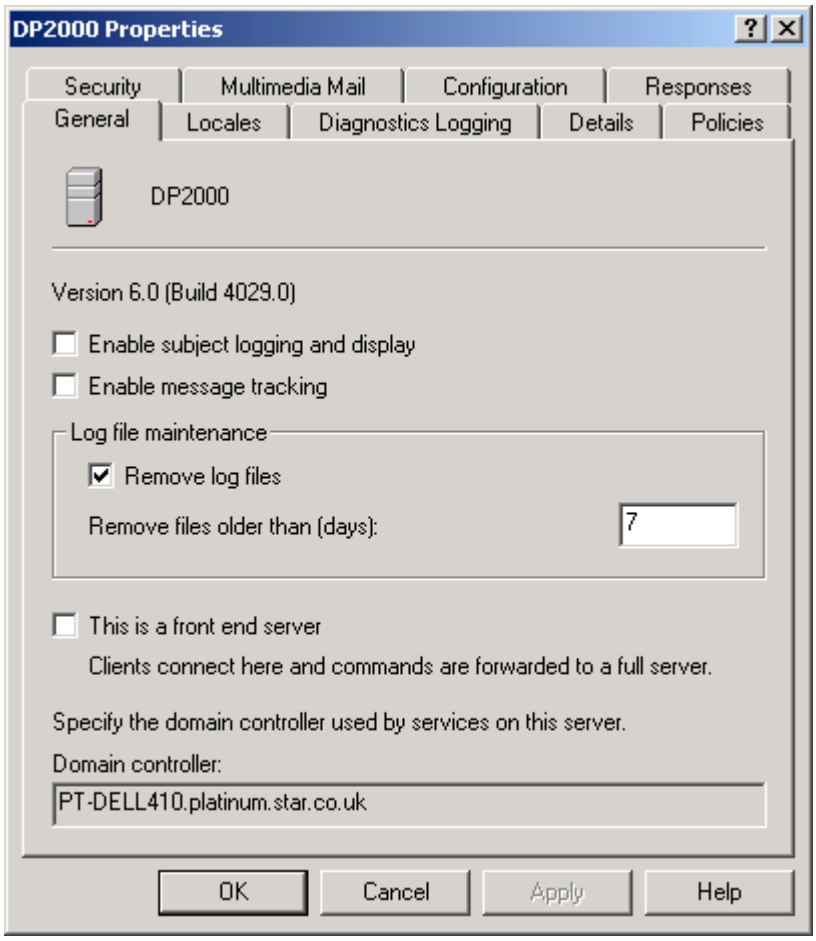
The queue viewer is greatly improved over earlier versions. Not only can you enumerate messages in the queue by a filter, but you can see the sender, receiver, and subject of the message. If a particular message is causing a problem, you may decide to return it to the sender with an NDR. If an entire queue is causing a problem, you can freeze it without affecting the other messages in the system.

The queues that System Manager shows are *link queues* (they show the next hop in the message path). These queues are dynamically created and cleaned up as necessary. If the next hop is unreachable, the queue state and icon change to reflect this.



Message Tracking

By default, a message can pass through the system without leaving a trace. You may decide that for management, monitoring, or billing reasons you need to enable message tracking. In this configuration, each message is tracked as it is submitted to the server, passes between components, and is transferred across the network. All tracking data is held in a set of plain-text log files on each server. This functionality is very similar to that in earlier versions of Exchange; however, now you have the option of logging the subject of each message.


DP2000 Properties
Security
Multimedia Mail
Configuration
Responses
General
Locales
Diagnostics Logging
Details
Policies
DP2000
Version 6.0 (Build 4029.0)
Enable subject logging and display
Enable message tracking
Log file maintenance
Remove log files
Remove files older than (days):
7
This is a front end server
Clients connect here and commands are forwarded to a full server.
Specify the domain controller used by services on this server.
Domain controller:
PT-DELL410.platinum.star.co.uk
OK
Cancel
Apply
Help

Appendix A: Understanding Advanced SMTP Command Verbs

The Exchange 2000 Server Simple Mail Transfer Protocol (SMTP) virtual server supports the advanced functionality required to make it as (or in some cases, more) functional and perform as well as the X.400 message transfer agent (MTA). All of the performance extensions that Exchange 2000 Server uses are standards covered by the Internet Engineering Task Force (IETF).

Chunking

SMTP as defined under Request for Comments (RFC) 821 specifies that the DATA command is issued by the SMTP client to signify the start of the actual message data representation. This standard also specifies that the end of the data is signified by the sending of a carriage return – line feed, full stop, carriage return – line feed. This is a very slow and inefficient way of sending the data chunk, because the SMTP host must continually scan for the end of the data representation. The majority of SMTP servers on the Internet, including Exchange Server 5.5, support only this method of sending the data.

To overcome this performance bottleneck, Exchange 2000 Server implements the BDAT command from the CHUNKING EMSTP specification, as defined under RFC 1830 (http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/18xx/1830). Essentially, this replaces the standard DATA command verb with BDAT and an argument. The argument specifies the expected number of bytes in the message chunk that the server (that is, the receiver) should expect. The server now needs to count the number of bytes received from the wire; when this number equals the value given in the BDAT argument, the server assumes it has received the entire message.

An Exchange 2000 server advertises that it supports the BDAT extension when a client sends an EHLO command. Similarly, when an Exchange 2000 server is acting as an SMTP client, uses chunking if the server advertises it.

Pipelining

The RFC 821 SMTP standard defines that for each SMTP command issued, an acknowledgment is sent. For many of the commands, this is normally the **250 OK** acknowledgement, which means that the last command was successful. As SMTP implementations have become more reliable and many of the interoperability problems have been eliminated, the overhead of waiting for a command acknowledgement can slow down the overall performance of the message transfer. This problem is particularly noticeable on high latency networks.

To overcome this performance bottleneck, Exchange 2000 Server implements pipelining as defined under RFC 2197 (http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/21xx/2197). This allows multiple commands (such as MAIL FROM, RCPT TO) to be streamed from the SMTP client to the host without waiting for an acknowledgement.