Microsoft® Exchange 2000 Server

Microsoft® Exchange 2000 Front-End Server and SMTP Gateway Hardware Scalability Guide

White Paper

Published: November 2000

Copyright

Table of Contents

Microsoft Exchange 2000 Front-End Server and SMTP Gateway Hardware Scalability Guide

White Paper

Published: November 2000

For the latest information, please see http://www.microsoft.com/exchange/

Introduction

This document provides hardware recommendations for Microsoft® Exchange 2000 Server front-end servers and Simple Mail Transfer Protocol (SMTP) gateways (also known as hubs). The recommendations in this document can help you build a highly scalable and reliable messaging system, customized to the needs of your organization.

A front-end server is a server that receives requests from clients and relays them to the appropriate back-end server. A back-end server is a server that hosts at least one database to which front-end servers connect when relaying requests from clients. SMTP gateways are servers that route electronic-mail messages through an organization or the Internet to their final destination.

There are many factors that affect the performance of a Exchange 2000 Server front-end server. These factors include the protocols being used, number of installed processors, amount of available memory, amount of network traffic anticipated, use of secure authentication, and use of Secure Sockets Layer (SSL) to encrypt network traffic. Such factors must be taken into account before choosing the correct hardware for a specific Exchange 2000 application.

POP3

Post Office Protocol version 3 (POP3) is an Internet protocol that allows a POP3 client to download e-mail from a server. This protocol works well for computers that are unable to maintain a continuous connection to a server. If you are running POP3 on a dedicated front-end server, and are trying to determine your hardware needs, you should consider the following factors:

- Processor scalability
- Memory needs

- Disk usage
- Expected network traffic
- Implications of using Secure Authentication
- Implications of using SSL encryption

The following sections discuss these factors and contain hardware recommendations. The following front-end server scenario will be used in this section to discuss hardware needs when running a front-end server dedicated to POP3 client requests.

Hardware: Pentium III 400-megahertz (MHz) front-end server with 1 gigabyte (GB) RAM. Two 100-megabit per second (Mbps) network cards for the front-end server. Each back-end server is configured with four 500-MHz Xeon processors, 1.3 GB of RAM, and 20 disk spindles. The number of physical hard drives attached to the server is commonly referred to as the number of disk spindles. Typically, these hard drives will be attached in a RAID (redundant array of inexpensive disks) configuration for performance or data redundancy purposes. Each disk spindle is capable of independently reading from or writing to disk; therefore, a large number of disk spindles will increase the number of simultaneous disk operations that can be performed by the server.

Scenario: The average size of messages being transferred from the front-end server to POP3 clients is 50 kilobytes (KB). The front-end server acts as a proxy between a POP3 client and the appropriate back-end server. The message traffic generated in this scenario uses 80 percent of the processor resources on each back-end server. The number of SMTP messages being delivered to mailboxes each second equals the number of messages retrieved and deleted via POP3 each second. The POP3 commands associated with message retrieval and deletion are RETR and DELE. The POP3 command issued by a POP3 client to determine the number of messages in a mailbox is STAT. Statistical counters for RETR, DELE, and STAT can be used to determine the number of POP3 transactions occurring on the front-end server each second. These counters can be accessed through the Performance Monitor application, which is included with the Microsoft Windows® 2000 operating system. In this scenario, the number of STAT commands received by the POP3 server is equal to twice the number of RETR and DELE commands. This profile is typical of an Internet service provider (ISP) environment in which one message is received and deleted for every two POP3 logins. The Exchange Stress and Performance (ESP) tool, which is available in the *Microsoft Exchange 2000 Server Resource Kit*, was used to generate this server load. The actual POP3 ESP script is included in the Appendix of this document.

The following data was obtained from an Exchange 2000 front-end server with two 400 MHz processors. The table reveals the variance in hardware usage when using a range of back-end servers with the given profile. Note that context switching on the front-end server does not grow significantly as processor usage increases. Context switching refers to a change made by the operating system in

which a previously inactive thread becomes active, or the opposite. See the Appendix for an overview of the significance of context switching activity.

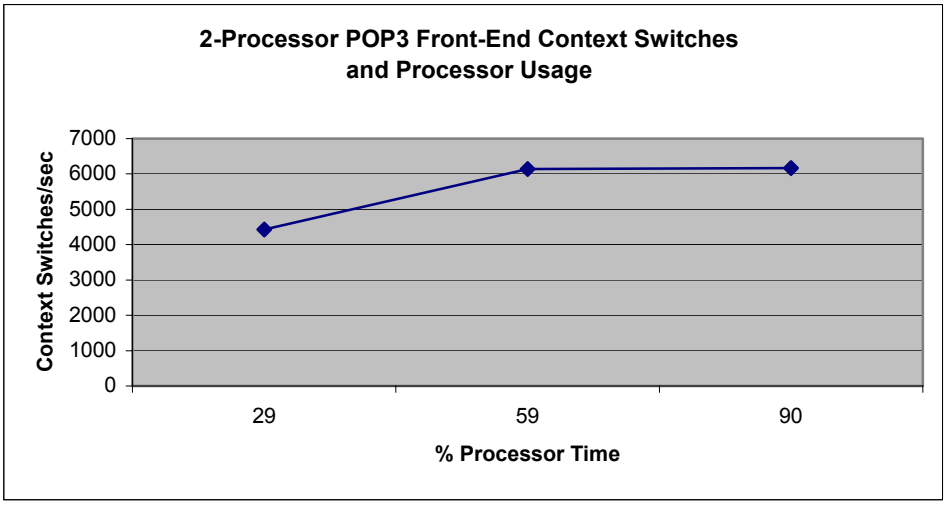
Back-End Servers	% Processor	Context Switches/Sec	DELE/Sec	STAT/Sec	Network Traffic	InetInfo Working Set (MB)
1 Back-end server	29	4429	23	53	1.5 MB/sec	105
2 Back-end servers	59	6141	44	95	3 MB/sec	105
3 Back-end servers	90	6169	65	120	4.5 MB/sec	105

For comparison purposes, the following data was obtained from an Exchange 2000 front-end server with four 400 MHz processors. In this configuration, context switching continues to increase significantly as processor usage increases on the front-end server. The significance of this finding in relation to processor scalability is discussed below.

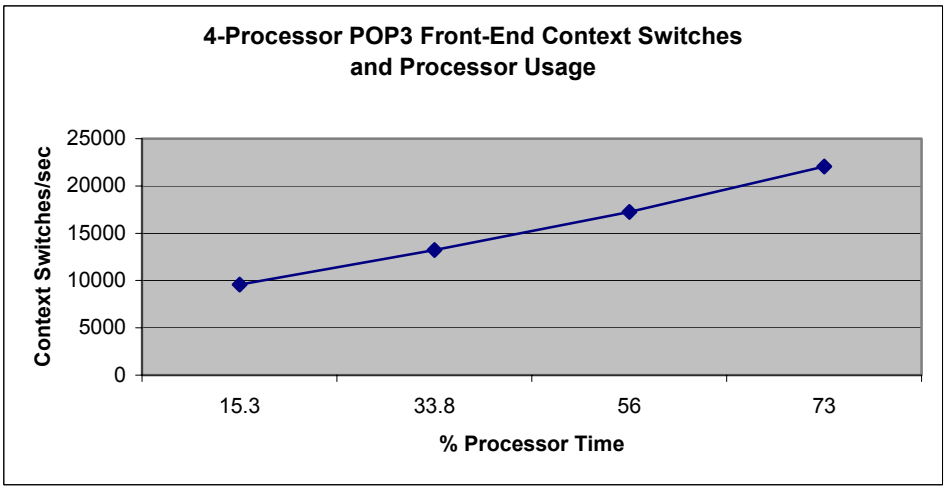
Back-end Servers	% Processor	Context Switches/sec	DELE/sec	STAT/sec	Network Traffic	InetInfo Working Set (MB)
1 Back-end server	15.3	9579	23	53	1.5 MB/sec	105
2 Back-end servers	33.8	13217	44	100	3 MB/sec	105
3 Back-end servers	56	17248	58	127	4.5 MB/sec	105
4 Back-end servers	73	22047	65	153	5.6 MB/sec	105

Processor Scalability

Processor scalability should be considered when determining your hardware needs. Processor scalability refers to efficient usage of available processing resources. POP3 exhibits very good 2-processor scalability, based on a low number of context switches while under heavy processor load. On a 4-processor front-end server, POP3 begins to context switch excessively at about 50 percent CPU usage. You should use a front-end server with no more than 2 processors for POP3 clients. As discussed in the Appendix, a large amount of context switching is usually indicative of too many active threads competing for the processors. The following chart demonstrates how context switching begins to level off on a 2-processor front-end server as processor usage increases.



If you are running a 4-processor front-end server, context switches continue to increase to excessive levels as processor usage increases. The following figure illustrates this scenario.



Although a 4-processor server may allow you to serve several additional back-end servers simultaneously, a large percentage of CPU time will be wasted by context switching activity. For example, if you are running a front-end server with two 400 MHz processors and back-end servers with 500 MHz processors, the ratio of back-end servers to each front-end server for POP3 is about 5 to 1. Using 4-processor front-end servers, this ratio may drop to 4 to 1 because of excessive context switching.

Memory

POP3 front-end servers require very little memory to operate efficiently. As the number of simultaneous POP3 sessions on the POP3 front-end server increases,

memory usage does not increase significantly. This is because POP3 clients do not usually maintain long connections to the front-end server, so the amount of memory used to run POP3 is relatively small. The MSExchangeIS (Store.exe) service can be disabled on POP3 front-end servers, yielding additional memory savings. If this service is disabled, a POP3 front-end server can run very comfortably with 256 MB of RAM. For additional information on disabling the MSExchangeIS service, as well as details on other service dependencies, see the "Exchange 2000 Front-end and Back-end Topology" white paper at http://www.microsoft.com/exchange/.

Disk Usage

When determining your hardware needs for a dedicated POP3 front-end server, be sure to consider your expected amount of disk usage. A POP3 front-end server will use its hard disk very rarely. This is because front-end servers act as proxies, passing each protocol session to the appropriate back-end server. This eliminates the need to store any user-specific data such as mailboxes on the front-end server. If protocol logging is enabled in Exchange System Manager for a POP3 virtual server, then the hard disk will be used on the front-end server to store the requested protocol log. The disk will also be used by the Windows 2000 cache manager to page information to and from the paging file. The paging file is used by the cache manager to temporarily store information from RAM that has not been accessed recently when additional memory is needed by active system processes. Paging activity can be minimized by ensuring that a sufficient amount of RAM is available on the server. A POP3 front-end server with 256 or more MB of physical memory will rarely page. One disk spindle for a POP3 front-end server should be sufficient for most applications. If you are running large servers with protocol logging enabled, you may want to consider adding a second spindle.

Network Usage

The amount of network traffic on your POP3 front-end server should be considered when you are trying to determine the type of hardware you want to use on this server. Because a POP3 front-end server can service multiple back-end servers, the network traffic that occurs on a front-end server is often very high. The minimum network requirements for any high-end front-end server is a single 100 Mbps network interface card (NIC) running in full duplex mode (meaning that data can be transmitted and received simultaneously). Using a ratio of one front-end server to four back-end servers, a 4x400 MHz front-end server can transfer around 5.6 MBps of data to a back-end server. This example creates an extremely heavy amount of network traffic as the saturation point of a 100 Mbps full duplex network connection is considered to be around 7 to 8 MBps.

> **Note** On high-end, front-end servers with two or more 800 MHz or higher processors, it is recommended that you use two 100 Mbps full duplex network connections, or a single Gigabit Ethernet connection. Servers of this class can easily handle more than enough incoming sessions to saturate a single 100 Mbps full duplex connection. For more information on network saturation, see the Appendix.

To balance the client load across multiple POP3 front-end servers, you can use Microsoft Network Load Balancing. The Microsoft Network Load Balancing service will allow multiple front-end servers to appear as one server, with incoming connections intelligently spread among the pool of available front-end servers. For more information on load balancing, see the *Microsoft Windows 2000 Server Resource Kit*.

Secure Authentication

When determining the type of hardware you want for your POP3 front-end server, you should consider the type of authentication method you will use. When users connect to the front-end server, they must authenticate their identity in some way. When you perform a secure (or encrypted) authentication, as opposed to a basic (or clear-text) authentication, there will be a 25 percent increase in CPU activity. The majority of the increased processor usage is due to extra context switching (almost twice as much). This increase in processor usage is only incurred during the secure login portion of the POP3 session, but this can be quite significant because of the high connect and disconnect rate of POP3 clients. Depending on the security requirements of your organization, secure authentication may be necessary. In this scenario, you will need to anticipate this significant increase in processor usage.

Secure Sockets Layer

Secure Sockets Layer is a protocol that provides encryption of network traffic between a server and a client program. When SSL encryption is enabled on the Exchange 2000 Server and is used by clients to protect their network traffic, a significant performance penalty is incurred. In general, SSL encryption of all POP3 protocol streams using a 512-bit length key will increase CPU usage by a factor of approximately 80 percent. This assumes that all POP3 sessions between clients and front-end servers are encrypted, and all sessions between front-end servers and back-end servers are not encrypted. In this configuration, memory requirements are increased by approximately 3 percent. Additional server capacity should be considered when designing an Exchange topology that will use SSL encryption. The following table shows a 77 percent increase in CPU usage and a 3 percent increase in memory use when using SSL encryption to protect POP3 transactions. To generate the information in this table, a single processor 600 MHz front-end server, with 512 MB of RAM, is used. The average message size of each message sent to the server is 50 KB.

POP3 Traffic	% Processor	DELE/sec	STAT/sec	InetInfo Working Set
Unencrypted	47.60	17	32	81.28 MB
Encrypted	84.40	16	30	83.53 MB

Quick POP3 Front-End Scalability Guide

Use this section as a quick reference for determinging your hardware needs for a POP3 front-end server. The following list outlines the conclusions from the previous section. For more information, refer to the previous section.

- POP3 scales well to 2-processor servers, but is not recommended for 4-processor servers.
- Use a ratio of one front-end server to four back-end servers.
- 256 MB of RAM is sufficient for nearly all applications (with Store dependency disabled).
- POP3 uses virtually no disk resources, unless the server is paging or POP3 protocol logging is turned on.
- POP3 requires a second 100 Mbps network interface card (NIC), or a Gigabit Ethernet connection, if run on a high-end, 800 MHz, 2-processor server.
- A POP3 front-end server can be load balanced using Microsoft Network Load Balancing.
- The processor capacity of a POP3 front-end server should be doubled, if all connections will be performed over SSL.

Outlook Web Access

If you are planning to use a dedicated front-end server to run Microsoft Outlook® Web Access, you should consider the following factors when determining your hardware needs:

- Processor scalability
- Memory needs
- Disk usage
- Expected network traffic
- Implications of using Secure Authentication
- Implications of using SSL

The following front-end server scenario will be used in this section to discuss hardware needs when running Outlook Web Access:

Hardware: Pentium III 400-MHz, front-end server with 1 GB RAM and two 100-Mbps network cards. Each back-end server is configured with four 500-MHz Xeon processors with 1.3 GB RAM and 20 disk spindles. This profile achieves 50 percent CPU usage on each back-end server with five SMTP messages delivered to local mailboxes per second and 2.5 messages sent per second to other servers.

Scenario: Average message size is 50 KB. Basic authentication is used with no SSL encryption. No transport traffic occurs on the front-end server. Each Outlook Web Access user's connection duration is approximately 15 minutes and the following actions are performed per user/per day:

Logons	1
Msgs. Sent	2.5
Recipients of Message Sent	1
Msgs. Received	5
Msgs. Read	5
Check Mails	3
Msgs. Moved	.5
Msgs. Deleted	1.5

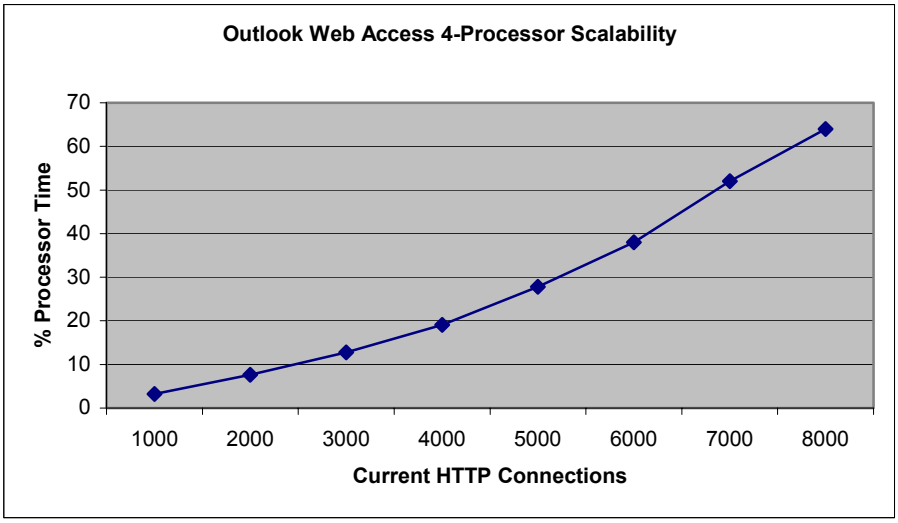
> **Note** See Appendix for actual Outlook Web Access Exchange Stress and Performance (ESP) script and definitions of context switching and Working Set. ISAPIs/sec is a rate counter for Outlook Web Access transactions. The ISAPIs/sec counter measures the number of requests to Internet Server API (ISAPI) extensions per second.

An Exchange 2000 front-end server configured with 4 400 MHz processors servicing only Outlook Web Access requests, exhibits the characteristics shown in the following table. The table gives an overview of processor usage, context switches/sec, network traffic, and memory usage on the front-end server as the number of back-end servers is increased. Access to Outlook Web Access is provided through Hypertext Transfer Protocol (HTTP). As back-end servers are added, the number of active HTTP connections to the front-end is increased to correspond with an increased number of simultaneous users.

Back-end Servers	% Processor	Context Switches/sec	HTTP Connections	ISAPI/sec	Network Traffic	InetInfo Working Set (MB)
1 Back-end server	3.2	2773	1000	25.4	1.2 MB/sec	146 MB
2 Back-end servers	7.6	4612	2000	51.27	2.2 MB/sec	169 MB
3 Back-end servers	12.8	6572	3000	77.3	3.5 MB/sec	200 MB
4 Back-end servers	19.1	7941	4000	101.5	4.3 MB/sec	243 MB
5 Back-end servers	27.8	9449	5000	123	5.3 MB/sec	274 MB
6 Back-end servers	38	9628	6000	147	6 MB/sec	299 MB
7 Back-end servers	52	11404	7000	168	7.8 MB/sec	327 MB
8 Back-end servers	64	12427	8000	198	8.6 MB/sec	360 MB

Processor Scalability

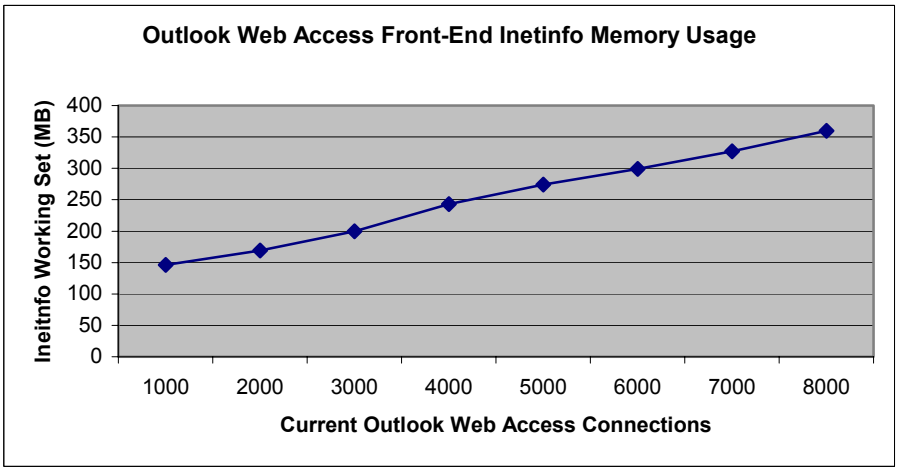
Unlike POP3, Outlook Web Access scales very well on 4-processor front-end server. In the Outlook Web Access scenario described earlier, only 12,427 context switches per second are generated when processor usage is at 64 percent. As the number of HTTP connections increases (corresponding to an increase in simultaneous Outlook Web Access users), the increase in processor usage is relatively linear. The following chart provides a graphic representation of this linear scalability.



The front-end server used to generate the information in the above diagram has four 400-MHz processors, while the back-end servers have four 500-MHz processors each. The back-end servers are running at 50 percent CPU, and, if you account for the 100-MHz difference in processor speed, the Outlook Web Access front-end to back-end ratio becomes approximately 1:4.

Memory

An Outlook Web Access front-end server will generally use approximately 30 KB of memory for each active Outlook Web Access connection, assuming that the traffic is not encrypted using Secure Sockets Layer. Compared to protocols such as POP3, this is a significant amount of memory consumption on the front-end server. As connections increase, the memory required by the InetInfo.exe process increases linearly as shown in the following graph.



Disk Usage

For information on disk usage for a dedicated Outlook Web Access front-end server, see the "Disk Usage" section, in the POP3 section of this document.

Network Usage

The amount of network traffic to your Outlook Web Access front-end server needs to be considered when determining your hardware requirements. The minimum network configuration for an Outlook Web Access front-end server is a single 100-Mbps full duplex network connection. Full duplex network connections can transmit and receive data simultaneously. Front-end servers hosting over 5,000 connections may need either an additional 100-Mbps full duplex connection, or be upgraded to a Gigabit Ethernet based network.

To balance the client load across multiple Outlook Web Access front-end servers, you can use Microsoft Network Load Balancing. For more information on load balancing, see the *Microsoft Windows 2000 Server Resource Kit*.

Secure Authentication

Secure authentication schemes such as Windows Integrated Security (including NTLM and Kerberos authentication) and HTTP Digest authentication are not supported in an Exchange 2000 Server front-end and back-end topology. Secure Sockets Layer can be used to provide strong encryption of all HTTP messages (including authentication tokens).

Secure Sockets Layer

Secure Sockets Layer encryption of HTTP sessions between clients and an Outlook Web Access front-end server requires increased CPU and memory resources. If SSL is used to encrypt all sessions between clients and the front-end server, CPU requirements can increase up to 3 times, based on number of ISAPI extension requests per second handled by the front-end server. Internet Server API (ISAPI) extension requests per second can be monitored using the Performance Monitor application and are a reliable indicator of the number of Outlook Web Access transactions occurring per second on the front-end server. The following chart demonstrates the difference in processor usage on a front-end server without SSL encryption versus one with SSL encryption over a range of ISAPI/sec levels.

ISAPI/sec	% Processor (Cleartext)	% Processor (SSL)
13	15	28
14	16	42
16	18	66
17	25	74

In this scenario, the amount of memory used by the InetInfo process increases by a factor of approximately 1.6 times, based on concurrent HTTP connections to the server. The InetInfo process, which is part of Microsoft Internet Information Services (IIS) manages Outlook Web Access connections. The following table demonstrates the difference in InetInfo memory usage on a front-end server without SSL encryption versus with one SSL encryption over a range of HTTP connection levels.

HTTP Connections	InetInfo Working Set (Cleartext)	InetInfo Working Set (SSL)
1,000	82.17 MB	134.94 MB
2,000	142.20 MB	233.28 MB
3,000	197.90 MB	323.52 MB
4,000	260.36 MB	383.19 MB

Deployment of SSL encryption on an Outlook Web Access front-end server should include careful planning to ensure adequate CPU and memory resources.

Quick Outlook Web Access Front-End Scalability Guide

Use this section as a quick reference for determinging your hardware needs for an Outlook Web Access front-end server. The following list outlines the conclusions from the previous section. For more information, refer to the previous section.

- Outlook Web Access scales well to 4-processor servers.
- Use a ratio of one front-end server to four back-end servers.
- Requires 30 KB of RAM per active connection.
- Outlook Web Acces uses virtually no disk resources, unless the front-end server is paging or HTTP-DAV protocol logging is turned on.
- Outlook Web Access requires a second 100-Mbps NIC if it is servicing over 5,000 connections.
- An Outlook Web Access front-end server can be load balanced using Microsoft Network Load Balancing.
- Outlook Web Access SSL connections require up to 3 times more processing power and 60 percent more memory on their front-end server.

IMAP4

If you are planning to use a dedicated front-end server to run Internet Message Access Protocol version 4 (IMAP4), you should consider the following factors when determining your hardware needs:

- Processor scalability

- Memory needs
- Disk usage
- Expected network traffic
- Implications of using Secure Authentication
- Implications of using SSL

The following front-end server scenario will be used in this section to discuss hardware needs when configuring a front-end server for IMAP4 access:

Hardware: Pentium III, 400-MHz front-end server with 1 GB of RAM and two 100-Mbps network cards for the front-end server. Each back-end server is configured with four 500-MHz Xeon processors with 1.3 GB of RAM and 20 disk spindles (individual hard drives). This configuration achieves 80 percent CPU usage on each back-end server, with one SMTP message delivered per second, for each back-end server.

Scenario: Average corporate IMAP4 user with mean message size of 50 KB (see Appendix for actual script). No transport traffic occurs on the front-end server. Basic authentication with no SSL encryption is used for client connections to the front-end server.

> **Note** See Appendix for actual Exchange Stress and Performance tool IMAP4 script and definitions of context switching and Working Set.

The following table provides an overview of the amount of IMAP4 traffic that can be handled by a single 2-processor 400 MHz front-end server with a range of back-end servers. The UID/sec rate counter measures the number of UID (unique ID) commands received by the IMAP server per second. This counter is a good measure of the overall number of IMAP4 transactions occurring per second.

Back-end Servers	% Processor	Context Switches/sec	UID/Sec	IMAP4 Connections	Network Traffic	InetInfo Working Set
1 Back-end server	6.2	1080	6.4	5,000	710 KB/sec	144 MB
2 Back-end servers	9.4	1635	10.3	10,000	877 KB/sec	147 MB
3 Back-end servers	12.5	2024	15.5	15,000	1104 KB/sec	151 MB

Processor Scalability

When determining your hardware needs for a dedicated IMAP4 front-end server, you will need to consider processor scalability. An Exchange 2000 front-end server only servicing IMAP4 clients does not use a large amount of CPU resources; as shown in the above table with 15,000 concurrent IMAP4 users, only 12.5 percent of available CPU is used. Using a profile in which users are logged on for long periods of time (typical of an average corporate user), it is possible for a single 2-processor front-end server to service ten 4-processor back-end servers where the front-end server and back-end servers all have the same processor speed. However, an Internet service provider (ISP) profile, with many more connections being created and terminated and less time between user actions, will have a much lower ratio, such as one 2-processor front-end server to five 4-processor back-end servers.

IMAP4 is most efficient when running on a 2-processor front-end server. Like POP3, IMAP4 begins to context switch excessively under moderate load on 4-processor front-end server. Subsequently, it is not recommended that you run an IMAP4 front-end server on hardware that has more than four processors.

Memory

Like POP3, IMAP4 does not require a large amount of physical memory. 256 MB of RAM is adequate for IMAP4 front-end servers that are servicing less then five back-end servers. If your front-end server will be servicing more than five back-end servers, you will need 512 MB of RAM installed on your front-end server.

Disk Usage

For information on disk usage for a dedicated IMAP4 front-end server, see the "Disk Usage" section, in the POP3 section of this document.

Network Usage

An IMAP4 front-end server uses a lot of network resources when servicing multiple back-end servers. The minimum network requirements of a high-end IMAP4 front-end server is a single 100-Mbps NIC running full duplex. Depending on the type of users accessing the system and the number of back-end servers being serviced, it may be necessary to add an additional 100-Mbps to each NIC, or move to a gigabit-based network. For more information on network usage, see the "Network Usage" section, in the POP3 section of this document.

Microsoft Network Load Balancing can also be used to balance the client load across multiple IMAP4 front-end servers. For more information on load balancing, see the *Microsoft Windows 2000 Server Resource Kit*.

Secure Authentication

When considering what hardware to purchase to support IMAP4 users on a front-end server, the type of authentication you plan to use may be an issue (although not as much of an issue as in the POP3 case). When you use secure authentication, there is a 25 percent increase in CPU activity, when compared to basic (or cleartext) authentication. The majority of this increased processor usage is due to extra context switching (almost twice as much). IMAP4 connections generally remain open as long as the client software is running, so logging on to the server is not as common as it is with POP3. Therefore, the processor usage associated with IMAP4 secure authentication is not increased significantly; in most cases, additional hardware resources will not be necessary to support secure authentication.

Secure Sockets Layer

When you are designing an Exchange topology that will use SSL encryption, you should consider additional server capacity. SSL encryption of all IMAP sessions between clients and a front-end server results in additional CPU requirements equal to approximately 1.5 times the CPU requirements of unencrypted IMAP sessions. This assumes that traffic between the front-end server and back-end servers is not encrypted, and that all client sessions are encrypted using a 512-bit length key. In this configuration, memory requirements are increased by 10 percent. The following table demonstrates a significant increase in processor usage and memory usage when SSL encryption is used and LOGIN operations per second and UID operations per second are standardized. The LOGIN/sec statistic indicates the number of new connections opened on the server per second, and the UID/sec statistic indicates the number of unique ID operations (a good indication of the number of IMAP4 transactions) on the server per second. To generate the information in this table, a single processor 600 MHz front-end server with 512 MB of RAM is used. The average message size of each message transferred from server to client is 50 KB.

IMAP4 Traffic	% Processor	LOGIN/sec	UID/sec	InetInfo Working Set
Unencrypted	27.61	18	35	58.06 MB
Encrypted	41.37	18	34	63.33 MB

Quick Front-End IMAP4 Scalability Guide

Use this section as a quick reference for determinging your hardware needs for an IMAP4 front-end server. The following list outlines the conclusions from the previous section. For more information, refer to the previous section.

- IMAP4 scales well to a 2-processor front-end server.
- Use a ratio of one IMAP4 front-end server to eight back-end servers.
- IMAP4 requires a minimum of 256 MB of RAM. However, if you are using a front-end server that services more than five back-end servers, you should use 512 MB of RAM.

- IMAP4 uses virtually no disk resources, unless the front-end server is paging or IMAP4 protocol logging is turned on.
- A single 100 Mbps full duplex network connection is sufficient for all but the most demanding front-end applications such as environments where large message attachments are common. Depending on the type of users you plan to service and the number of back-end servers being serviced, it may be necessary to add an additional 100 Mbps full duplex NIC or move to a gigabit-based network.
- An IMAP4 front-end server can be load balanced using Microsoft Network Load Balancing.
- SSL connections generate a 50 percent increase in CPU activity, and require an additional 10 percent of physical memory.

SMTP Gateway Scalability

Simple Mail Transfer Protocol (SMTP) is a network protocol designed to move electronic mail messages across a network to a destination server. In large data centers, it is generally recommended that you dedicate specific Exchange 2000 servers to handle only inbound and outbound SMTP traffic. These servers are usually called SMTP gateways or SMTP hubs. They are responsible for moving SMTP mail between clients and Exchange 2000 mailbox servers (which are back-end servers). This section describes only SMTP characteristics; Message Transfer Agent (MTA) and message conversion is not covered here.

If you are planning to use a dedicated front-end server as an SMTP gateway, you should consider the following factors when determining your hardware needs:

- Processor scalability
- Memory needs
- Disk usage
- Expected network traffic
- Implications of using Secure Authentication
- Implications of using SSL

The following front-end server scenario will be used in this section to discuss hardware needs for an SMTP gateway:

Hardware: Pentium III 550-MHz server with 2 GB of RAM and two 100-Mbps network cards. The SMTP Queue directory was placed on a RAID (redundant array of inexpensive disks) 0+1 disk array with 12 spindles backed by 64 MB of write cache. A RAID 0+1 disk array is striped and mirrored, meaning that half of the

spindles in the array are striped together such that they appear as one large disk, and those 6 spindles are mirrored to the remaining 6 spindles in the array for data redundancy. The write cache allows the system to write up to 64 MB of data to the disks without waiting for the disks to complete the write process.

Scenario: The average size of messages transferred between clients and the SMTP gateway is 50 KB. Anonymous authentication is used with no SSL Transport Layer Security (TLS) encryption of the network traffic.

Note See Appendix for definition of context switching.

The following chart provides an overview of performance on a 4-processor 550 MHz SMTP gateway server serving a range of back-end servers (which are the final destination for messages sent through the SMTP gateway).

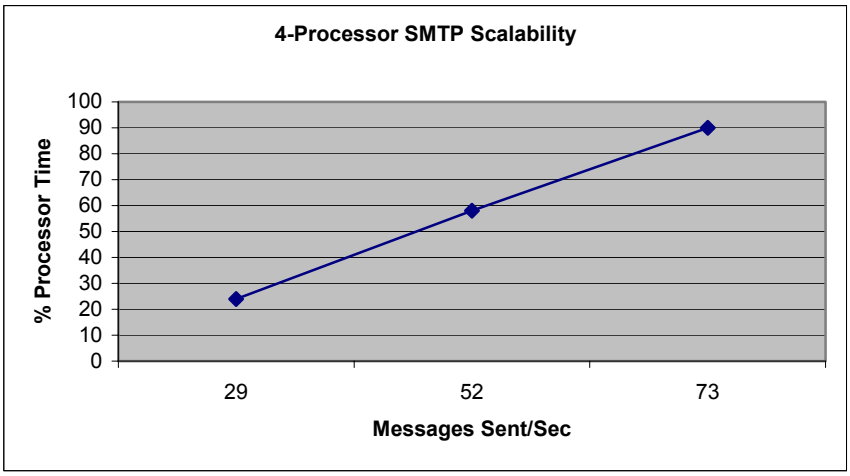
Back-end Servers	% Processor	Context Switches/sec	Messages Sent/sec	Network Usage	Disk Writes/sec
4 Back-end servers	24	6,995	29	2.75 MB	210
8 Back-end servers	58	11,781	52	4.9 MB	338
12 Back-end servers	90	14,156	73	7 MB	522

The next table provides an overview of performance on an 8-processor 550 MHz SMTP gateway server. Note the excessive context switching on this server at 74 percent CPU usage, which is indicative of poor scalability.

Back-end Servers	% Processor	Context Switches/sec	Messages Sent/sec	Network Usage	Disk Writes/sec
15 Back-end servers	74	29,954	86.8	8.6 MB	725

Processor Scalability

SMTP gateway servers scale well to 4-processor servers. This is because running 90 percent CPU usage on a high-end 4-processor server will only cause 14,000 context switches to occur per second, which is quite efficient. On the other hand, SMTP gateway servers do not scale well to 8-processor servers. The scenario described above only has 74 percent CPU usage, with almost 30,000 context switches occurring per second, which is an inefficient use of the processors. The following chart demonstrates the relative linear scalability of SMTP traffic on a 4-processor server.



Memory

SMTP gateway servers use memory primarily for maintaining connections and keeping track of vital information about messages in the queues. SMTP servers generally do not have thousands of connections, so the memory required for this purpose is not significant. However, the memory used to store message properties on queued e-mail can be significant. SMTP stores messages in the queue in two states: opened (keeps a handle open) or closed (closes handle). SMTP can keep 1,000 messages in the queue open at any given time, closing old messages as new ones arrive. An open message in the queue consumes approximately 10 KB of memory in the InetInfo process, and a closed message consumes approximately 4 KB of InetInfo memory.

The amount of memory required by an SMTP gateway server is a function of how many messages the server can be expected to have queued at a given time. The following examples demonstrate how to calculate the approximate amount of memory that will be consumed by SMTP queues.

- 1,000 open messages=10 MB of InetInfo memory
- 1,000 open messages + 20,000 closed messages = 80 MB of InetInfo memory
- 1,000 open messages + 89,000 closed messages = 366 MB of InetInfo memory

> **Note** The default maximum number of messages that an SMTP gateway will queue before refusing new messages is 90,000.

SMTP gateways are often used for distribution list expansion. Distribution lists allow messages to be sent to a single address and then distributed to multiple recipients. Distribution list expansion refers to the process of delivering a distribution list message to all members of the list. Distribution list expansion also affects the amount of memory used on an SMTP gateway. Each recipient expanded by a distribution list uses 1 KB of InetInfo memory. Servers that expand very large distribution lists, or often expand medium sized distribution lists, will require more memory.

Low traffic SMTP gateway servers can perform adequately with 256 MB of RAM. In high-traffic data centers, where large queues are common and large distribution lists are expanded, at least 512 MB of RAM is preferred. In general, an SMTP gateway server will not benefit by having more than 1 GB of memory. When a large SMTP queue is generated on a server that does not have enough memory, excessive paging will occur. This will dramatically increase the time it takes for the server to handle the queues.

> **Note** The maximum number of messages that are kept open and the maximum number of messages allowed in the queue can be configured manually.

Disk Usage

A server running as an SMTP gateway uses hard disks extensively. Every message that is received by an SMTP gateway is saved to a disk. If you refer back to the 4-Processor SMTP Scalability chart in the "SMTP Gateway Scalability" section, with a default message size of 50 KB, about 7 to 8 disk writes will occur for each message processed by the SMTP gateway. This is expected. In general, SMTP will do 7 disk writes for every message queued that is under 32 KB in size. An SMTP gateway's write buffer is 32 KB, so messages that are over this size will require an additional disk write for every 32 KB. For example, a 100 KB message will require 10 disk writes to save the message in the queue.

For these reasons, an SMTP gateway server requires a high performance disk-subsystem. It is recommended that a RAID (redundant array of inexpensive disks) 0+1 array be used with multiple disk spindles for the SMTP queue drive if the server is functioning as an SMTP gateway. A RAID 0+1 array is configured with half of the spindles in the array striped together to create one large volume, and this volume is mirrored to the other half of the available spindles for data redundancy. The number of disk spindles and the size of the write cache can be based on the expected SMTP message throughput of the server. For more information on disk subsystems, see the *Microsoft Windows 2000 Server Resource Kit*.

Network Usage

An SMTP gateway has the ability to process a large amount of SMTP data. As the load on the server increases, the server's network resources are put under

pressure. The minimum network capacity for a server acting as an SMTP gateway should be a single 100-Mbps, full duplex network connection. Full duplex connections are capable of receiving and transmitting data simultaneously. Depending on the expected server load, a second 100-Mbps connection may be required, or moving to a Gigabit Ethernet-based network. Sending around 50 messages per second, when the average message size is 50 KB, generates almost 5 MBps of network traffic. Servers expected to handle a load greater than this should have either two 100-Mbps, full duplex network connections, or have a single Gigabit Ethernet network connection.

You can also use Microsoft Network Load Balancing to balance SMTP traffic across multiple SMTP gateway servers. For more information on load balancing, see the *Microsoft Windows 2000 Server Resource Kit*.

Secure Sockets Layer Transport Layer Security

SMTP uses a protocol known as Transport Layer Security (TLS) to handle SSL encrypted network sessions. SSL encryption of SMTP streams does not appear to result in significant increases to CPU or memory requirements. More specifically, a 7.6 percent increase in processor activity and a 9.6 percent increase in memory requirements were seen. The topology consists of two SMTP servers relaying mail through SSL-encrypted sessions to a single server (where the measurements were taken). The average message size for this scenario is 50 KB and the front-end server being used is a single processor, 600 MHz server with 512 MB of RAM. The following table provides the raw data obtained in this test. Note the small difference in processor usage and InetInfo Working Set (memory consumption) between unencrypted and encrypted SMTP configurations.

SMTP Traffic	% Processor	Front-End Messages Received/Sec	InetInfo Working Set
Unencrypted	53.63	17.90	30.19 MB
Encrypted	57.69	17.95	33.10 MB

Quick SMTP Gateway Scalability Guide

Use this section as a quick reference for determinging your hardware needs for an SMTP gateway. The following list outlines the conclusions from the previous section. For more information, refer to the previous section.

- An SMTP gateway scales well to 4-processor servers, but not to 8-processors servers.
- An SMTP gateway requires a minimum of 256 MB of RAM to manage large queues and expand distribution lists. 512 MB of RAM or higher is recommended.
- SMTP gateways require a large amount of disk resources; at least 7 disk writes are required for each message queued. The drive containing the queue

on high-end servers should be configured as RAID 0+1 with multiple disk spindles making use of a write caching array controller.

- 2-processor gateways require a single 100-Mbps connection. High-end, 4-processor servers should be fitted with two 100-Mbps connections or a single Gigabit Ethernet connection.
- SMTP traffic can be load balanced using Microsoft Network Load Balancing, and encrypting SMTP traffic using Transport Layer Security (TLS) does not have a significant impact on the processor or memory needs.

Front-End Licensing Guide

Microsoft Windows 2000 supports two types of licensing models: per seat and per server. The Windows 2000 License Logging Service maintains a list (on disk and in memory) of all users who authenticate against a server using the per seat licensing model. The per server model does not keep a list of users. The list of authenticated users in the per seat configuration does not consume an excessive amount of memory on servers with less than 50,000 users. However, the memory footprint of the License Logging Service can become excessive when running with a per seat licensing model in a front-end and back-end topology, with hundreds of thousands of users.

In this scenario, with the front-end servers load balancing the client requests, the front-end server's License Logging Service will eventually build a list of all users in the site (including all users on all back-end servers). Depending on the size of the site, this can consume hundreds of megabytes of memory in the License Logging Service on the front-end server. For this reason, it is recommended that the per server licensing model be used in large front-end and back-end topologies. For more information on the License Logging Service, see the Windows 2000 documentation and the *Microsoft Windows 2000 Server Resource Kit*.

Appendix

The appendix includes additional detailed information about concepts discussed in the body of this document. In addition, the Exchange Stress and Performance (ESP) tool scripts, which were used to obtain the data discussed in this document, are included.

Definitions

Context Switching

A context switch occurs when the Windows 2000 kernel switches the processor from one thread to another—for example, when a thread with a higher priority than the running thread becomes ready. Context switching activity is important for several reasons. A program that monopolizes the processor lowers the rate of context switches because it does not allow much processor time for the other processes' threads. A high rate of context switching means that the processor is being shared repeatedly, by many threads of equal priority. A high context-switch rate often indicates that there are too many threads competing for the processors on the system. The rate of context switches can also affect performance of multiprocessor computers. For information about how to monitor and tune context-switch activity on multiprocessor systems, see "Measuring Multiprocessor System Activity" section in the *Microsoft Windows 2000 Server Resource Kit*. You can view context switch data by monitoring the *System/Context Switches/sec* counter in System Monitor.

Working Set

The Working Set is the set of memory pages (areas of memory allocated to a process) recently used by the threads in a process. If available memory on the server is above a specified threshold, pages are left in the Working Set of a process even if they are not in use. When available memory falls below a specified threshold, pages are removed from the Working Set. If these pages are needed, they will be soft-faulted back into the Working Set before they leave main memory and are made available for other processes to use.

4/8 Processor Scalability

In the scenario where 8-processor scalability was compared to 4-processor scalability, the comparison tests were run on a Compaq 8500 server. Four of the eight processors were physically removed when testing in the 4-processor scenario.

Network Saturation

In this document, network saturation means that, with 7 to 8 MBps of network traffic, a 100-Mbps full duplex network connection begins to get overloaded.

Front-end server

A server that acts as a proxy, passing along the protocol session to the appropriate back-end server.

Exchange Stress and Performance (ESP) Tool

An Exchange stress tool used to generate protocol load against the server. You can access ESP from the *Microsoft Exchange 2000 Server Resource Kit*, available at http://mspress.microsoft.com/prod/books/4355.htm

Protocol Scripts

The protocol scripts contained in this section can be used with the Exchange Stress and Performance (ESP) tool to obtain the data discussed in this document.

> **Note** Indentations in the sample script indicate lines that are continued.

POP3 ESP Script

The following sample POP3 ESP script was used to obtain the data discussed in the POP3 section of this document. This simple script connects to a random mailbox and proceeds to retrieve and delete all messages in the user's mailbox.

```
connect
mailbox RANDLIST(userlist.txt)
SLEEP RANDNUMBER(200,200)
LIST
retr all
dele all
quit
```

IMAP4 ESP Script

The following sample IMAP4 ESP script was used to obtain the data discussed in the IMAP4 section of this document. This script connects to a mailbox and emulates the usage pattern of an average corporate user.

```
CONNECT
CAPABILITY
AUTHENTICATE SEQULIST(IMAP_Users_1000-4999.txt)
LIST "" "*"
SELECT "INBOX"
FETCHUID
UIDFETCH 1:* (FLAGS)
UIDFETCH UIDALL (UID RFC822.SIZE RFC822.HEADER)
NOOP
UIDFETCH LAST 1 (UID RFC822.SIZE RFC822)
CHECK
```

```
SLEEP RANDNUMBER(360000,840000)
NOOP
FETCH LAST 1 (UID)
UIDFETCH LAST 2 (FLAGS)
NOOP
FETCHUID
UIDFETCH 1:* (FLAGS)
UIDFETCH LAST 1 (UID RFC822.SIZE RFC822.HEADER)
UIDFETCH LAST 1 (UID RFC822.SIZE RFC822)
CHECK
SLEEP RANDNUMBER(248000,712000)
%80 SKIP 6
FETCHUID
UIDCOPY LAST 1 RANDLIST(IMAP_UPS_FolderList.txt)
UIDSTORE LAST 1 +FLAGS (\DELETED)
UIDCOPY RANDOM RANDLIST(IMAP_UPS_FolderList.txt)
UIDSTORE SAME +FLAGS (\DELETED)
NOOP
FETCHUID
UIDCOPY LAST 1 "Trash"
UIDSTORE LAST 1 +FLAGS (\Deleted)
UIDCOPY RANDOM "Trash"
UIDSTORE SAME +FLAGS (\Deleted)
SLEEP RANDNUMBER(60000,180000)
NOOP
FETCH LAST 1 (UID)
UIDFETCH LAST 2 (FLAGS)
NOOP
CHECK
%80 SKIP 4
FETCHUID
UIDCOPY RANDOM RANDLIST(IMAP_UPS_FolderList.txt)
UIDSTORE SAME +FLAGS (\DELETED)
NOOP
FETCHUID
UIDFETCH RANDOM (UID RFC822.SIZE RFC822)
UIDCOPY SAME "Trash"
UIDSTORE SAME +FLAGS (\DELETED)
SLEEP RANDNUMBER(240000,420000)
FETCHUID
UIDFETCH RANDOM (UID RFC822.SIZE RFC822)
CHECK
%80 SKIP 4
FETCHUID
UIDCOPY RANDOM RANDLIST(IMAP_UPS_FolderList.txt)
UIDSTORE SAME +FLAGS (\DELETED)
NOOP
UIDFETCH RANDOM (UID RFC822.SIZE RFC822)
UIDCOPY SAME "Trash"
UIDSTORE SAME +FLAGS (\DELETED)
SLEEP RANDNUMBER(240000,420000)
CHECK
SLEEP RANDNUMBER(30000,90000)
CLOSE
SELECT "Trash"
STORE ALL +FLAGS (\Deleted)
EXPUNGE
LIST "Trash" "*"
SLEEP RANDNUMBER(15000,45000)
CLOSE
SELECT RANDLIST(IMAP_UPS_FolderList.txt)
FETCHUID
UIDFETCH 1:* (FLAGS)
CHECK
%80 SKIP 4
FETCHUID
UIDCOPY SAME RANDLIST(IMAP_UPS_FolderList.txt)
```

```
UIDSTORE SAME +FLAGS (\DELETED)
NOOP
FETCHUID
UIDFETCH RANDOM (UID RFC822.SIZE RFC822)
UIDCOPY SAME "Trash"
UIDSTORE SAME +FLAGS (\DELETED)
SLEEP RANDNUMBER(240000,420000)
CHECK
%80 SKIP 4
FETCHUID
UIDCOPY RANDOM RANDLIST(IMAP_UPS_FolderList.txt)
UIDSTORE SAME +FLAGS (\DELETED)
NOOP
FETCHUID
UIDFETCH RANDOM (UID RFC822.SIZE RFC822)
UIDCOPY SAME "Trash"
UIDSTORE SAME +FLAGS (\DELETED)
SLEEP RANDNUMBER(240000,420000)
CLOSE
SKIP 4
```

Outlook Web Access ESP Script

The following sample Outlook Web Access ESP script was used to obtain the data discussed in the Outlook Web Access section of this document. This script emulates the usage pattern of a typical Outlook Web Access user.

```
REM Note: No Cookies required for BMOVE, MOVE, PROPFIND, or SEARCH
 requests!
REM SLEEPs of => 60 min cause user context to time out; new cookie
 must be set w/ first ISAPI.
REM When counting lines for a SKIP, count all lines except REM
 lines, which are not compiled.
SET $xml$ = new XMLList
SET $host$ PutServerNameHere
SET $filepath$ PutScriptDirectoryFilePathHere
SET $authorization$ SEQULIST(c:\esp\davscripts\4312\webusers.txt)
SET $senduser1$ SEQULIST(c:\esp\davscripts\4312\smtpusers.txt)
SET $senduser2$ SEQULIST(c:\esp\davscripts\4312\smtpusers.txt)
SET $senduser3$ SEQULIST(c:\esp\davscripts\4312\smtpusers.txt)
SET $senduser4$ SEQULIST(c:\esp\davscripts\4312\smtpusers.txt)
SET $folders$ RANDLIST(c:\esp\davscripts\4312\folders.txt)
SET $month$ 06
SET $day$ RANDNUMBER (03,21)
SET $time$ RANDNUMBER (9,12)
SET $mailbox$ exchange
SET $cookie$ RESPHEADER(Set-Cookie)
REM
REM      <LOGON\AUTHENTICATE\INBOXVIEW>
REM
ADDHEADER Accept: image/gif, image/x-xbitmap, image/jpeg,
 image/pjpeg, */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
GET /$mailbox$/$USER$/
ADDHEADER Accept: */*
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/?Cmd=navbar
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/?Cmd=navbar
GET /exchweb/controls/navbar.css
```

```
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
GET /exchweb/controls/olstyle.css
GET /exchweb/controls/olprint.css
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/?Cmd=navbar
GET /exchweb/controls/navbar.js
GET /exchweb/img/logo-ie5.gif
GET /exchweb/img/navbar-inbox.gif
DELHEADER Referer:
DELHEADER Accept-Language:
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/
ADDHEADER Accept-Language: en-us
GET /exchweb/controls/dropmenu.htc
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/?Cmd=navbar
GET /exchweb/img/navbar-calendar.gif
GET /exchweb/img/navbar-contacts.gif
GET /exchweb/img/navbar-options.gif
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
GET /exchweb/controls/webclientutil.js
GET /exchweb/controls/note.js
GET /exchweb/controls/buttons.js
GET /exchweb/img/tool-newmail.gif
GET /exchweb/img/tool-reply.gif
GET /exchweb/controls/blank.htm
GET /exchweb/img/tool-forward.gif
GET /exchweb/img/tool-refresh.gif
GET /exchweb/img/tool-delete.gif
GET /exchweb/img/tool-empty.gif
GET /exchweb/img/tool-addressbook.gif
GET /exchweb/img/tool-help.gif
GET /exchweb/img/tool-newcntc.gif
GET /exchweb/img/status-info.gif
GET /exchweb/img/view-prevpage.gif
GET /exchweb/img/view-nextpage.gif
DELHEADER Referer:
DELHEADER Accept-Language:
GET /exchweb/CONTROLS/view.htc
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER Accept-Language: en-us
GET /exchweb/controls/view.js
DELHEADER Accept-Language:
GET /exchweb/views/standardview.xsl
ADDHEADER Accept-Language: en-us
GET /exchweb/img/tool-send.gif
DELHEADER Accept-Language:
DELHEADER Accept:
DELHEADER Accept-Encoding:
ADDHEADER content-type: text/xml
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.1
DELHEADER content-type:
DELHEADER Brief:
DELHEADER range:
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
GET /exchweb/img/view-importance.gif
GET /exchweb/img/view-document.gif
GET /exchweb/img/view-flag.gif
GET /exchweb/img/view-paperclip.gif
GET /exchweb/img/icon-msg-unread.gif
```

```
GET /exchweb/img/view-sortdown.gif
DELHEADER Accept:
DELHEADER Referer:
DELHEADER Accept-Language:
DELHEADER Accept-Encoding:
DELHEADER User-Agent:
REM
REM              <CHECK MAIL FOR READ>
REM
ADDHEADER content-type: text/xml
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.1
 PARSEXML dav:!href
DELHEADER content-type:
DELHEADER Brief:
DELHEADER range:
SET $name1$ $xml$.NEXT
SET $name2$ $xml$.NEXT
SET $name3$ $xml$.NEXT
SET $name4$ $xml$.NEXT
REM
REM             <READ 1>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.0;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Cookie: $cookie$
GET $name2$?Cmd=open
ADDHEADER Accept: */*
ADDHEADER Referer: $name2$?Cmd=open
GET /exchweb/controls/olstyle.css
GET /exchweb/controls/olprint.css
GET /exchweb/controls/webclientutil.js
GET /exchweb/controls/MMReadMsgUtil.js
GET /exchweb/controls/readnote.js
GET /exchweb/controls/buttons.js
GET /exchweb/img/tool-reply.gif
GET /exchweb/img/tool-reply_all.gif
GET /exchweb/img/tool-forward.gif
GET /exchweb/img/tool-move.gif
GET /exchweb/img/tool-help.gif
GET /exchweb/img/status-info.gif
DELHEADER Accept:
DELHEADER Referer:
DELHEADER Accept-Language:
DELHEADER Accept-Encoding:
DELHEADER User-Agent:
SLEEP RANDNUMBER(360000,840000)
REM
REM             <READ 1>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.0;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER Cookie: $cookie$
GET $name3$?Cmd=open
REM  SET $cookie$ RESPHEADER(Set-Cookie)
SLEEP RANDNUMBER(124000,240000)
```

```
REM
REM             <SEND 1 (to 4 Recips)>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/Drafts/?Cmd=new
DELHEADER Cookie:
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/Drafts/?Cmd=new
GET /exchweb/controls/olstyle.css
GET /exchweb/controls/olprint.css
GET /exchweb/controls/webclientutil.js
GET /exchweb/controls/recipients.js
GET /exchweb/controls/MMReadMsgUtil.js
GET /exchweb/controls/MMComposeMsgUtil.js
GET /exchweb/controls/composenote.js
GET /exchweb/controls/buttons.js
GET /exchweb/img/tool-send.gif
GET /exchweb/img/tool-save.gif
GET /exchweb/img/tool-print.gif
DELHEADER Referer:
DELHEADER Accept-Language:
GET /exchweb/controls/msgCtrl.htc
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/Drafts/?Cmd=new
ADDHEADER Accept-Language: en-us
GET /exchweb/img/tool-checknames.gif
DELHEADER Referer:
GET /exchweb/controls/msgCtrl.htc
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/Drafts/?Cmd=new
GET /exchweb/img/tool-highimport.gif
GET /exchweb/img/tool-lowimport.gif
GET /exchweb/img/winme.gif
DELHEADER Referer:
DELHEADER Accept-Language:
REM GET /exchweb/CONTROLS/formatbar.htc
ADDHEADER Accept-Language: en-us
GET /exchweb/CONTROLS/formatbar.htc
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/Drafts/?Cmd=new
GET /exchweb/img/form-fgcolor.gif
GET /exchweb/img/form-bold.gif
GET /exchweb/img/form-italic.gif
GET /exchweb/img/form-justify_center.gif
GET /exchweb/img/form-justify_left.gif
GET /exchweb/img/form-justify_right.gif
GET /exchweb/img/form-bullist.gif
GET /exchweb/img/form-numlist.gif
GET /exchweb/img/form-deindent.gif
GET /exchweb/img/form-inindent.gif
DELHEADER Accept:
DELHEADER Accept-Encoding:
ADDHEADER Content-type: application/x-www-UTF8-encoded
ADDHEADER Cookie: $cookie$
POST /$mailbox$/$USER$/Drafts $filepath$4312-IE5-SendMsgTo4Data.1
 replacevars
DELHEADER Accept-Language:
DELHEADER Content-type:
DELHEADER Referer:
DELHEADER User-Agent:
SLEEP RANDNUMBER(124000,240000)
REM
REM             <READ 1>
REM
ADDHEADER Accept: */*
```

```
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.0;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER Cookie: $cookie$
GET $name4$?Cmd=open
REM
REM             <CHECK MAIL FOR MOVE, DELETE>
REM
ADDHEADER content-type: text/xml
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.2
 PARSEXML dav:!displayname
DELHEADER content-type:
DELHEADER Brief:
DELHEADER range:
SET $name1$ $xml$.NEXT
URIESCAPE $name1$
SET $name2$ $xml$.NEXT
URIESCAPE $name2$
SET $name3$ $xml$.NEXT
URIESCAPE $name3$
SET $name4$ $xml$.NEXT
URIESCAPE $name4$
REM
REM             <MOVE 1>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/?Cmd=dialog&template=dlg_movecopy
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/?Cmd=dialog&template=dlg_movecopy
GET /exchweb/controls/webclientutil.js
DELHEADER Referer:
DELHEADER Accept-Language:
GET /exchweb/controls/ExchangeTree.htc
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/?Cmd=dialog&template=dlg_movecopy
ADDHEADER Accept-Language: en-us
GET /exchweb/controls/exchangetree.js
DELHEADER Accept:
DELHEADER Accept-Language:
DELHEADER Accept-Encoding:
ADDHEADER Depth: 0
ADDHEADER Brief: t
ADDHEADER Content-type: text/xml
PROPFIND /$mailbox$/$USER$/ $filepath$4312-IE5-MoveMsgDialogData.1
DELHEADER Depth:
DELHEADER Brief:
DELHEADER Content-type:
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
GET /exchweb/img/tree-splus.gif
DELHEADER Accept:
DELHEADER Accept-Language:
DELHEADER Accept-Encoding:
ADDHEADER Content-type: text/xml
```

```
ADDHEADER Brief: t
ADDHEADER Translate: f
SEARCH /$mailbox$/$USER$/ $filepath$4312-IE5-MoveMsgDialogData.2
DELHEADER Content-type:
DELHEADER Brief:
DELHEADER Translate:
ADDHEADER Accept: */*
ADDHEADER Accept-Encoding: gzip, deflate
GET /exchweb/img/tree-folder.gif
GET /exchweb/img/tree-clear9.gif
GET /exchweb/img/tree-appointment.gif
GET /exchweb/img/tree-contact.gif
GET /exchweb/img/tree-deleted.gif
GET /exchweb/img/tree-inbox.gif
GET /exchweb/img/tree-journal.gif
GET /exchweb/img/tree-stickynote.gif
GET /exchweb/img/tree-outbox.gif
GET /exchweb/img/tree-sent_itm.gif
GET /exchweb/img/tree-task.gif
GET /exchweb/img/tree-sminus.gif
DELHEADER Accept:
DELHEADER Accept-Encoding:
ADDHEADER Destination:
 http://$host$/$mailbox$/$USER$/$folders$/$name2$
ADDHEADER Content-type: text/xml
ADDHEADER Overwrite: F
ADDHEADER Translate: F
ADDHEADER Allow-rename: t
MOVE /$mailbox$/$USER$/Inbox/$name2$
DELHEADER Destination:
DELHEADER Overwrite:
DELHEADER Translate:
DELHEADER Allow-rename:
ADDHEADER Accept-Language: en-us
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.1
DELHEADER Brief:
DELHEADER range:
ADDHEADER Accept: */*
ADDHEADER Accept-Encoding: gzip, deflate
GET /exchweb/img/view-importance.gif
GET /exchweb/img/view-paperclip.gif
GET /exchweb/img/icon-msg-unread.gif
GET /exchweb/img/view-document.gif
GET /exchweb/img/view-flag.gif
GET /exchweb/img/view-sortdown.gif
GET /exchweb/img/tool-send.gif
DELHEADER Accept:
DELHEADER Referer:
DELHEADER Accept-Language:
DELHEADER Accept-Encoding:
DELHEADER User-Agent:
REM
REM             <DELETE 1>
REM
ADDHEADER Content-type: text/xml
ADDHEADER Translate: f
ADDHEADER Allow-rename: t
ADDHEADER Overwrite: f
ADDHEADER Destination:
 http://$host$/$mailbox$/$USER$/Deleted%20Items
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
```

```
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
BMOVE /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-DelMsgXData.2
 replacevars
DELHEADER Translate:
DELHEADER Allow-rename:
DELHEADER Overwrite:
DELHEADER Destination:
ADDHEADER Accept-Language: en-us
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.1
DELHEADER Brief:
DELHEADER range:
ADDHEADER Accept: */*
ADDHEADER Accept-Encoding: gzip, deflate
GET /exchweb/img/view-importance.gif
GET /exchweb/img/view-document.gif
GET /exchweb/img/view-flag.gif
GET /exchweb/img/view-paperclip.gif
GET /exchweb/img/view-sortdown.gif
GET /exchweb/img/icon-msg-unread.gif
GET /exchweb/img/tool-send.gif
DELHEADER Accept:
DELHEADER Referer:
DELHEADER Accept-Language:
DELHEADER Accept-Encoding:
DELHEADER User-Agent:
SLEEP RANDNUMBER(60000,180000)
REM
REM             <DELETE 1>
REM
ADDHEADER Content-type: text/xml
ADDHEADER Translate: f
ADDHEADER Allow-rename: t
ADDHEADER Overwrite: f
ADDHEADER Destination:
 http://$host$/$mailbox$/$USER$/Deleted%20Items
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
BMOVE /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-DelMsgXData.2b
 replacevars
DELHEADER Translate:
DELHEADER Allow-rename:
DELHEADER Overwrite:
DELHEADER Destination:
ADDHEADER Accept-Language: en-us
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.1
DELHEADER Brief:
DELHEADER range:
DELHEADER Referer:
DELHEADER Accept-Language:
DELHEADER User-Agent:
SLEEP RANDNUMBER(360000,540000)
REM
REM             <CHECK MAIL FOR READ>
REM
ADDHEADER content-type: text/xml
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
```

```
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.1
 PARSEXML dav:!href
DELHEADER content-type:
DELHEADER Brief:
DELHEADER range:
SET $name1$ $xml$.NEXT
SET $name2$ $xml$.NEXT
SET $name3$ $xml$.NEXT
SET $name4$ $xml$.NEXT
SET $name5$ $xml$.NEXT
REM
REM             <READ 1>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.0;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER Cookie: $cookie$
GET $name2$?Cmd=open
REM  SET $cookie$ RESPHEADER(Set-Cookie)
REM
REM             <READ 1>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.0;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER Cookie: $cookie$
GET $name3$?Cmd=open
REM
REM             <READ 1>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.0;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER Cookie: $cookie$
GET $name4$?Cmd=open
SLEEP RANDNUMBER(135000,225000)
REM
REM              <SEND 1 (to 4 Recips)>
REM
%80 SKIP 16
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/Drafts/?Cmd=new
DELHEADER Cookie:
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/Drafts/?Cmd=new
ADDHEADER Content-type: application/x-www-UTF8-encoded
ADDHEADER Cookie: $cookie$
```

```
POST /$mailbox$/$USER$/Drafts $filepath$4312-IE5-SendMsgTo4Data.1
 replacevars
DELHEADER Accept-Language:
DELHEADER Content-type:
DELHEADER Referer:
DELHEADER User-Agent:
REM
REM              <READ 1>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.0;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER Cookie: $cookie$
GET $name5$?Cmd=open
SLEEP RANDNUMBER(135000,225000)
REM
REM              <CHECK MAIL FOR MOVE, DELETE>
REM
ADDHEADER content-type: text/xml
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.2
 PARSEXML dav:!displayname
DELHEADER content-type:
DELHEADER Brief:
DELHEADER range:
SET $name1$ $xml$.NEXT
URIESCAPE $name1$
SET $name2$ $xml$.NEXT
URIESCAPE $name2$
SET $name3$ $xml$.NEXT
URIESCAPE $name3$
SET $name4$ $xml$.NEXT
URIESCAPE $name4$
REM
REM              <DELETE 1>
REM
ADDHEADER Content-type: text/xml
ADDHEADER Translate: f
ADDHEADER Allow-rename: t
ADDHEADER Overwrite: f
ADDHEADER Destination:
 http://$host$/$mailbox$/$USER$/Deleted%20Items
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
BMOVE /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-DelMsgXData.2b
 replacevars
DELHEADER Translate:
DELHEADER Allow-rename:
DELHEADER Overwrite:
DELHEADER Destination:
ADDHEADER Accept-Language: en-us
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
SEARCH /$mailbox$/$USER$/Inbox/ $filepath$4312-IE5-SearchInboxData.1
DELHEADER Brief:
DELHEADER range:
DELHEADER Referer:
DELHEADER Accept-Language:
```

```
DELHEADER User-Agent:
DELHEADER Content-type:
REM
REM          <CALENDARING SECTION>
REM
%75 SKIP 156
REM
REM          <GET CALENDAR VIEW>
REM
ADDHEADER Accept: image/gif, image/x-xbitmap, image/jpeg,
 image/pjpeg, */*
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/?Cmd=navbar
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/Calendar/?Cmd=contents
ADDHEADER Accept: */*
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Calendar/?Cmd=contents
GET /exchweb/controls/olstyle.css
GET /exchweb/controls/olprint.css
GET /exchweb/controls/webclientutil.js
GET /exchweb/controls/buttons.js
GET /exchweb/img/tool-newmail.gif
GET /exchweb/img/tool-newmtg.gif
GET /exchweb/img/tool-newcntc.gif
GET /exchweb/img/tool-newfolder.gif
GET /exchweb/img/tool-newappt.gif
GET /exchweb/img/tool-downarrow.gif
GET /exchweb/img/tool-delete.gif
GET /exchweb/img/tool-refresh.gif
GET /exchweb/img/tool-dayview.gif
GET /exchweb/img/tool-weekview.gif
GET /exchweb/img/tool-monthview.gif
GET /exchweb/img/tool-help.gif
GET /exchweb/img/tool-addressbook.gif
GET /exchweb/img/tree-folder.gif
GET /exchweb/controls/calendar.js
DELHEADER Referer:
DELHEADER Accept-Language:
GET /exchweb/controls/CalendarView.htc
GET /exchweb/controls/ExchangeCalendar.htc
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/?Cmd=navbar
ADDHEADER Accept-Language: en-us
GET /exchweb/controls/CalendarView.htc
GET /exchweb/controls/ExchangeCalendar.htc
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Calendar/?Cmd=contents
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/Calendar/
GET /exchweb/img/form-prev.gif
GET /exchweb/img/form-next.gif
DELHEADER Accept:
DELHEADER Accept-Language:
DELHEADER Accept-Encoding:
ADDHEADER Content-Type: text/xml
SEARCH /$mailbox$/$USER$/Calendar
 $filepath$IE5_4323_GetCalView_data.1
SEARCH /$mailbox$/$USER$/Calendar
 $filepath$IE5_4323_GetCalView_data.2
DELHEADER Content-Type:
DELHEADER Referer:
DELHEADER User-Agent:
DELHEADER Host:
```

```
REM
REM          <GET 7 DAY VIEW>
REM
ADDHEADER Content-Type: text/xml
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Calendar/?Cmd=contents
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
SEARCH /$mailbox$/$USER$/
 Calendar $filepath$IE5_4323_7-DayView_data.1
DELHEADER Content-Type:
DELHEADER Referer:
DELHEADER User-Agent:
DELHEADER Host:
SLEEP RANDNUMBER(135000,225000)
REM
REM         <CREATE APPOINTMENT AND SAVE>
REM
ADDHEADER Accept-Language: en-us
ADDHEADER Content-type: application/x-www-UTF8-encoded
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Calendar/?Cmd=new&mm=2&dd=20&yy=2000
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Cookie: $cookie$
POST /$mailbox$/$USER$/
 Calendar $filepath$IE5_4323_CreateAppt_data.1 replacevars
DELHEADER Accept-Language:
DELHEADER Content-type:
ADDHEADER Content-Type: text/xml
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Calendar/?Cmd=contents
SEARCH /$mailbox$/$USER$/Calendar
 $filepath$IE5_4323_CreateAppt_data.2
SEARCH /$mailbox$/$USER$/Calendar
 $filepath$IE5_4323_CreateAppt_data.3
DELHEADER Content-Type:
DELHEADER Referer:
DELHEADER User-Agent:
DELHEADER Host:
REM
REM        <SEND MEETING REQUEST TO 2 RECIPIENTS>
REM
ADDHEADER Accept: */*
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/Calendar/?Cmd=new&mm=4&dd=01&yy=2000
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Calendar/?Cmd=new&mm=4&dd=01&yy=2000
GET /exchweb/controls/recipients.js
GET /exchweb/controls/composeappt.js
GET /exchweb/img/tool-send.gif
GET /exchweb/img/tool-save.gif
GET /exchweb/img/tool-accept.gif
GET /exchweb/img/tool-tentative.gif
GET /exchweb/img/tool-decline.gif
GET /exchweb/img/tool-attach.gif
GET /exchweb/img/tool-print.gif
GET /exchweb/img/tool-checknames.gif
GET /exchweb/img/tool-highimport.gif
GET /exchweb/img/tool-lowimport.gif
```

```
GET /exchweb/img/tool-recur.gif
GET /exchweb/img/tool-cancelmtg.gif
GET /exchweb/img/status-info.gif
GET /exchweb/img/form-clock36.gif
GET /exchweb/img/form-down7x7.gif
DELHEADER Referer:
DELHEADER Accept-Language:
GET /exchweb/controls/msgCtrl.htc
ADDHEADER Accept-Language: en-us
GET /exchweb/controls/msgCtrl.htc
DELHEADER Accept-Language:
GET /exchweb/controls/datepicker.htc
ADDHEADER Accept-Language: en-us
GET /exchweb/controls/datepicker.htc
DELHEADER Accept:
DELHEADER Accept-Encoding:
ADDHEADER Content-type: application/x-www-UTF8-encoded
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Calendar/?Cmd=new&mm=4&dd=01&yy=2000
ADDHEADER Cookie: $cookie$
POST /$mailbox$/$USER$/Calendar
 $filepath$IE5_4323_SendMeetingRequest_data.1 replacevars
DELHEADER Accept-Language:
DELHEADER Content-type:
ADDHEADER Content-Type: text/xml
ADDHEADER Referer:
 http://$host$/$mailbox$/$USER$/Calendar/?Cmd=contents
SEARCH /$mailbox$/$USER$/Calendar
 $filepath$IE5_4323_SendMeetingRequest_data.2
SEARCH /$mailbox$/$USER$/Calendar
 $filepath$IE5_4323_SendMeetingRequest_data.3
DELHEADER Content-Type:
DELHEADER Referer:
DELHEADER User-Agent:
DELHEADER Host:
SLEEP RANDNUMBER(135000,225000)
REM
REM          <RETURN TO INBOX>
REM
ADDHEADER Accept: image/gif, image/x-xbitmap, image/jpeg,
 image/pjpeg, application/msword, */*
ADDHEADER Referer: http://$host$/$mailbox$/$USER$/?Cmd=navbar
ADDHEADER Accept-Language: en-us
ADDHEADER Accept-Encoding: gzip, deflate
ADDHEADER User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
 Windows NT 5.0)
ADDHEADER Host: $host$
ADDHEADER Cookie: $cookie$
GET /$mailbox$/$USER$/Inbox/?Cmd=contents&Page=1
DELHEADER Accept-Language:
ADDHEADER Accept: */*
ADDHEADER REFERER:
 http://$host$/$mailbox$/$USER$/Inbox/?Cmd=contents&Page=1
GET /exchweb/views/standardview.xsl
DELHEADER Accept:
DELHEADER Accept-Encoding:
ADDHEADER content-type: text/xml
ADDHEADER Accept-Language: en-us
ADDHEADER Brief: t
ADDHEADER range: rows=0-24
SEARCH /$mailbox$/$USER$/Inbox/
 $filepath$IE5_4323_ReturntoInbox_data.1
DELHEADER content-type:
DELHEADER Accept-Language:
DELHEADER Brief:
DELHEADER range:
DELHEADER Referer:
```

```
DELHEADER User-Agent:
DELHEADER Host:
SET $senduser1$ SEQULIST(c:\esp\davscripts\4312\smtpusers.txt)
SET $senduser2$ SEQULIST(c:\esp\davscripts\4312\smtpusers.txt)
SET $senduser3$ SEQULIST(c:\esp\davscripts\4312\smtpusers.txt)
SET $senduser4$ SEQULIST(c:\esp\davscripts\4312\smtpusers.txt)
SET $month$ 06
SET $day$ RANDNUMBER (01,28)
SET $time$ RANDNUMBER (9,12)
SET $folders$ RANDLIST(c:\esp\davscripts\4312\folders.txt)
SKIP 96
```

◆ ◆ ◆ ◆