

---

# MailEnable

## Enterprise Infrastructure Configuration Guide

---

<b>1</b>	<b>Purpose .....</b>	<b>2</b>
<b>2</b>	<b>Storage overview.....</b>	<b>3</b>
2.1	Log file storage .....	3
2.2	Mail storage .....	3
2.3	Mail queues .....	3
2.4	Configuration Storage.....	3
2.5	Operating system .....	3
2.6	Program files.....	3
2.7	Storage area I/O profiles .....	3
<b>3</b>	<b>Server Implementations.....</b>	<b>5</b>
3.1	Assumptions.....	5
3.2	Configurations.....	5
3.2.1	Single disk configuration .....	5
3.2.2	Dual disk channel configuration .....	5
3.2.3	Three disk channel configuration .....	5
3.2.4	Four disk channel configuration .....	6
<b>4</b>	<b>Mail server clustering .....</b>	<b>7</b>
4.1	Scalability clustering.....	7
4.2	Service distribution .....	7
4.3	Redundancy clustering.....	8
<b>5</b>	<b>Additional Considerations.....</b>	<b>10</b>
5.1	Operating system information.....	10
5.2	Mail access protocol scalability considerations .....	10
<b>6</b>	<b>Troubleshooting disk I/O issues .....</b>	<b>11</b>
6.1	Monitoring the I/O subsystem .....	11
6.2	Resolving I/O subsystem issues.....	11

---

# 1 Purpose

The purpose of this guide is to outline how to configure and tune MailEnable to facilitate scalability. This is particularly relevant where the mail system is intended to host thousands of mailboxes.

The document outlines considerations for gaining the best performance from MailEnable by considering disk I/O activity and disk capacity planning (the two undisputed considerations of running any mail server). The document also provides a brief insight into redundancy clustering.

## 2 Storage overview

MailEnable's architecture facilitates at least 5 logical data stores. In effect, these data stores are logical repositories for writing and reading information from disk. These stores are outlined under the following respective headings:

### 2.1 Log file storage

MailEnable log files are written to whenever any activity occurs on the system. Log file activity increases as more messages pass through the system. When MailEnable writes to the log files, all write activity to a single file must be serialized.

### 2.2 Mail storage

MailEnable stores its messages in a format called MDIR. This format is essentially a structure of directories (representing folders) and each message represented as a file. Enterprise Edition provides folder indexing and this can significantly reduce the I/O associated with accessing the message store.

Even with indexing, message store access contributes heavily to disk I/O. The disk I/O is dependant on the number and size of user mailboxes stored on the server.

### 2.3 Mail queues

The mail queues store information that is waiting for delivery to other mail systems or for processing by MailEnable connectors. For each MailEnable connector, there are usually two queues, the inbound queue and the outgoing queue. Disk I/O activity on the queues will increase at the rate that messages are sent and received by the mail system. Disk activity can be considered to be comprised of read and writes in equal proportions.

Since the queue information is a temporary store it is less important to have fault redundancy for the queues themselves i.e. if a disk containing the queues fails, only the information in the queues will be lost. The impact of this loss depends on your obligations to the end users in terms of data loss.

### 2.4 Configuration Storage

MailEnable stores some configuration in text files under the CONFIG directory.

This information is accessed regularly, since the files are accessed whenever messages pass through MailEnable. Larger systems (greater than 5,000 mailboxes) should consider using a database to store this information as it is more efficient at providing access to discrete transactional data. If tab-delimited storage is used for larger systems, the configuration file sizes can become large, and accessing the files will generate moderate read activity to the disk. If a database is used, either leverage the database from another server or should follow the vendor's suggestions as to I/O considerations.

### 2.5 Operating system

### 2.6 Program files

### 2.7 Storage area I/O profiles

The following table outlines typical MailEnable data and shows a profile of disk I/O intensity and characterization.

<b>Data Type</b>	<b>I/O Intensity</b>	<b>Write Activity</b>	<b>Read Activity</b>	<b>Description</b>
Log File Storage	Medium	95%	5%	Various MailEnable log files
Mail Store	High	40%	60%	Mail messages
Queues	High	50%	50%	In transit messages
Configuration Store	Medium	5%	95%	Text files to store configuration settings
O/S	N/A	N/A	N/A	Operating system
Program Files	Low	0%	100%	Program files

As can be seen from the table above, each of the MailEnable data types have different profiles for disk I/O utilization. The values shown in the above tables are provided as estimates only.

## 3 Server Implementations

The previous section provided an insight into the logical disk I/O areas of MailEnable and their respective profiles. Some general principals can be applied to provide guidance in both logical and physical hardware configurations.

### 3.1 Assumptions

The suggestions are supplied for server configurations under the assumption that they are functioning as a mail storage server – i.e. intended use. If the server is intended to function as a relay server or is only used for routing, the storage requirements may not be the same (particularly in the area of the message store).

### 3.2 Configurations

Each of the MailEnable data stores should be mapped to a discrete I/O subsystem that best suits the type of access required. In most cases available hardware, cost, size will constrain such implementations.

#### 3.2.1 Single disk configuration

Single disk configuration stores all data and logs on a single disk. Ideally, the disk would be partitioned to have a separate logical volume (logical disk) for each of the data types mentioned in the previous table. The disk would be partitioned with 6 separate partitions for Operating System, Program Files, Queues, Message Store, Log Files, and Configuration Repository). The reason for placing each on its own partition is to facilitate future expansion. This will make it possible to easily introduce a larger disk array as a replacement for that logical disk, or extend the volume should space be available to do so.

Disk No.	Type	RAID	Description
Disk 1	IDE or SCSI	N/A	Since only 1 disk is available, all information will be on a single disk.

#### 3.2.2 Dual disk channel configuration

For dual disk implementations it is best to divide the data stores across the disks.

Disk No.	Type	RAID	Description
Disk 1:	IDE or SCSI	N/A	Operating System, Queues, Logs, Configuration, Programs
Disk 2:	IDE or SCSI	1 or 5	Message Store

#### 3.2.3 Three disk channel configuration

This implementation should be adopted as a minimum implementation to host over 10,000 mailboxes. The data stores are divided across different disk channels, providing much more efficient data throughput.

Disk No.	Type	RAID	Description
Disk 1:	IDE or SCSI	N/A	Operating System, Logs, Configuration, Programs
Disk 2:	IDE or SCSI	1 or 5	Queues
Disk 3:	IDE or SCSI	1 or 5	Message Store

### 3.2.4 Four disk channel configuration

For four disk channel implementations, split each of data stores across each disk subsystem.

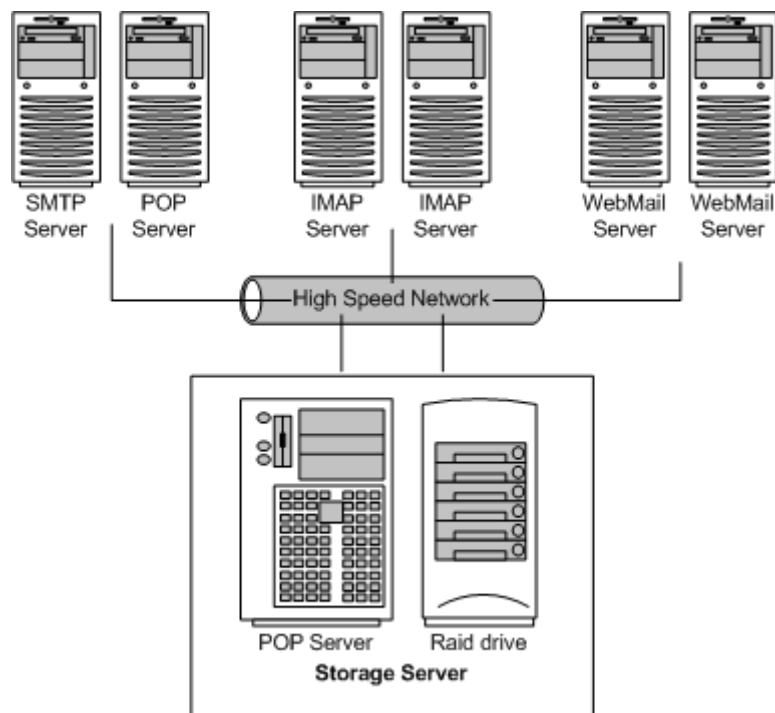
Disk No.	Type	RAID	Description
Disk 1:	IDE or SCSI	N/A	Operating System Queues, Programs, Logs
Disk 2:	IDE or SCSI	1 or 5	Configuration
Disk 3:	IDE or SCSI	1 or 5	Queues
Disk 4:	IDE or SCSI	1 or 5	Message Store

## 4 Mail server clustering

MailEnable’s architecture facilitates server clustering. There are three primary reasons for implementing clustering with MailEnable; scalability, service distribution (load balancing) and redundancy.

### 4.1 Scalability clustering

Because MailEnable stores its configuration and message store on file services, a separate server can be configured to share the same message store and configuration data. This allows organizations to configure separate servers to act as front-end servers, leaving the message store and configuration on an alternate server. This approach means new servers can be added to the environment to accommodate load. This is particularly relevant if using antivirus or content filtering, since these services can be resource intensive.

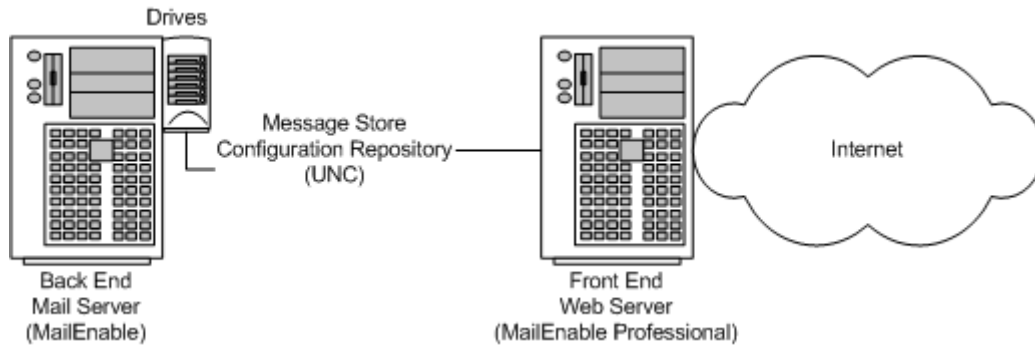


In the above example, multiple front-end servers are clustered to a shared backend storage server. The storage server is configured with a high performance disk array to provide effective throughput to the front-end servers. It is also important that the network throughput is adequate to service the requests of the front-end servers.

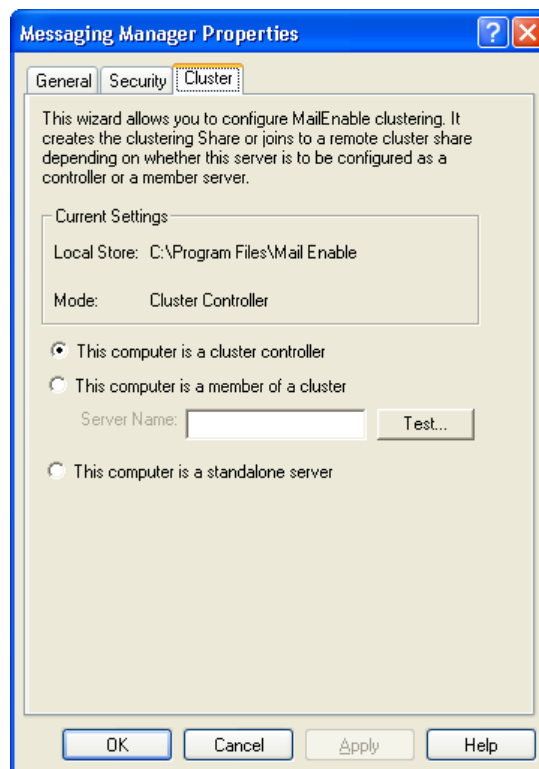
### 4.2 Service distribution

In some cases, it may be desirable to distribute functions across multiple servers. For example, it may be necessary to have a web server providing web mail functions only (with a back-end sever providing back-end messaging functions).

The following procedure explains how to share configuration and data amongst MailEnable servers.



MailEnable Enterprise Edition can define the role of a server as being a cluster controller or as a member. This is configured in the MailEnable Administration program and is outlined in detail in the Enterprise Edition Manual. The configuration screen is shown below:

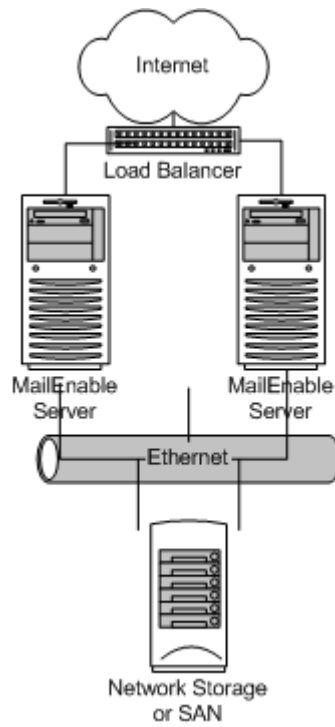


When a server is configured as a cluster controller, a hidden share is created on that server as called MAILENABLE\$. Any member server configured to connect to the controller server will connect to that share. In the example above, the back-end server would be the cluster controller with the front-end server being configured as a member server.

### 4.3 Redundancy clustering

Clustering may be implemented to introduce redundancy into the system; with the objective of mitigating the risk of system failure or reducing downtime. This usually means having two servers configured to access a shared redundant disk or network attached storage (NAS).

A clustered IP address is provided for the front-end interface to the mail servers. Mail clients are configured to connect to the clustered IP address of the server. The affinity of the servers is distributed by a hardware IP load balancer or by Windows Clustering Services. An example implementation is shown below:



## **5 Additional Considerations**

### **5.1 Operating system information**

It is best to keep the operating system on a separate disk to applications.

It is also beneficial to keep the operating systems paging file on a discrete disk that is not impacted by other concurrent system activity.

Once the operating system is loaded, there is very little activity in terms of disk access. The primary considerations for storage are in respect to redundancy. For simplicity, we will assume that the operating system and the paging file are stored on the same disk – on the basis that the operating system itself will perform minimal I/O (other than paging – which will of course involve the paging file).

### **5.2 Mail access protocol scalability considerations**

Some mail access protocols are more intensive on the server than others. The POP Protocol is the least intensive because it involves pulling messages off the server. Other protocols like HTTPMail, web mail and IMAP are more intensive on the server because each mail client must enumerate server side messages and folders.

## 6 Troubleshooting disk I/O issues

This section deals with how to troubleshoot I/O issues within a MailEnable environment.

### 6.1 Monitoring the I/O subsystem

The simplest way to diagnose the I/O utilization of the server is with the Windows Performance Monitor (under the Administrative Tools Program Group).

The performance counter is Physical Disk| %Disk Time|Disk Instance.

In summary, the higher the percentage disk time, the busier the system is processing requests for I/O to that disk. This number should only peak at bursts - a sustained horizontal line at 100% usage indicates I/O subsystem performance issues.

### 6.2 Resolving I/O subsystem issues

Often I/O subsystem issues can result from a system being abused or a system crash occurring. Such situations will cause a backlog of mail in the queues, and moving this mail will place the I/O subsystem under extreme constant load (perhaps affecting other services). In this situation, it becomes apparent why separating the message store from the queues is advantageous.

In some cases, it may be necessary to feed the queues. This is done by stopping the MTA and copying all the command files from the inbound queue into another directory (effectively hiding them from the MTA – but leaving the actual message contents in the Messages folder). It is then possible to grab bunches of the command messages and drop them back into the Inbound Queue folder and monitor how quickly the messages move. This will control and regulate the I/O usage, potentially allowing messages to be processed more efficiently (at least until the queues clear).