Chapter 1

An Evolution of Server Management

I've always been told to begin at the beginning. So that is what I will do.

When you picked up this book at your local bookstore, or looked at it by browsing to your favorite online bookseller, you already knew that you needed to learn how to evaluate Exchange Server's performance using System Center Operations Manager. You may even have had a desire to know how you could do it yourself. Rest assured, this book is the solution. You'll not only understand what Operations Manager can do for monitoring Exchange in your environment, you'll also know how to perform many of the same tests by hand should you need to. ☺

My plan is for you to achieve the following goals by the end of this book:

- Understand the origins of Microsoft Exchange Server 2007 and Microsoft System Center Operations Manager 2007 (OpsMgr)
- Understand how server health and performance management were once done manually and why tools such as OpsMgr are so valuable
- Understand the Exchange Server environment and how it interacts with its dependencies (such as Active Directory and Windows Server)
- Understand the OpsMgr environment and how it interacts with its dependencies (such as SQL Server and Active Directory)
- Master the installation and configuration of OpsMgr
- Effectively monitor the health and performance of your Exchange Server(s) and its dependencies using OpsMgr (including monitoring the OpsMgr server).

Along with knowing what you are going to learn, it is also worthwhile to know what I will not cover here: I will not show you how to install Exchange Server or Windows Server or how to design an Active Directory infrastructure. Those things should've been done beforehand.

My focus will be on Exchange Server 2007. However, where practical, I will also present relevant information regarding Exchange Server 2003. On the outside, there are huge differences between the different versions of the products, but under the covers there are some amazing similarities, especially when it comes to monitoring for system and server health.

Exchange Server 2007: A Little History

Microsoft Exchange Server started its life as Exchange Server 4.0, shortly after Microsoft discovered the Internet. Prior to Exchange, Microsoft had another email product called Microsoft Mail (MS-Mail). The last version of MS-Mail was 3.5 and it was released in September 1995. Microsoft hoped that using 4.0 for the first version of Exchange would convince users to migrate quickly.

Prior to Exchange Server 4.0, internally Microsoft was using an email system based on another operating system it supported, Xenix. As of Exchange Server 4.0, Microsoft's internal email system moved to Windows.

Regardless of the version number, Exchange 4.0 was a version 1.0 product and was not an upgrade to MS-Mail 3.5. Getting from MS-Mail 3.5 to Exchange Server 4.0 required a full-scale migration. Exchange 4.0 also had a number of severe limitations and somewhat limited functionality (especially when compared to the products available today). However, it introduced many capabilities that are absolutely recognizable even in today's much more mature product. Exchange Server 4.0 was followed by Exchange Server 5.0 and then by Exchange Server 5.5.

Exchange Server 5.5 was the first version of the Exchange product line to really take off. Exchange Server 5.5, especially by service pack 3, was a very usable product. It is estimated that, at this writing (late 2008), more than 15 percent of Exchange customers are still using Exchange Server 5.5—even though three major versions of Exchange have been released since Exchange Server 5.5. It was, and still is, a stable product.

Exchange Server 5.5 was the first version of Exchange to truly support Internet mail; it did so with the Internet Mail Connector (IMC). Therefore, if companies are still using older versions of Exchange Server, they are probably not connected to the Internet, at least for email! It's doubtful that very many copies of Exchange Server prior to version 5.5 are still in operation.

The releases of Exchange Server up to and including Exchange Server 5.5 all shared a common feature: a directory that was maintained by the Exchange Server itself. For the purpose of this discussion, a directory is a list of items (such as subscribers, distribution lists, contacts, public folders, routing tables, etc.) and all of the attributes (information and data) about each of those items. All of the items and their attributes were stored in a database maintained by Exchange—and this database was not part of the operating system.

In a revolutionary change, Exchange 2000 Server moved the directory into the brand-new Active Directory. Starting with Exchange 2000 Server, Exchange requires Active Directory and heavily uses some of its features and functionality.

Since the Active Directory database was originally based on the Exchange database technology, it was a natural step.

There were other revolutionary changes in Exchange 2000 Server, among them the inclusion of Conferencing Server (which provided additional collaboration capabilities to Exchange) as well as instant messaging capabilities. These changes were moved back out of Exchange Server and into a separate product (Microsoft Live Communications Server) as of Exchange Server 2003. (Microsoft Live Communications Server was renamed Office Communications Server with its release in 2007.)

As a minor aside, the change in name between all prior versions of Exchange Server and Exchange 2000 Server was another modification that lasted for only a single product release.

The movement to Active Directory with Exchange 2000 Server also significantly changed the administrative model for Exchange. Historically, an Exchange administrator could control everything about an Exchange site—the subscribers, mailboxes, distribution lists, server configuration, and so on. With the integration to Active Directory, however, that is no longer always the case. Exchange management is now separate from user and group management (groups include distribution groups, which were known as distribution lists in Exchange 5.5 and prior versions).

In larger companies, this administrative split makes very good sense because messaging and collaboration are just applications. Being an administrator of the messaging system shouldn't

provide application-level administrators full administrative control of the computer network too. For many (if not most) smaller companies, the distinction is meaningless, and it often doesn't seem to make any sense—why is it now necessary to use two programs for administration, whereas in Exchange 5.5 and earlier, everything could be done in a single program?

Thankfully, for these smaller environments, the multiple administrative consoles may be merged into a single custom console providing the single point of administration that these companies are used to.

Exchange 2000 Server represented a major change to the core architecture of the Exchange product, and in some ways the integration with Active Directory again made it a version 1.0 product. Even so, Exchange 2000 Server was a significant improvement over Exchange Server 5.5. Many of the Exchange 2000 Server deficiencies were corrected in Exchange Server 2003. Exchange Server 2003 proved itself to be an extremely stable and feature-rich platform.

With Exchange Server 2007, after service pack 1, Exchange Server again has regained its position as the preeminent messaging application. The changes in Exchange Server 2007 from Exchange Server 2003 primarily revolve around providing the system administrator with additional control and allowing Exchange Server to increase performance in 64-bit environments.

System Center Operations Manager 2007: A Little History

OnePoint Operations Manager was originally written and sold by a company named Mission Critical Software (which no longer exists). The product was released in October 1999. According to a Security Exchange Commission prospectus for Mission Critical Software (MCS) dated November 12, 1999:

> *We provide systems administration and operations management software products for corporate and Internet-based Windows NT networks. Our OnePoint product suite is designed to improve the reliability, performance and security of even the most complex computing environments by simplifying and automating key systems management functions. Our products can be deployed quickly, are based on an open and extensible architecture and are easy to use.*

MCS had a suite of products known as OnePoint Enterprise Administrator that included other pieces, including its Enterprise Domain Administrator (for migrating to Windows 2000 Active Directory), Enterprise Exchange Administrator (for synchronizing Exchange Server 5.5 and Active Directory), and more. The OnePoint Operations Manager product was, according to several published accounts, used within Microsoft itself by Microsoft IT.

Even in the beginning, OnePoint Operations Manager was management pack focused. The product was designed to be expanded to include information about applications and application systems as they were released.

MCS was purchased by NetIQ in May 2000. NetIQ and MCS were competitors in several product areas, including OnePoint Operations Manager versus NetIQ's Application Manager. The Application Manager solution was primarily script based, as opposed to the rule-based environment of Operations Manager.

With the release of Windows 2000 and Active Directory, Microsoft was trying to extend its reach into enterprise-sized companies. With Windows 2000 and Active Directory (AD), Microsoft had demonstrated that the server capabilities were now present to compete with mainframes and UNIX systems, but Microsoft had no system management solution. Microsoft reviewed the market and eventually decided to purchase OnePoint Operations Manager from

NetIQ. For NetIQ, this removed the burden of competing products from their product line. For Microsoft, Operations Manager was fairly easy to bring in-house because Microsoft IT already used the product and it was based on core Microsoft technologies: WMI, COM+, and SQL Server.

In October 2000, Microsoft completed the purchase of Operations Manager from NetIQ, although at that time the relationship was presented as a "partnership." One can only presume that NetIQ discovered, as other Microsoft software partners have in the past, that Microsoft always wins in partnerships.

Based on the terms of the deal, Microsoft and NetIQ were going to do joint development and joint marketing for the Operations Manager product. For several years, NetIQ did release add-on management packs and software for the renamed Microsoft Operations Manager (MOM). After the terms of the deal passed, NetIQ refocused on security and its AppManager product.

MOM 2000 was released in June 2001, basically a rebranding of the NetIQ product, with a major service pack in early 2003 that began to move the product to "the Microsoft way." MOM 2005 added a number of features important to large enterprises and improved the administration and deployment options of the product. Along the way were dozens, if not hundreds, of management packs that were released by various partners and Microsoft itself.

System Center Operations Manager 2007, also known as SCOM and OpsMgr, is the next evolutionary release of Operations Manager. Enhancements to OpsMgr have included "end-to-end" monitoring, which adds the monitoring of client workstations to the suite. A long-awaited addition is Audit Collection Services (ACS), which allows event log aggregation, intelligent reporting, and reduction across an enterprise.

Now that I've talked about the history of the products, I want to tell you why you might want to use them together.

Monitoring: Do We Care?

In the beginning, Windows-based servers were few and far between. They were comparatively inexpensive. They were often hidden (the MIS department in most companies regarded Windows as a toy). Most companies had only one, two, or a few servers to begin with and they were dedicated to "important" functions for a single department. Compared to the performance of today's hardware and software, those servers were quite slow and had a distinct lack of capabilities. But compared to the available options—mini-computers and mainframe computers and the expensive squads of dedicated personnel required to operate them—the Windows-based servers were cheap and easy to deploy and operate. So, the number of Windows-based servers grew and grew.

As the number of server-based applications increased and the number of people within an enterprise using those applications increased, the obvious happened. The applications and the servers that they were running upon became more and more critical to the enterprise.

Today's servers are as powerful as the mainframes of years past. In fact, they are often much more powerful. In the last 15 or 20 years it has been shown that the reliability, resiliency, and availability requirements for Windows servers has reached the same levels once required for the mainframes. In fact, mainframes are now truly dinosaurs. Most companies that once had mainframes have moved to farms of servers, generally Windows or UNIX/Linux based.

Many of the operational requirements that once were only a part of mainframe operations have now become requirements in the Windows server world too. These include monitoring the health of servers (uptime, application availability, processor and memory utilization, message

velocity and queue depths, etc.) as well as the baseline and ongoing measurements that allow companies to track the change in various types of resource utilization.

These needs continue to become more sophisticated. The monitoring "hooks" into the applications and the operating system become deeper and more complex. It is no longer enough to be able to say, "Yep, that server is up." You also need to be able to say, "That server is up and applications A, B, and C are responding properly to user input within specified parameters." And when the applications are responding outside of acceptable parameters, appropriate personnel need to be notified and other specific actions taken. This is what System Center Operations Manager 2007 does.

Commonly referred to as SCOM (although the recommended abbreviation from Microsoft is OpsMgr, which is the term I will use throughout this book), this tool provides end-to-end monitoring (also known as top-to-bottom or A-to-Z monitoring) of specific applications and services as well as basic server health. OpsMgr provides its capabilities using plug-ins known as management packs (MPs). A management pack is a targeted solution for a given application that defines the parameters for the good health of that target.

OpsMgr is a best-of-breed solution. Each group within Microsoft is committed to releasing a management pack for its server solution within 90 days of that solution being released. This management pack provides expert guidance from the teams that generated the server solution as to the appropriate parameters that define the health of the solution. Typical management packs include Windows Server, SQL Server, Active Directory, and (the item of primary interest) Exchange Server. I will touch on all of these management packs in this book.

OpsMgr also provides automation as part of the management packs, which are designed to reduce the number of mindless routine tasks that a system administrator is required to perform. This includes such simple-in-concept (but difficult-to-implement) ideas as synthetic transactions, where OpsMgr appears to be a user to an application in order to produce a pass/fail result and responsiveness statistics, as well as automated collection and reduction of event logs from servers and from workstations.

OpsMgr also includes interfaces that allow it to provide cross-platform support for alternate reporting mechanisms such as Web Services Management (WS-Man, commonly used by infrastructure hardware such as SAN, NAS, fiber, and backup), Syslog (commonly used by Linux/UNIX servers), and Simple Network Management Protocol (SNMP, commonly used by network devices such as routers and switches).

To repeat the original question: monitoring—do we care? If your company is like most of the companies today that depend on Windows-based servers for their corporate computing needs, the answer is a resounding *yes*. You want a solution that helps you keep your systems running smoothly, providing high reliability and resilience to your system administrators and high availability to your user communities.

OpsMgr is for you. And I'll explain how to use it to keep Exchange running right along with all the other fun services and applications that support it.

Health Monitoring in the Old Days…

Health monitoring, service monitoring, and so on are certainly not new ideas, not even in the Windows-based server world. However, monitoring used to be much more manual than it is now with OpsMgr. A suite of tools, each providing a singular piece of information, was once used to determine the health of a server and its various services and applications.

Today, there are many utilities that work to automate the simplest of these tools. However, a fully configured system that automates almost everything is difficult to find. OpsMgr is one of the few fully featured systems and is arguably the most affordable.

Many of the tools used to test server health and application availability originally grew out of the UNIX world. While graphical user interface (GUI) versions are available in the Windows world (and in the UNIX/Linux worlds too), most system administrators are more familiar with the command-line versions that provide quick answers. Also, the command-line versions of the tools are provided with just about every operating system. The GUI versions are specific to a given operating system and their features and functionality vary greatly.

To understand some of the services provided by OpsMgr, let's discuss what it entails to execute some of these tests manually and individually. Although I refer to them as old-style, you may still find yourself using these tools on a regular basis.

Tools Background

Whenever a user accesses an application on a server, you can be certain that a specific set of activities are executed by the user application before it can utilize the server-based application:

- Finding the network address of the server
- Determining a way for the client to access the server
- Attempting to access the server
- Connecting to the application on the server

The first four tools I present map directly to that list.

The `nslookup.exe` tool takes a server name and turns it into an IP address (this is called name resolution). The `tracert.exe` tool (named `traceroute` on UNIX/Linux) takes the IP address of the client computer and finds a path through the network to the server computer (this is typically referred to as a route). The `ping.exe` utility sends a particular type of network message from the client computer through the network route displayed by the `tracert.exe` utility and attempts to determine whether the server is alive. Finally, the `telnet.exe` utility attempts to connect to a specific application on the server (by specifying a type of connection object known as a port).

NSLOOKUP

The name resolution performed by `nslookup.exe` is not unique to that utility. In fact, all of the other utilities named earlier will perform the same name resolution. However, they do not provide the detailed name resolution information and capabilities provided by `nslookup.exe`.

```
C:\>nslookup win2003-exch.essential.local
Server:  win2003-dc.essential.local
Address:  192.168.1.55

Name:    win2003-exch.essential.local
Address:  192.168.1.65

C:\>
```

As shown here, the user has opened a command-prompt window (`cmd.exe`). Within the command prompt, the user has entered **`nslookup`** followed by the name of a particular server. In this case, the server's name is win2003-exch.essential.local. A number of pieces of information are significant in the output of the command. The first line of output is the name of the server that is providing the name resolution service, win2003-dc.essential.local. The second line is the IP address of that server. The third line is blank. The fourth line repeats the name input by the user and the fifth line gives the IP address for that computer.

In the case of any error, the `nslookup.exe` utility would provide detailed error information, as shown in this sample:

```
C:\>nslookup win2009-exch.essential.local
Server:  win2003-dc.essential.local
Address:  192.168.1.55

*** win2003-dc.essential.local can't find win2009-exch.essential.local:
   Non-existent domain

C:\>
```

Here we see an error provided when an invalid server name is supplied to the `nslookup.exe` utility.

The `nslookup.exe` utility is often the first utility employed in debugging a network access situation. It ensures that an originating computer can determine how to access a destination server. The `nslookup.exe` utility utilizes the Domain Name System (DNS) to perform its queries. DNS is described in RFC 1035.

NOTE *RFC* stands for *Request for Comments*, and there is an entire library of RFC documents that is maintained by the Internet Engineering Task Force (IETF), the group responsible for standards on the Internet. Some folks say that *RFC* actually means *Requirement for Conformance* because if your applications do not follow RFCs, you will have issues interoperating with other computer systems on the Internet. To access the RFC repository, visit `www.ietf.org/rfc` and enter the number of the RFC in which you are interested.

Ping

The `ping.exe` command uses a specific type of IP message called Internet Control Message Protocol (ICMP, defined in RFC 791) to send a message from the source computer to the destination server. The destination server then replies. This command allows a system administrator to determine whether a remote server is available and how long responses from the destination computer take to return to the source computer.

```
C:\>ping win2003-dc.essential.local

Pinging win2003-dc.essential.local [192.168.1.55] with 32 bytes of data:

Reply from 192.168.1.55: bytes=32 time<1ms TTL=128
Reply from 192.168.1.55: bytes=32 time<1ms TTL=128
Reply from 192.168.1.55: bytes=32 time<1ms TTL=128
```

```
Reply from 192.168.1.55: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.55:
   Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
   Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

When using the `ping.exe` utility, you should pay attention to loss percentages and average round-trip times. Should the loss percentage be greater than zero on a regular basis, there is a network issue that needs to be addressed. Similarly, if average round-trip times exceed baseline measurements, there is a network issue that needs to be addressed.

The `ping.exe` utility is often the second utility employed when debugging a network access situation. It is used to verify connectivity between an originating computer and a destination server. If `nslookup.exe` succeeds but `ping.exe` fails, the system administrator can reach three potential conclusions:

- The remote server is down.
- A route does not exist from the originating computer to the destination server.
- A hardware problem exists between the two computers.

The `tracert.exe` and `telnet.exe` utilities assist in narrowing down the set of potential problems.

TRACERT (TRACEROUTE)

Similar to the `ping.exe` command, the `tracert.exe` command uses a specific type of IP message called Internet Control Message Protocol (ICMP, defined in RFC 791) to send a message from the source computer to the destination server. The destination server then replies. However, instead of attempting to access the destination server with a single IP packet, `tracert.exe` attempts to access the destination server one hop at a time. Each hop along the path represents a piece of hardware known as a router. A router accepts a packet of information, decides where it should go next, and moves it to the next network. Once the packet of information reaches the destination network, it is delivered to the destination server.

In the following sample, I have asked the `tracert.exe` utility to display the route to `yahoo.com`:

```
C:\>tracert yahoo.com

Tracing route to yahoo.com [216.109.112.135]
over a maximum of 30 hops:

  1    2 ms    2 ms    2 ms  va-x-y-z.sta.embarqhsd.net [69.x.y.z]
  2   42 ms   43 ms   44 ms  va-z-y-x.sta.embarqhsd.net [69.z.y.x]
  3   42 ms   43 ms   43 ms  208.33.154.233
  4   53 ms   58 ms   56 ms  ge-6-20.car1.Washington1.Level3.net [4.79.18.209]
  5   55 ms   59 ms   67 ms  4.79.228.2
```

```
  6    54 ms    55 ms    55 ms  ge-1-0-0-p100.msr1.dcn.yahoo.com [216.115.108.41]
  7   114 ms    55 ms    57 ms  ge2-2.bas2-m.dcn.yahoo.com [216.109.120.153]
  8    62 ms    55 ms    57 ms  w2.rc.vip.dcn.yahoo.com [216.109.112.135]

Trace complete.

C:\>
```

The first two lines of output are obfuscated a little bit—that is, I intentionally changed the real information in case someone would try to access my home system! So, the first line represents the hop from my workstation to my home router. The number in the first column is the number of hops traversed so far. The next three numbers are how long it took to get a reply from that router. The utility tries three times. Sometimes, an asterisk (*) may be shown in the reply columns. This means that the destination router did not return a result. The rest of the line is the name of the destination router (if available) and its IP address.

The `tracert.exe` utility will, by default, attempt only 30 hops. If you have more than 30 hops, there are likely problems in your network. A large number of asterisks in the output from `tracert.exe` also indicates a problem in your network.

Telnet

At this point in debugging a problem, we've had `nslookup.exe` verify that name resolution is working in the environment, `ping.exe` verify that the destination server is up, and `tracert.exe` verify the path to the destination server. However, so far nothing is known about the application—is it responding or not? This is where the `telnet.exe` utility comes into play.

If the application running on the destination server utilizes a fixed port (that is, a standard connection point object to be used by the IP protocol), then the `telnet.exe` application determines if that port is available. This allows easy testing for many of the applications that a system administrator may be interested in, including HTTP, SMTP, POP3, and others. Here is a simple example:

```
C:\>telnet Win2003-exch.essential.local smtp
220 win2003-exch.theessentialexchange.com Microsoft ESMTP MAIL Service,
   Version: 6.0.3790.3959 ready at  Tue, 18 Mar 2008 13:49:01 -0400
help
214-This server supports the following commands:
214 HELO EHLO STARTTLS RCPT DATA RSET MAIL QUIT HELP AUTH TURN ETRN BDAT VRFY
quit
221 2.0.0 win2003-exch.theessentialexchange.com Service closing
   transmission channel


Connection to host lost.

C:\>
```

In this example the `telnet.exe` utility is passed the name of an Exchange server along with the protocol it should use to connect to that server. In this case, the protocol named is SMTP (for Simple Mail Transfer Protocol, which is described in RFC 2821). After some header information

is displayed, the user enters the **help** command, which causes the remote server to display the supported commands, and finally the user enters **quit** to terminate the connection.

NOTE A sharp-eyed reviewer noted that the SMTP header from Microsoft Exchange says ESMTP, not just SMTP, and asked if that was important. Well, it is and it isn't. In this case, it isn't. However, sometimes it can be. The *E* in *ESMTP* stands for *Extended*. So ESMTP is Extended Simple Mail Transfer Protocol. This just means that an ESMTP server understands more commands and has more features than a non-extended SMTP server. For the usage in this book, you can consider ESMTP and SMTP to be the same thing.

What Other Tools Are Available?

Given the commands and tools covered thus far, it is possible to verify that a network knows how to determine the network identifier (IP address) for a destination server, that there is a known network route to the destination server, that the server is available, and that the server is responding to connection attempts on a given port.

In the old days, this was pretty much all that was available. To get further information, a system administrator would generally have to access a specific application or visit the server room.

There are many inexpensive applications on the market today that automate these four tools—and provide some type of notification to a system administrator—and that's pretty much all that they do. Of course, OpsMgr does these things too, and much more.

Remote Desktop Protocol (RDP, also known as T/S for Terminal Services), introduced as part of the core operating system in Windows Server 2000, generally allows the system administrator to be lazy—uh, that is, to be more efficient by not having to actually walk to the server room.

However, that doesn't provide all of the required information. The next section goes into other resources that may provide additional information about server and application health.

Additional Health Resources

When an Internet Service Provider (ISP) hosts a server, the ISP will often warrant that what it supplies is "ping, power, and pipe." That is, it supplies notification of whether the server is up or down by using `ping.exe`, power to plug in the server, and Internet connectivity.

As discussed earlier, that's only the beginning. In the following sections, I'll present some of the other factors that make server health an interesting topic of discussion.

Disk Space Utilization and Connection Velocity

Anyone who pays attention to discussions about websites and the amount of traffic that a website has received is familiar with the concept of weblogs. Weblogs store information about each hit that a website receives, such as the number of bytes for the request, the name of the file requested, the originating computer's IP address, whether or not the server could respond to the request, and how long the server took to respond to the request. What may not be immediately obvious is that weblogs take up disk space. For a busy website, the weblogs take up *huge* amounts of space. Weblogs can be much larger than the space consumed by the website itself.

However, it can be argued that the information contained within the weblogs is just as important as the website! Many websites are marketing tools and information resources. Knowing how

well they perform is critical to their designers and supporting departments. So weblogs grow and grow, and they have to be managed. But how often do they need to be managed? What kind of management do they need?

Weblogs are not the only kind of information or file that constantly grows and needs to be managed. Think, for example, of databases, of music collections, of client documentation, of just about any kind of structured information. Once you begin to accumulate data, it never seems to stop growing. Also, for some configurations of a server, if the boot/operating system volume on the server fills up, the server will crash. That is definitely not a desirable outcome.

These facts indicate another piece of health information about a server that is important to know: disk space utilization for all volumes on that server, and perhaps not only disk space utilization, but utilization by type of space used (music, document, database, etc.). This is a type of monitoring that is like insurance; it may never be needed, but once the need arises, you are always glad you have it.

Returning to the website example, another key piece of information that can be garnered from the weblogs is called the connection velocity. This defines how quickly connections are coming into a website (or into any application). Should this number drastically increase, it may indicate a rise in popularity of the website—a marketing success! Or, disastrously, it may indicate that a website is undergoing a denial-of-service attack. Should connections drop to zero, it can indicate a number of bad things, from a poorly deployed upgrade to the website to a possibly defaced website that now redirects to a pornographic site!

Regardless, the system administrator needs to know when the connection velocity changes by a significant amount. And while it can be determined by regularly scanning the weblog (or tracking the change in its size), there are more efficient ways of obtaining this information, by looking at internal counters maintained by IIS. OpsMgr can do this. Many other utilities cannot.

WMI

WMI stands for Windows Management Instrumentation. WMI is the Microsoft implementation of something known as CIM, which stands for Common Information Model. CIM (and therefore WMI) is an industry-standard way of representing information about computing objects, including processors, processes, tasks, networks, IP addresses, routers, and switches. There are literally hundreds of WMI objects implemented within modern versions of Windows (WMI was first available in Windows 2000 Server).

WMI provides a schema (that is, a description of the information that is available) and a specification of the format of the data contained within the schema. Within WMI, Microsoft has also defined a simple and standard mechanism for accessing the information contained therein.

What does this mean to the system administrator?

For all practical purposes, it means that the system administrator can likely obtain information about any visible object on a computer system, and can do so either locally or remotely. As a single example, note the information that is contained within the `Win32_Process` class (and this information is available for every single process that is executing on a given computer system):

```
class Win32_Process : CIM_Process
{
  string Caption;
  string CommandLine;
```

```
  string CreationClassName;
  datetime CreationDate;
  string CSCreationClassName;
  string CSName;
  string Description;
  string ExecutablePath;
  uint16 ExecutionState;
  string Handle;
  uint32 HandleCount;
  datetime InstallDate;
  uint64 KernelModeTime;
  uint32 MaximumWorkingSetSize;
  uint32 MinimumWorkingSetSize;
  string Name;
  string OSCreationClassName;
  string OSName;
  uint64 OtherOperationCount;
  uint64 OtherTransferCount;
  uint32 PageFaults;
  uint32 PageFileUsage;
  uint32 ParentProcessId;
  uint32 PeakPageFileUsage;
  uint64 PeakVirtualSize;
  uint32 PeakWorkingSetSize;
  uint32 Priority;
  uint64 PrivatePageCount;
  uint32 ProcessId;
  uint32 QuotaNonPagedPoolUsage;
  uint32 QuotaPagedPoolUsage;
  uint32 QuotaPeakNonPagedPoolUsage;
  uint32 QuotaPeakPagedPoolUsage;
  uint64 ReadOperationCount;
  uint64 ReadTransferCount;
  uint32 SessionId;
  string Status;
  datetime TerminationDate;
  uint32 ThreadCount;
  uint64 UserModeTime;
  uint64 VirtualSize;
  string WindowsVersion;
  uint64 WorkingSetSize;
  uint64 WriteOperationCount;
  uint64 WriteTransferCount;
};
```

The general meaning of many of these fields is immediately obvious. However, for specific definitions of individual fields, you should refer to the Microsoft web page defining those fields.

In this case, that would be: http://msdn2.microsoft.com/en-us/library/aa394372(VS.85).aspx. Accessing this information can also be done very simply, using the Microsoft PowerShell:

```
PS C:\> gwmi win32_process | where { $_.threadcount -gt 50 } | `
   fl name, threadcount, workingsetsize

name         : System
threadcount   : 130
workingsetsize : 16285696

name         : svchost.exe
threadcount   : 58
workingsetsize : 131121152

name         : iexplore.exe
threadcount   : 55
workingsetsize : 311443456

PS C:\>
```

This particular query displays the three tasks currently on my computer that are using more than 50 threads as part of the process and the size of their working set (a particular kind of memory measurement).

When you are monitoring a computer system, a large amount of real-time information that is of interest can be obtained. As in the preceding example query, it can be interesting to know how many tasks are using more than 100MB, or 500MB, or some arbitrary value. WMI is one of the mechanisms available to obtain that type of information in modern Windows operating systems. OpsMgr can do this. Many other utilities cannot.

Interestingly enough, often WMI is not the only mechanism available for obtaining this type of information in Windows. The other mechanism is known as Performance Objects, and it's discussed in the next section.

System Monitor (Perfmon)

Perfmon, also known as System Monitor or Reliability and Performance Monitor (depending on the version of Windows you are using; however, you can always start the tool by clicking Start ➢ Run, entering **perfmon.exe**, and clicking OK), is a tool designed to peer into the depths of a running system and its running applications and allow a tool or system administrator to extract information. This information is presented via a set of performance objects and performance counters. The performance counters are methods and attributes associated with the performance objects.

On the outside, it often appears that Perfmon is just another interface over WMI. However, as I will show in later chapters, there is much information that can be obtained via Perfmon that is not available via WMI, including the infamous Squeaky Lobster I/O data (stay tuned for more information about that!).

Here is another way to get the same thread performance data that we obtained via WMI before:

```
$perfObject = new-object `
   System.Diagnostics.PerformanceCounterCategory("Process")
```

```
$perfObject.GetInstanceNames() | foreach {
       $counter = ($perfObject.GetCounters($_))[0];
       $counter.CounterName = "Thread Count";
       if ($counter.RawValue -gt 50 -and $_ -ne "_Total")
       {
          $_;
          ("threadcount: " + $counter.RawValue);
          $counter.CounterName = "Working Set";
          ("workingsetsize: " + $counter.RawValue);
          " ";
       }
}
```

And this produces a similar output as follows:

```
System
threadcount: 132
workingsetsize: 8744960

svchost#5
threadcount: 51
workingsetsize: 85086208

iexplore
threadcount: 55
workingsetsize: 211428628
```

While the code that produces the output is longer, it also exposes more capabilities. For example, iterating through the method `GetInstanceNames()` will display all of the active processes on a system. Iterating through the contents of `GetCounters()` will display all of the available Performance Counters available for the Process Performance Object.

The Perfmon tool itself will allow a system administrator to easily display a real-time graph or chart of selected performance counters. It also provides a mechanism for storing the real-time information for later analysis and for generating a baseline of performance data. This is useful for comparing changes in system usage against that baseline.

Protocol-Specific Tests

According to Wikipedia, a communications protocol is "a set of rules governing communication between electronic devices" (`http://en.wikipedia.org/wiki/Protocol`). That is a good definition to use when discussing server health and monitoring. For example, the SMTP protocol is a protocol for transferring email between two email servers. It defines, quite specifically, the information that must pass back and forth between a sending server and a receiving server in order to transfer a piece of email. In general terms, the protocol is a conversation, like the conversation between two acquaintances who meet on a street. It begins with a handshake ("Hi Bob" and "Hi Jim"). The handshake is followed by an information transfer ("Bob, did you and Jane hear about Mary? No? Well, please tell Jane that…"). The information transfer may be lengthy or short (short emails, long emails, multiple emails, etc.) and it may go both ways (each server can send emails

to the other). After the information transfer, the conversation ends with a termination sequence ("OK Jim, gotta go. See ya!").

At any point in this conversation, something can go wrong (it may not be Jim whom Bob sees; Bob may not want to hear about Mary or may refuse to pass the information along to Jane; etc.).

When that something wrong gets transferred into the SMTP world of connected computers, it means that email is not being delivered. When email isn't delivered, users get unhappy. This is something else that needs to be monitored. While a port test, as discussed in the section on Telnet earlier in this chapter, can provide some information (e.g., that a service is listening on that particular connection object), it does not communicate whether a service is actually in operation (e.g., the mail spool is out of disk space and is causing the email server to refuse all email).

This situation is certainly not unique to SMTP. Other common protocols must actually be exercised to be tested. HTTP (access to web pages), HTTPS (secure access to web pages), POP3 (a client-based email retrieval protocol), IMAP (another client-based email retrieval protocol), and MAPI (the RPC-based protocol that Outlook uses to communicate with Exchange) are all protocols that require end-to-end testing. This kind of end-to-end testing is fairly rare in the monitoring and health world. OpsMgr does it well.

Usability

Once end-to-end monitoring is in place, it is then possible to begin measuring the usability of a protocol. The concept of usability has already been introduced, from a different perspective, in terms of connection velocity. However, with usability the system administrator is concerned not that connections may be occurring too quickly, but that they may be completing too slowly.

If, for example, an end-to-end test on a web page completes successfully but the web page takes five minutes to load, then the protocol-specific test would pass but the usability test would not. In general, a web page should load in less than 30 seconds. Or perhaps the load could take up to a minute, at the very worst. Beyond that, a user is likely to move on, and not spend any more time on a website.

In the early days of the World Wide Web, the number and size of graphics on a web page were a common problem—when 80 percent of the world was on a dial-up connection! Today, with the ubiquity of broadband Internet service (cable modem and DSL at home, T1s and higher at the office), a slow-loading website is more likely to be some type of issue occurring on the web server (or its database backend) or a network problem. In any case, the issue needs to be addressed.

To be able to fix a problem, the system administrator has to know about the problem. This is another class of problem that OpsMgr can evaluate and communicate to the system administrator.

Resource Availability

In the section "Disk Space Utilization and Connection Velocity" earlier in this chapter, I discussed one resource that needs to retain a certain level of availability: disk space. But disk space certainly isn't the only resource that applications require and that should therefore be monitored.

Along with disk space, applications need real memory, virtual memory, low disk I/O latency, processor availability, network bandwidth, IP addresses, database connections, and so on. Each and every one of these affects whether an application can properly serve the needs of its user

community. For example, a disk may have huge amounts of available space, but there also may be so much I/O (input-output) being executed against the disk that latency (the amount of time it takes to complete a single I/O request) is high enough to cause a serious degradation in user response time. When that is the case, the issue must be investigated and alleviated. As has been shown before, the system administrator needs to know about the problem before it can be addressed.

Resource availability, using several of the technologies already discussed, such as WMI and performance counters, is another class of problems that OpsMgr can evaluate and communicate to a system administrator.

SNMP

Not every device in a network is Windows based. In fact, most of the network infrastructure will certainly not be Windows based. However, that doesn't make them any less important. I am specifically referring to such devices as routers, switches, load balancers, firewalls, and other pieces of computing equipment that are dedicated to a specific purpose (as opposed to being a general-purpose computer or server).

Syslog

Many devices that report their status via SNMP use a reporting mechanism known as Syslog. Syslog was developed on UNIX systems and is similar in nature to the event log available on Windows systems (many would argue that it is much more configurable). OpsMgr has a Syslog server built into the product that allows Syslog information to be natively included into OpsMgr rules.

Many of these devices are based on embedded operating systems and do not have in-depth reporting features built into their functionality. However, many of them do support something called Simple Network Management Protocol (SNMP). SNMP identifies everything by an object identifier(OID). OIDs are globally unique, as every manufacturer of equipment is supposed to request a unique OID prefix.

With SNMP, OpsMgr and other tools can interrogate many network devices as to their health and status. Monitoring the health of the network is just as important as monitoring the health of an individual server or application. Without the network, users will not be able to access the server or application.

Managed vs. Unmanaged

It is common to hear switches (and to a lesser extent, bridges and routers) referred to as managed or unmanaged. A managed switch will provide the system administrator with some mechanism to control and/or modify the behavior of the device and its various ports. An unmanaged switch is just a box where cables can be plugged in.

Most managed switches will support SNMP. Most unmanaged switches will not.

Exchange Server Roles

All Exchange servers are not created equal. However, in releases of Exchange Server prior to Exchange Server 2007, this was difficult to tell from the management console or from the steps executed when installing an Exchange server. In those earlier releases, optimizing an Exchange server for certain tasks took a significant amount of manual effort.

There are two primary reasons for optimizing an Exchange server for executing specific tasks:

- To reduce the attack surface of the server
- To improve the performance of the Exchange server in executing its assigned task

In Exchange Server 2007, Microsoft recognized the need to separate the roles assigned to an Exchange server and made this separation a part of the installation process for Exchange.

How Does the Separation of Server Roles Affect a Small Shop?

It doesn't. It is still possible to combine all Exchange roles onto a single server. In fact, the Small Business Server (SBS) 2008 and Essential Business Server (EBS) 2008 packages from Microsoft do exactly that. Even in larger shops, some consolidation of server roles is likely.

In the next two sections, I'll discuss how individual server roles affect what should be monitored on a given server. As the roles are somewhat different between Exchange Server 2003 and Exchange Server 2007, I will discuss these versions of Exchange separately.

Exchange 2003 and Earlier

Regardless of the task(s) that a server would be used for with Exchange Server 2003, the installation process was the same. All of the pieces to support all of the various sets of functionality for Exchange were installed on all servers. However, after installation, it was common to customize an Exchange server into a number of sets of specific functionality:

- Front-End
  - Outlook Web Access (OWA)
  - RPC/HTTP
  - Legacy protocols
- Back-End
  - Mailbox
  - Public folder
- SMTP relay

SMTP Relay

While many companies used (and still use) an Exchange server as their SMTP relay, this is not required. The SMTP service that is part of Windows Server 2003 is perfectly adequate when an SMTP relay is the only functionality required of an Exchange server.

However, that service is not sufficient to use as an Exchange Server 2007 Hub Transport.

In general terms, a front-end server is something that a client/end user may interact with directly. A front-end server may be exposed directly to the Internet or published to the Internet via a product such as Internet Security and Acceleration (ISA) Server.

In general terms, a backend server is something that hosts a storage group. A storage group may contain zero or more mailbox stores and zero or more public folder stores (this is somewhat dependent upon the edition of Exchange installed, either Standard or Enterprise). If a backend server hosts one or more mailbox stores, then it is a mailbox server. If a backend server hosts one or more public folder stores, then it is a public folder server. It is perfectly okay for a backend server to be both a mailbox server and a public folder server.

For Exchange Server 2003, in some cases the front-end server was basically a proxy to a backend server. In the case of legacy protocols (POP3 and IMAP), the front-end server would connect to the backend server using the same protocols and simply forward the replies and responses that originated from the backend server. In other cases, such as for SMTP, Web-based Distributed Authoring and Versioning (WebDAV), and OWA, the front-end server would process the command(s) and handle them as appropriate. In the case of RPC/HTTP (now referred to as Outlook Anywhere), the front-end server was also a proxy for a MAPI connection to a backend server.

Legacy Protocol Proxy Security

It is common to protect a front-end server with a Secure Sockets Layer (SSL) certificate. This allows (and may optionally require) data passing between the client and the front-end server to be encrypted. However, in Exchange Server 2003, communication between a front-end server and a back-end server is not protected in this way. Data is always passed in the clear.

So how does this affect what needs to be monitored depending upon the server role?

- All servers need to be monitored for "alive" (ping test).
- All servers need to be monitored for disk space, processor usage, memory usage, and so on.
- All servers need to be monitored for legacy protocol availability and usability, if those protocols are enabled.
- Front-end servers need to be monitored for WebDAV and HTTP (or HTTPS) availability and usability.

- Front-end servers need to be monitored for the Windows services that support WebDAV and HTTP (or HTTPS).
- Back-end servers need to be monitored for MAPI availability and usability.
- Back-end servers need to be monitored for general performance of the I/O subsystem (disk queues, latency, etc.).

That seems like quite a bit. But is it all? No. There are other things that affect Exchange performance:

- Are configured Active Directory (AD) servers available?
- Are configured Global Catalog (GC) servers available?
- Is Name Service Provider Interface (NSPI) working on the GC servers?
- How long does a query take to the GC?
- How long does a query take to AD?
- How long does a NSPI query take the GC?

This doesn't even include the obvious items like these:

- Is there a working route to the Exchange server?
- Is there a working route to the Internet?
- Are all required switches operational?
- Are all required routers operational?
- Is Internet access available?
- How congested is the network?

Now that you understand the basics of Exchange Server 2003 monitoring, it's time to review similar concepts for Exchange Server 2007.

Exchange 2007

As described earlier, as of Exchange Server 2007, Microsoft allows a system administrator to make the distinction, during the installation process, as to the particular purpose for which an Exchange server will be used. Along with these distinctions, Microsoft added feature content to Exchange Server 2007 and moved around various sets of features so that functionality better mapped to the roles they chose.

The Microsoft roles are as follows:

Edge Transport The Edge server role maps most closely to a SMTP relay server. However, the Edge server is designed to be placed into a perimeter network and to synchronize to the internal network over well-known ports. The Edge server is not a member of the internal Active Directory domain. The primary purpose of the Edge server is for message hygiene. This means it will process all incoming messages for spam, viruses, and so on prior to passing a message to the internal network. Many companies use a third-party provider for these

services. Microsoft was somewhat late to the game for providing message hygiene services. The Edge server uses a process known as EdgeSync to allow it to be aware of internal email addresses and user objects. The Edge server role is completely new for Exchange Server 2007.

Client Access Server (CAS) The CAS role handles OWA, other legacy protocols, Outlook Anywhere, and WebDAV. This also includes Exchange Active Sync (EAS). Significantly, the CAS does not handle SMTP or MAPI.

Hub Transport (HT) The HT role handles SMTP and nothing but SMTP. This brings with it specific issues to be aware of when you begin planning your Exchange Server 2007 deployment. If you want your external users, who are using IMAP or POP3, to be able to send email, they also will require access to a HT. This means either you must expose an additional server to the Internet or you should collocate HT and CAS on the same server (which may otherwise complicate your internal configuration if there is a desire to separate internal SMTP from external SMTP).

Mailbox (MB) The MB role handles MAPI and nothing but MAPI. While a mailbox server may host both mailbox stores as well as public folder stores, the de-emphasis of public folders within Exchange explains the naming selected by Microsoft. Functionality of public folders with Exchange Server 2007 (at service pack 1 or higher) is equivalent to that of Exchange Server 2003 service pack 2. As of this writing, Microsoft has committed to supporting public folders in E14 (the Exchange Server release following Exchange Server 2007) at the current level but has made no comments as to support beyond that release.

Unified Messaging (UM) The UM role moves the concept of voicemail into Exchange Server 2007. UM, when integrated with a supported PBX or other interface, provides call-control services and message storage. While this book will not cover UM in detail (UM is an entire book unto itself), I will present the key performance factors to examine for UM. The other roles with Exchange Server 2007 are fairly easily understood. Like Edge, the UM role is completely new in Exchange Server 2007.

If your environment requires only MAPI, you are still required to have both MB and HT servers. All messages in Exchange Server 2007 will go through an HT prior to delivery, even if the recipient is in the same mailbox store as the originator. This is to ensure that transport rules can be processed for compliance and journaling.

So how does this affect what needs to be monitored depending upon the server role? It really isn't so different from Exchange Server 2003:

- All servers need to be monitored for "alive" (ping test).
- All servers need to be monitored for disk space, processor usage, memory usage, etc.
- All CAS servers need to be monitored for legacy protocol availability and usability, if those protocols are enabled.
- All CAS servers need to be monitored for WebDAV and HTTP (or HTTPS) availability and usability.
- All CAS servers need to be monitored for the Windows services that support WebDAV and HTTP (or HTTPS).
- All Mailbox servers need to be monitored for MAPI availability and usability.

- All Mailbox servers need to be monitored for general performance of the I/O subsystem (disk queues, latency, etc.).
- All HT servers need to be monitored for SMTP protocol availability and usability.

That seems like quite a bit. But is it all? No. There are other things that affect Exchange Server 2007 performance:

- Are configured Active Directory (AD) servers available?
- Are configured Global Catalog (GC) servers available?
- Is NSPI working on the GC servers?
- How long does a query take to the GC?
- How long does a query take to AD?
- How long does a NSPI query take the GC?

This doesn't even include the obvious items like the following:

- Is there a working route to the Exchange server?
- Is there a working route to the Internet?
- Are all required switches operational?
- Are all required routers operational?
- Is Internet access available?
- How congested is the network?

Given an understanding of what has to be monitored, it is now time to review the information available (outside of performance information) that documents the activities of Exchange servers.

Extracting Information from the Environment (Logging)

Historically speaking, logging for applications in the Windows Server world is pretty hit or miss. The official place for all application logging is the event log, specifically the application event log. Unfortunately, Microsoft decided long ago that the format of the event logs would be binary, not text. This makes processing the event logs somewhat unwieldy and difficult as an Event Log Viewer application is required to examine the event logs. The Event Log Viewer provided by Microsoft has only basic functionality. This has, of course, been improved upon by many third parties.

Over a number of years of development, Exchange Server has gained a number of different logging capabilities. Each of them is briefly discussed in the following sections.

Diagnostic Logging

Diagnostic logging is used when there is reason to believe that either Exchange Server has a bug or unexpected and unexplainable results are being obtained. Enabling diagnostic logging

causes additional information about a particular module of Exchange Server to be entered into the event log.

NOTE A typical use of diagnostic logging is to have Exchange Server log the details of its search for all of the domain controllers in its Active Directory site.

Diagnostic logging is normally configured in the Exchange Management Console (Exchange System Manager for Exchange Server 2003 servers). Each module may have up to five different values; however, the GUI exposes only four of them: None (the default), Low, Medium, and High. There is also a fifth value, variously known as Internal or Expert, which won't be further discussed here.

Diagnostic logging is real-time logging. As events are occurring, the application event log is continuously updated with the output from any enabled diagnostic logging.

WARNING Increasing the level of diagnostic logging can cause large numbers of messages to be posted into your event log. It can also lead to significantly increased disk I/O and processor utilization. Use diagnostic logging to identify issues, but then turn it off as soon as you can.

Protocol Logging

In the section "Protocol-Specific Tests" earlier in this chapter, I showed that every protocol has communication that goes back and forth between the source and destination servers. There are times when it is necessary or desirable to examine the contents of the communication. This is done by enabling protocol logging.

TIP You should enable your protocol logs. All protocol logs are disabled by default. If you do not have the protocol logs, you may not be able to figure out a problem after it has occurred.

In Exchange Server 2003, protocol logging was an attribute of the protocol virtual server (SMTP, IMAP, and POP3) and was enabled or disabled for each virtual server. With Exchange Server 2003, you have the capability of identifying each and every field that would be included in the log, how often the log will switch (hourly, daily, when it reaches a certain size, etc.), and the location (directory) of the log files. The system administrator is responsible for regularly removing log files so that they do not grow to consume all available disk space.

In Exchange Server 2007, protocol logging is an attribute of receive connectors, send connectors, and the intra-organizational Hub Transport server. All receive connectors will share a common log file, and all send connectors will share a common log file (this is on a per-server basis, of course). As with Exchange Server 2003, you may identify the location (directory) of the log files. However, with Exchange Server 2007, you limit log files by size and by age only. A great new feature is that, by default, logs are retained for only 30 days and then automatically removed. The system administrator does have the option of changing the 30-day default.

TIP To enable protocol logging for SMTP communications, you'll have to use Exchange Management Shell (EMS). Refer to `http://technet.microsoft.com/en-us/library/bb124531.aspx`. To enable protocol logging for POP3 and IMAP, an XML file must be edited. Refer to `http://technet.microsoft.com/en-us/library/aa997690(EXCHG.80).aspx`.

Event Logging

A great deal of information about the event log has already been presented. It is worthwhile noting here that Exchange Server logs a number of processes to the Event Log on a normal basis, purely as informational items. These tend to be processes such as service startup and shutdown, the completion of daily online maintenance for each mailbox store, and any errors that may arise. All errors should be tracked and evaluated, of course. Some of the informational messages, such as the amount of white space available in a mailbox store, may be good information as well. Using this type of information for application health is something that the management packs for OpsMgr do quite well. This topic will be covered in depth throughout the book.

TIP Learn to use `www.EventID.Net`; even break out your wallet and subscribe to it. While Microsoft has made great strides in its documentation, your best place to learn about event ID errors and how to resolve them is by using `www.EventID.Net`.

Message Tracking

Message tracking logs provide a summary of the transfer of all messages into or out of any Exchange Server with the Mailbox, Hub Transport, or Edge Transport roles. Message tracking is enabled by default on all Exchange Servers with those roles installed.

The concept of message tracking has remained the same between Exchange Server 2003 and Exchange Server 2007. However, the message tracking log file and its contents have changed dramatically. Message tracking provides after-the-fact analysis to determine where, how, and when a message came into a particular server and was either delivered, forwarded, or dropped. From the perspective of this book, message tracking allows the generation of analysis reports describing such events as top 10 message originators, top 10 message destinations, and total messages by day. While message tracking can be used to analyze whether particular messages have arrived or been delivered from a server, generally the protocol logs are a better and more detailed source for that type of information.

As with the protocol logs, the system administrator can control the size of the message tracking logs, how long they are retained on the server disk, how large the folder containing the logs is allowed to grow, and the location of the message tracking log folder.

Performance Monitor

Performance Monitor is a real workhorse. Actually, it isn't the Performance Monitor application itself but the hooks into the system and into applications via performance objects and performance counters that provide the power expressed through the Performance Monitor application.

As previously discussed, the Performance Monitor application can provide a real-time view into large numbers of application and operating system parameters. It can also store this information for later analysis.

However, applications such as OpsMgr can also interrogate performance counters, using the same application programming interfaces (APIs) that were put into place for Performance Monitor. This allows OpsMgr, for example, to look at the total processor usage on a system and keep a running average, to look at the total memory utilization on a system and keep a running average, or to examine the amount of paging going on and notify a system administrator that action needs to be taken.

Any system administrator can use the Performance Monitor tool to identify objects of interest and use other tools like PowerShell to trap the contents of that object and to alert the system administrator when the objects exceed certain boundaries. However, OpsMgr provides a much more powerful system in which dozens and hundreds of tests can be grouped together and modified by the system administrator where appropriate. You will learn about the management packs in detail in future chapters.

Helping Yourself

Even with all of the types of logging I've discussed, you may not be able to find a piece of information that you want or need when your in-brain memory fails you. So generate ad hoc logs. Anytime you run a script or a program—anything that modifies the state of the system—you should generate some permanent logging type of output. Whether the logging is to the event log, a flat file, or somewhere else dictated by your corporate rules and policies, you should generate that log. Being able to prove when and where a particular process was executed can be a career-saving event.

In my 27 years of experience in information technologies, the number one cause of downtime, in shops of all sizes, is poor change management. People do not track what changes they make and then cannot go back and reverse those changes when the modifications go wrong. Don't let that happen to you. Log every manual change you make, just as I suggested having a document describing every automated change. Make this a policy for your organization.

Experience is your best teacher for methods and processes to follow to solve problems. When you see an issue, the best way to resolve it is to dive in and start to figure it out. If you aren't certain that you know the answer, don't guess. Ask for help. There are many Microsoft-oriented communities that are happy to provide help.

TIP Google has a special search engine just for Microsoft searches, `http://google.com/microsoft`, which searches only Microsoft web properties for hits. This is a good way to hunt for solutions to Microsoft issues.

Administrative Models

In general, companies that have small implementations of Exchange Server want to use GUI administration tools because they have little churn, or number of changes that happen on a regular basis. Companies with large implementations want to be able to utilize Exchange their way, without any preconceived ideas getting in the way—especially Microsoft's ideas. And, of course, there are those many companies in between; they usually want to use the GUI, but occasionally they need to make bulk changes that necessitate a scripting interface.

This has led to the development of two administrative models: one based on the command-line interface and one based on the GUI. Throughout the history of Exchange Server, until Exchange Server 2007, the GUI was always more capable than the command line. As of Exchange Server 2007, all of the capabilities of the GUI are actually implemented by the command line, which is based on a set of PowerShell cmdlets.

In the next two sections, I will talk about the prior technologies used to support Exchange administration.

Command-Line Interface (CLI)

As Exchange Server has grown and matured as a product, the various tools available for modifying Exchange and its objects have grown and matured as well:

CDO—Collaboration Data Objects In the beginning, there was CDO … CDO was the first of the exposed messaging interfaces and it was much simpler to use than MAPI (although it was based on MAPI). And the capabilities that CDO exposed were pretty limited. It was primarily good for sending email via an Exchange server. CDO also includes various incarnations known as CDONTS and CDOSYS, which are included with Windows NT 4.0 Server and above, that support sending SMTP email without using Exchange Server.

CDOEX—CDO for Exchange CDO for Exchange is an enhanced version of CDO that is available only on an Exchange server. CDOEX includes support for other Exchange objects, such as folders, appointments, calendars, tasks, contacts, mailboxes, and so on. Using CDOEX, you can build complete messaging applications. CDOEX is not supported as a part of Exchange Server 2007.

CDOWF—CDO for Workflow Initially, Exchange Server was positioned by Microsoft as a Lotus Notes killer, and the work flow capabilities of Exchange were strongly pushed. CDOWF provided the work flow support in Exchange, supporting event-driven work flows (e.g., if A happens then do B). For various reasons, CDOWF never caught on. As of Exchange Server 2003, CDOWF was turned into a legacy feature (no new development, but still supported). The capabilities present in CDOWF have been absorbed and significantly expanded upon by Windows SharePoint Services. CDOWF is not supported as a part of Exchange Server 2007.

CDOEXM—CDO for Exchange Management CDOEXM provides the capabilities for managing an Exchange system—dealing with servers, storage groups, message stores, public folder trees, and so on. Each of these objects can be manipulated in various ways, such as creating, deleting, mounting, and moving. CDOEXM is limited in the objects it handles (it does not allow you to modify or create address lists or recipient policies, for example), and a program or script using CDOEXM can be run only on a server on which Exchange Server is installed or a computer on which the Exchange System Management Tools are installed. Some information (particularly some message-store-related information) can be accessed only via CDOEXM and not via other technologies. CDOEXM is not supported as a part of Exchange Server 2007.

ADSI—Active Directory Service Interfaces Not a uniquely Exchange scripting technology, ADSI provides interfaces into Active Directory. Most of the capabilities of CDOEX and CDOEXM can be replaced with ADSI scripts. Doing so has the advantage of allowing the scripts to run on computers that do not have either Exchange Server or the Exchange Server System Management Tools installed on them. Many of the scripts shown in this book will use ADSI. The value of ADSI within an Exchange Server 2007 environment is even higher than it was as a part of an Exchange Server 2003 environment. However, many of the cmdlets that are a part of PowerShell can simplify the access to Active Directory as opposed to using ADSI.

ADO—ActiveX Data Objects These components provide capabilities for accessing data from multiple places through a common interface. Both Active Directory and Exchange Server support ADO.

WMI—Windows Management Instrumentation WMI is the Microsoft implementation of Web-Based Enterprise Management (WBEM). WMI is simply a way for information to be presented, interrogated, and modified in an industry-standard way. Both Exchange Server and Active Directory expose a large amount of their data via WMI. With Exchange Server 2003, WMI provides more access to more information than any other interface. The Exchange_* specific interfaces that were present in WMI for Exchange 2000 Server and Exchange Server 2003 are no longer available in Exchange Server 2007. They have been enhanced and completely replaced by PowerShell cmdlets.

PS—PowerShell PowerShell contains all the capabilities of the preceding technologies and more. The Exchange Management Console in Exchange Server 2007 is written to use PowerShell cmdlets, and its capabilities are based on them. All of the capabilities of the EMC are available to third-party developers and administrative script writers.

Graphical User Interface (GUI)

With Exchange Server 2007, the GUI to Exchange Server, the Exchange Management Console, is based completely upon cmdlets available within PowerShell. The mechanism as to how PowerShell cmdlets are called from the EMC is beyond the scope of this discussion. However, suffice it to say that the EMC provides only a subset of the capabilities available from the PowerShell command line—a complete reversal from the situation in earlier versions of Exchange Server.

In Exchange Server 2003, all the way back to Exchange 4.0, Microsoft utilized "hidden" application programming interfaces (APIs) in order to accomplish much of the functionality available within the Exchange System Manager. This has always made it difficult to automate a number of significant capabilities that are available within the management console but are not otherwise available in the published APIs. As described in the preceding section on the command-line interface, Microsoft has attempted to address this shortfall in a number of ways with a number of technologies, but it has always fallen short, until the release of Exchange Server 2007 with PowerShell.

Today, the EMC provides a fairly comprehensive experience for smaller companies, at least equivalent to those capabilities available in the Exchange Server 2003 ESM. Also, the cmdlets available in PowerShell provide all of the capabilities in the EMC plus many other functions a system administrator may wish to perform.

Where does that leave the system administrator of today?

Mixed Interfaces (GUI and CLI)

With the capabilities present in both EMC and PowerShell, these mixed interfaces provide the system administrator of Exchange Server 2007 systems with the capability of using the EMC for one-off changes, but for *every* change that may be performed, the system administrator may choose to script it to perform these functions in a more efficient and time-effective manner.

I am fond of saying that one-off changes are for GUIs, but every other change that may conceivably be repeated should be scripted and documented and stored for further use.

TIP Scripting a solution to be used in the future is pointless if you do not document the solution. Whether your solution is 3 lines long or 300 lines long, without documentation you will not be aware, in the future, of the problem that you were attempting to solve. Without documentation, using a saved solution will be rife with conceivable problems.

Summary

In this chapter, I have presented significant background regarding Exchange Server and OpsMgr 2007. You have learned about the following topics:

- The history of the products involved
- Manual mechanisms for performing some of the tests that may be a part of determining the health of a given server and/or application
- Various sets of information available from Windows operating systems that allow a system administrator to peer into the depths of a server's or application's health
- Roles and responsibilities of various installations of Exchange Server
- Types of logging available on Windows to allow the system administrator to obtain additional information about server health and application health
- Administrative interfaces available in Exchange Server 2007 and in prior versions of Exchange Server

In Chapter 2, I will discuss, in detail, the interesting items and objects associated with properly monitoring Exchange Server.