Chapter 18 The Redirector

Microsoft Windows offers applications the capability to communicate over a network using built-in file system services. This is sometimes referred to as the network operating system (NOS) capability. This chapter explores these networking capabilities using Windows file system components available in Windows 95, Windows 98, Windows Me, Windows NT, and Windows CE. The purpose of this chapter is to provide an understanding of these capabilities as they relate to the mailslot and named pipe networking technologies, which are covered in greater detail in Chapters 19 and 20, respectively.

When applications want to access files on a local system, they rely on the operating system to service I/O requests. This is typically referred to as local I/O. For example, when an application opens or closes a file, the operating system determines how to access a device that contains the contents of the specified file. Once the device is found, the I/O request is forwarded to a local device driver. The same operating principle is also available for accessing devices over a network. However, the I/O request must be forwarded over a network to the remote device. This is referred to as I/O redirection. For example, Windows allows you to map or redirect a local disk identifier—such as E:—to a directory share point on a remote computer. When applications reference E:, the operating system redirects the I/O to a device called a redirector. The redirector forms a communication channel to a remote computer to access the desired remote directory. This functionality allows applications to use common file system API functions, such as ReadFile and WriteFile, to access remote files across a network.

This chapter discusses how the redirector is used to redirect I/O requests to remote devices. This is important information—it is the foundation for communication in the mailslot and named pipe technologies. First we cover how files can be referenced over a network with the UNC using the MUP resource locator. This is followed by an explanation of how MUP calls a network provider, which exposes a redirector to form communications among computers using the SMB protocol. Finally, we describe network security considerations when accessing files over a network using basic file I/O operations.

18.1 Universal Naming Convention

UNC paths provide a standardized way of accessing files and devices over a network without specifying or referencing a local drive letter that has been mapped to a remote file system. This is important because it allows applications to become drive-letter–independent and work seamlessly in a network environment. UNC names are better than names that reference a local drive letter because you don't have to worry about running out of drive letters when forming connections to access server shares. Drive letters also operate on a per-user basis—processes that are not running in your user context cannot access your drive mappings.

UNC names are specified as follows:

\\[server]\[share]\[path]

The first portion, \\[server], starts with two backslashes followed by a server name. The server name represents a remote server in which an application wants to reference a remote file. The second portion, \[share], represents a share point on the remote server. A share point is simply a directory in a file system that is identified on a network as shared for network user access. The third portion, \[path], represents a directory path to a file in a file system. For example, suppose you have a server named Myserver that contains a directory on a local drive named D:\Myfiles\CoolMusic that is shared out as Myshare. Let's also assume the shared directory contains a file named SAMPLE.MP3. If you would like to reference SAMPLE.MP3 from a remote machine, simply specify the UNC name \\Myserver\Myshare\Sample.mp3. As you can see, it's much easier to reference a file across a network than it is to map a local drive to the shared directory Myshare.

Referencing files over a network using UNC names hides the details of forming a connection over a network from an application. This is great—a system can easily locate network server directory shares and file paths with UNC names, even over a modem connection. All of the network communication details are handled by a network provider's redirector, which we discuss later in this chapter. As discussed in Chapters 19 and 20, the mailslot and named pipe technologies depend solely on the use of UNC names for identification.

Figure 18-1 illustrates the common components that form UNC connections on the NOS in Windows. The figure also shows how the data flows among client and server NOS components. Using the UNC path \\Myserver\ Myshare\Sample.mp3 described earlier, the remainder of this chapter describes each component and demonstrates what happens when we open this file across a network.

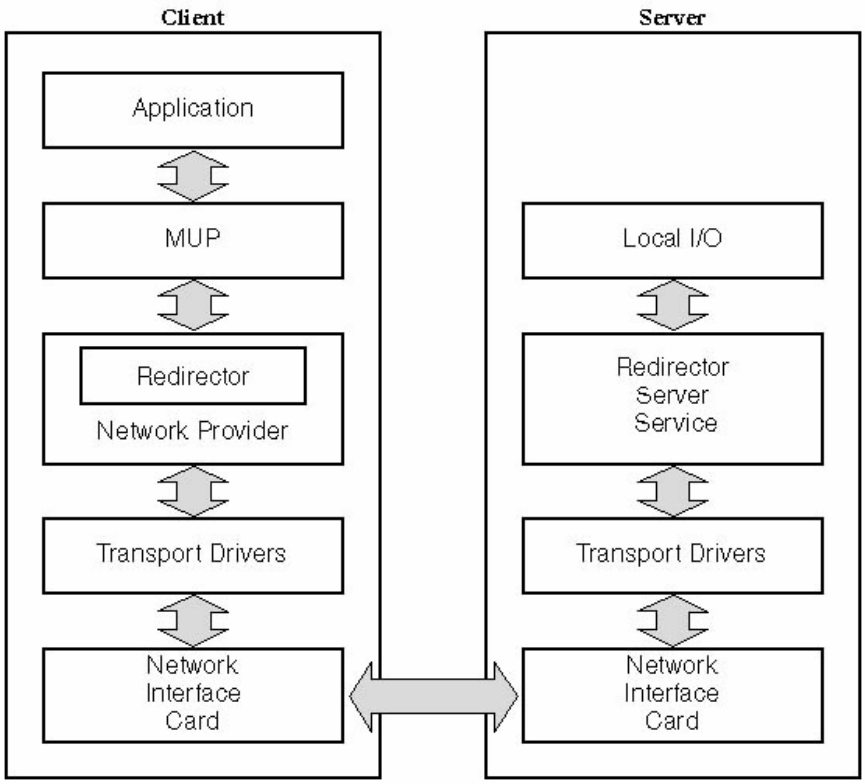

Figure 18-1 *Redirector components*

18.2 Multiple UNC Provider

MUP is a resource locator responsible for selecting a network provider to service UNC connections. A network provider is a service that can use network hardware to access resources—such as files and printers—located on a remote computer. MUP uses a network provider to form communications on all UNC-name–based I/O requests for files and printers. We discuss the details of a network provider later in this chapter.

Windows 95, Windows 98, and Windows NT platforms are all capable of having multiple network providers installed. For example, Windows platforms provide a network provider named Client for Microsoft Networks. It is also possible to install other third-party network providers, such as Novell's Novell Client version 3.01 for Windows. Thus, more than one network provider might be able to service a single UNC request at a time. On the other hand, Windows CE can have only one network provider, Client for Microsoft Networks.

The primary role of MUP is to decide which network provider should service a UNC request. MUP makes this decision by sending the UNC names in the request to each installed provider (in parallel). If a network provider indicates that it can service a request involving the UNC names, MUP sends the rest of the request to the provider. If more than one provider is capable of servicing a UNC request, MUP chooses the network provider with the most priority. Network provider priority is determined by the order in which providers are installed on your system. In Windows 95, Windows 98, and Windows NT, the priority can be managed by modifying the registry key ProviderOrder in the following directory in the Windows registry:

```
\HKEY_LOCAL_MACHINE
    \SYSTEM
        \CurrentControlSet
            \Control
                \NetworkProvider
                    \Order
```

The installed providers are listed first to last in order of priority. Because Windows CE can have only one provider, it does not use MUP to resolve UNC names. Instead, the UNC requests go directly to its single provider.

18.3 Network Providers

As mentioned earlier, a network provider is a service that uses network hardware to access files and printers located on a remote computer. This is considered to be the core function of a NOS. One of a provider's main capabilities is redirecting a local disk identifier—such as E:—to a disk directory located on a remote computer. Providers must also be able to service UNC connection requests. In Windows, network providers do this by exposing a redirector to the operating system.

Windows features a network provider named Client for Microsoft Networks, formally known as the Microsoft Networking Provider (MSNP). The MSNP enables communications among Windows 95, Windows 98, Windows NT, and Windows CE platforms. Windows CE, however, does not support multiple network providers and provides only built-in client-side support for the MSNP.

18.4 Redirector

A redirector is a component exposed by a network provider to an operating system that accepts and processes remote I/O service requests by formulating service request messages and sending them to a remote computer's redirector server service. The remote computer's redirector server service receives the request and services it by making local I/O requests. Because a redirector provides I/O services to higher level

services such as MUP, a redirector hides the details of the network layer from applications so that applications don't have to supply protocol-specific parameters to a redirector. Thus, a network provider is protocol-independent: applications can operate in almost any network configuration.

The MSNP provides a redirector that works directly with the networking transport layer and NetBIOS to form communication between a client and a server. The NetBIOS API discussed in Chapter 17 provides a programming interface with these same transports. This redirector provided by MSNP is often referred to as the LAN manager redirector because it is designed around the old Microsoft LAN manager software that provided NOS capability to MS-DOS applications in the past. (For more detailed information about the NetBIOS programming interface, see Chapter 17.) The NetBIOS interface is capable of communicating over numerous network protocols. This makes the MSNP redirector protocol-independent: your application does not have to concern itself with the specific details of a network protocol. When your application uses the MSNP redirector, it can communicate over TCP/IP, NetBEUI, or even IPX/SPX. This is a helpful feature because it allows applications to communicate no matter what the physical network comprises. However, one important detail needs to be considered. For two applications to communicate with each other over the network, the two workstations must have at least one transport protocol in common. For example, if Workstation A has only TCP/IP installed and Workstation B has only IPX installed, the MSNP redirector will not be able to establish communication between the two workstations over a network.

The MSNP redirector communicates with other workstations by sending messages to a remote workstation's redirector server service. These messages are set up in a well-defined structure known as SMB. The actual protocol for how the redirector sends and receives messages to a remote workstation is known as the Server Message Block File Sharing Protocol, or simply the SMB protocol.

18.5 Server Message Block

The SMB protocol was originally developed by Microsoft and Intel in the late 1980s to allow remote file systems to be transparently accessed by MS-DOS applications. Today this protocol simply allows a Windows MSNP redirector to communicate with a remote workstation's MSNP server service using an SMB data structure. An SMB data structure contains three basic components: a command code, command-specific parameters, and user data.

The SMB protocol is centered on a simple client request and server response messaging model. An MSNP redirector client creates an SMB structure with a specific request indicated in the command code field. If the command requires sending data, such as an SMB Write instruction, data accompanies the request. The SMB structure is then sent over a transport protocol such as TCP/IP to a remote

workstation's server service. This server service processes the client's request and transmits an SMB response data structure back to the client.

Now that we've covered the basics of the components used in forming communication through the MSNP redirector, let's follow how each component communicates when we try to open \\Myserver\Myshare\Sample.mp3 across a network. It does so by following these steps:

1. An application submits a request to the local operating system to open \\Myserver\Myshare\Sample.mp3 using the CreateFile API function.
2. The local operating system's file system determines that the I/O request is destined for a remote machine named \\Myserver based on the UNC path description, so it passes the request to MUP.
3. MUP determines that this I/O request is destined for the MSNP provider because the MSNP provider finds \\Myserver on the network using NetBIOS name resolution.
4. The I/O request is passed to the MSNP provider's redirector.
5. The redirector formats the I/O request as an SMB message to open the file SAMPLE.MP3 that is contained in the remote \Myshare directory.
6. The formatted SMB message is transmitted over a network transport protocol.
7. The server named \\Myserver receives the SMB request from the network and passes the request to the server's MSNP redirector server service.
8. The server's redirector server service submits a local I/O request to open the SAMPLE.MP3 file that is located on the \Myshare share point.
9. The server's redirector server service formats an SMB response message regarding the success or failure of the local file open I/O request.
10. The server's SMB response message is sent back to the client over a network transport protocol.
11. The MSNP redirector receives the server's SMB response and passes a return code back to the local operating system.
12. The local operating system returns the return code to the application CreateFile API request.

As you can see, the MSNP redirector must go through quite a few steps to grant applications access to remote resources. The MSNP redirector also provides access control to resources on a network as a form of network security.

18.6 Security

Our discussion of security focuses on accessing resources over a network. However, before we can discuss how security is enforced on resources over a network, we need to discuss security basics on a local machine. Windows NT platforms provide the capability to locally and remotely control access to system resources such as files and directories. These resources are considered securable objects. When an application

attempts to access a securable object, the operating system checks whether an application has access rights to that object. The three basic access types are read, write, and execute privileges. Windows NT systems accomplish access control through security descriptors and access tokens.

18.6.1 Security Descriptors

All securable objects contain a security descriptor that defines their access control information. A security descriptor consists of a SECURITY_DESCRIPTOR structure and its associated security information, which includes the following items:

- Owner Security Identifier (SID). Represents the owner of the object.
- Group SID. Represents the primary group owner of the object.
- Discretionary Access Control List (DACL). Specifies who has what type of access to the object. Access types include read, write, and execute privileges.
- System Access Control List (SACL). Specifies the types of access attempts that generate audit records for the object.

Applications cannot directly manipulate the contents of a security descriptor structure. A descriptor can, however, be manipulated indirectly through Windows security APIs that provide functions for setting and retrieving the security information. We demonstrate this at the end of this chapter.

18.6.2 Access Control Lists and Access Control Entities

The DACL and SACL fields of a security descriptor are access control lists (ACLs) that contain zero or more access control entities (ACEs). Each ACE controls or monitors access to an object by a specified user or group. An ACE contains the following types of access control information:

- A SID that identifies the user or the group that the ACE applies to
- A mask that specifies access rights such as read, write, and execute privileges
- A flag that indicates ACE type (allow-access, deny-access, or system-audit)

Note that system audit ACE types are used only in SACLs, whereas allow-access and deny-access ACE types are used in DACLs. Figure 18-2 shows a file object with an associated DACL.

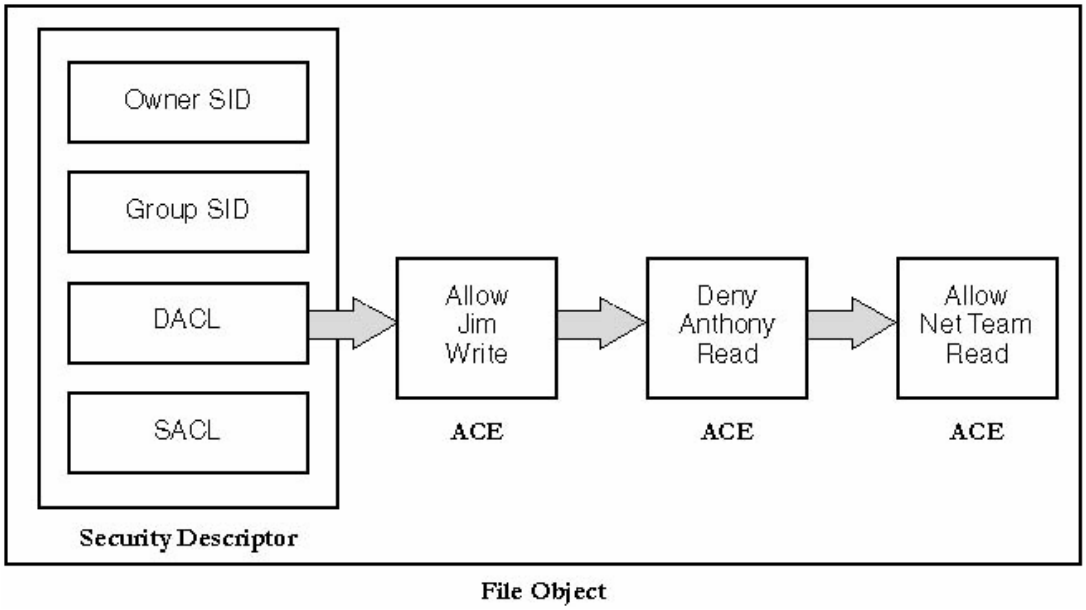

Figure 18-2 *File object with an associated DACL*

If a secured object does not have a DACL (its DACL has been set to a null value using the SetSecurityDescriptorDacl API function), the system allows everyone full access. If an object has a DACL, the system allows only the access that is explicitly allowed by the ACEs in the DACL. If there are no ACEs in the DACL, the system does not allow access to anyone. Similarly, if a DACL has some allow-access ACEs, the system implicitly denies access to all users and groups not included in the ACEs.

In most cases, you need to specify only allow-access ACEs, with the following exception: if you include an allow-access ACE for a group, you might have to use deny-access ACEs to exclude members of that particular group. To do this, you must place a user's deny-access ACE ahead of a group's allow-access ACE. Note that the order of the ACL is important because the system reads the ACEs in sequence until access is granted or denied. The user's access-denied ACE must appear first; otherwise, when the system reads the group's access-allowed ACE, it will grant access to the restricted user.

Figure 18-2 shows how to set up a DACL that grants read access to a group named Net Team. Let's assume that the Net Team group consists of Anthony, Jim, and Gary, and we want to grant read access to everyone in the group except Anthony. To do this, we must have a deny-access ACE for Anthony set before the allow-access ACE for the Net Team. Figure 18-2 also includes an allow-access ACE to grant Jim write access. Remember that applications do not directly manipulate ACLs; instead, they use security APIs to perform these transactions.

18.6.3 Security Identifiers

We have noted that security descriptors and ACEs for securable objects include a SID, which is a unique value used to identify a user account, group account, or logon session. A security authority, such as a Windows NT server domain, maintains SID information in a security account database. When a user logs on, the system retrieves the user's SID from the database and places it in a user's access token. The system then uses the SID in the user's access token to identify the user in all subsequent interactions with Windows NT security.

18.6.4 Access Tokens

When a user logs on to a Windows NT system, the system authenticates the user's account name and password, which are known together as login credentials. If a user logs on successfully, the system creates an access token and assigns it the user's SID. Every process executed on behalf of this user will have a copy of this access token. When a process attempts to access a secured object, the SID in the access token is compared with access rights assigned to SIDs in DACLs.

18.7 Network Security

Now that we've briefly explained how security is enforced on a local machine, we are ready to look at security when accessing secured objects over a network. As we saw earlier, the MSNP redirector is responsible for accessing resources among computers. The MSNP redirector is also responsible for establishing a secure link between a client and a server by creating user session credentials.

Session Credentials

There are two types of user credentials: primary login and session credentials. When a user sitting in front of a workstation logs on to the machine, the user name and the password presented by the user become the primary set of credentials and are stored in an access token. Only one set of primary credentials exists at any given time. When a user attempts to establish a connection (either mapping a drive or connecting through UNC names) to a remote resource, the user's primary credentials are used to validate access to the remote resource. Note that with Windows NT systems, the user has the option of supplying a different set of credentials to be used in validating with the remote resource. If the user's credentials are valid, the MSNP redirector establishes a session between the user's computer and the remote resource. The redirector associates the session with session credentials, which consist of a copy of the credentials the user's computer used to validate the connection with the remote

resource. Only one set of session credentials can be established at a time between a user's computer and a remote server. If Machine B has two share points, \Hack and \Slash, and if the user of Machine A maps \Hack to G and \Slash to H, both sessions share the same session credentials because they both refer to the same remote server.

The MSNP redirector server service handles security access control on a remote server. When the MSNP redirector server attempts to access a secured object, it uses the session credentials to create a remote access token. From there, security is managed as if the access were made locally. Figure 18-3 demonstrates how the MSNP redirector establishes security credentials using Windows NT domain security.

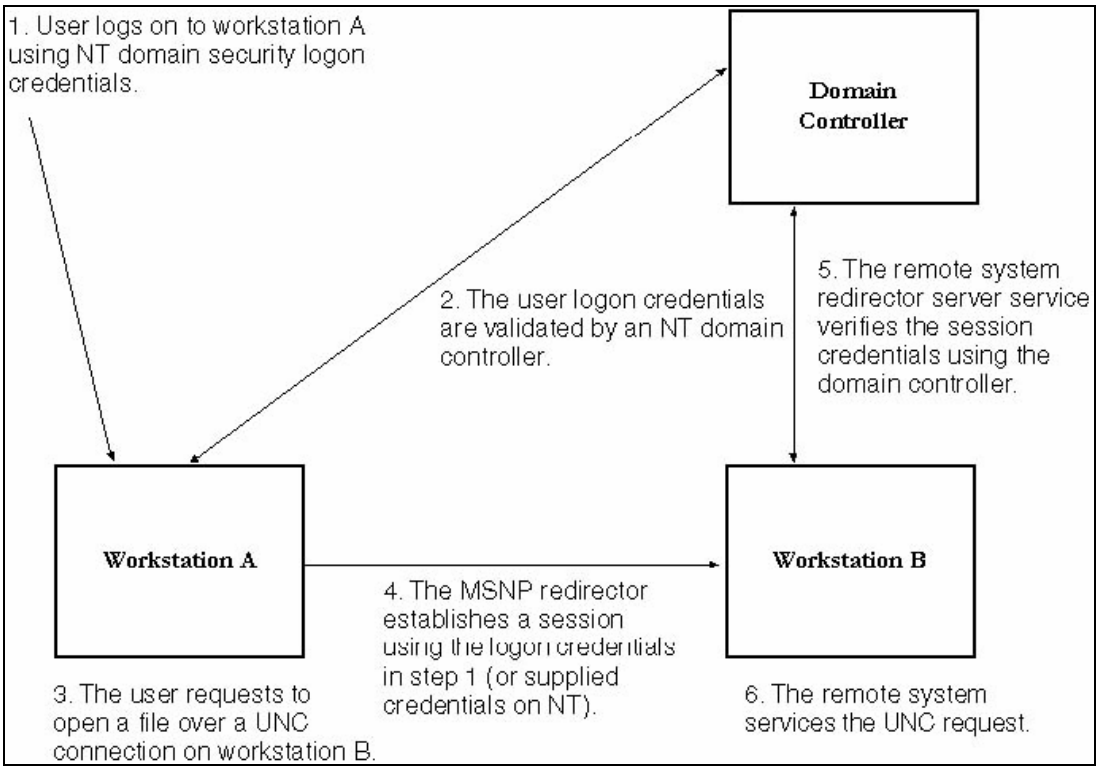

Figure 18-3. *Security credentials demonstration*

18.8 A Practical Example

Windows applications can use the CreateFile, ReadFile, and WriteFile API functions to create, access, and modify files over a network using the MSNP redirector. Windows NT systems are the only platforms that support Windows security. The following sample demonstrates how to write a simple application that will create a file over a UNC connection. You will find a file with this code called FILEIO.CPP on the companion CD.

```
#include <windows.h>
#include <stdio.h>
```

```
void main(void)
{
    HANDLE FileHandle;
    DWORD BytesWritten;

    // Open a handle to file \\Myserver\Myshare\Sample.txt
    if ((FileHandle = CreateFile("\\\\Myserver\\Myshare\\Sample.txt",
        GENERIC_WRITE | GENERIC_READ,
        FILE_SHARE_READ | FILE_SHARE_WRITE, NULL,
        CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL))
        == INVALID_HANDLE_VALUE)
    {
        printf("CreateFile failed with error %d\n", GetLastError());
        return;
    }

    // Write 14 bytes to our new file
    if (WriteFile(FileHandle, "This is a test", 14,
        &BytesWritten, NULL) == 0)
    {
        printf("WriteFile failed with error %d\n", GetLastError());
        return;
    }

    if (CloseHandle(FileHandle) == 0)
    {
        printf("CloseHandle failed with error %d\n", GetLastError());
        return;
    }
}
```

18.9 Conclusion

This chapter introduced you to the Windows redirector, which enables an application to access Windows file system resources over a network. We explained the fundamental way in which the redirector communicates over a network, followed by a discussion of security features offered by Windows NT systems when applications use the redirector. The next two chapters cover the mailslot and named pipe technologies, which depend solely on the redirector for all network communications.