B

MPR Support

The Multiple Provider Router (MPR) exports general networking APIs in Win32 and interacts with underlying network providers to provide the exported networking services. Applications do not interact with the network provider DLLs directly; rather, they invoke the common networking APIs and are thereby protected from the vagaries of specific network providers. Also, a common look and feel is presented by the MPR to applications that request such networking services.

If you design and implement a network redirector, you may choose to implement a network provider dynamic link library (DLL); this will allow you to leverage existing commands and interfaces (e.g., the net command) that users can utilize to request services from your network redirector. Such services can include determining the capabilities of your network, establishing a connection to a remote resource, getting information about connected resources, closing connections, and so on.

The MPR will dynamically load your DLL and call the appropriate entry points whenever your network is active.

Registry Modifications

The MPR examines the contents of the following key in the Registry to determine the various network provider DLLs that are present and also to determine the order in which these network providers should be invoked:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\NetworkProvider\order
```

The order key has a value-entry called ProviderOrder, which is a string-type value. The string value is a comma-separated list of key names. Each key name

identifies a network provider by referring to the Registry key associated with that provider. Each key name is actually a relative path from HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\, defining a node that the network vendor would have created during its installation.

As an example, consider the following entry in the Registry:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\NetworkProvider\
    order\ProviderOrder = "LanmanWorkStation,YourNetworkServiceKeyName"
```

This informs the MPR that it should expect to find two specific Registry key entries:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanmanWorkStation
```

and

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\
    YourNetworkServiceKeyName
```

It also informs the MPR that the order in which requests should be directed to the network providers present on the system is first, to the LAN Manager Work Station network provider and then your network provider.

You are expected to have the following entries in the Registry associated with the YourNetworkServiceKeyName key:

- `Group:REG_SZ:NetworkProvider`
- `NetworkProvider (subkey)`

  The NetworkProvider subkey should have the following values:

  Name *(REG_SZ)*

  The name of the network provider. This name is displayed to the user as the name of the network in the browse dialogs.

  ProviderPath (REG_EXPAND_SZj>

  The full path of the DLL that implements the network provider. MPR will perform a LoadLibrary () on this path.

Network Provider DLL Implementation

On Windows NT platforms, your network provider can implement one or more of the following functions:

Function Name	Ordinal Value
NPGetConnection()	12
NPGetCaps ()[1]	13
NPDeviceMode()	14

Function Name	Ordinal Value
NPGetUser()	16
NPAddConnection()	17
NPCancelConnection()	18
NPPropertyDialog()	29
NPGetDirectoryType()	30
NPDirectoryNotify()	31
NPGetPropertyText()	32
NPOpenEnutn()	33
NPEnumResource()	34
NPCloseEnum()	35
NPSearchDialog()	38

[1] This is the only function that is mandatory for your network provider to implement since it is the method by which the user (and the MPR DLL) can determine the capabilities of your network.

Note that your DLL does not need to contain stubs for those functions that are not supported and/or implemented by your network provider.

When implementing the network provider DLL, you should keep the following points in mind:

Speed

When your network provider DLL gets invoked, you should quickly try to determine whether the target resource is one that belongs to you or not. If your DLL does not own the resource, return WN_BAD_NETNAME (the list of error code definitions is given later in this appendix) so that the MPR can continue cycling through the list of available providers.

Validation

Your network provider DLL must validate calls using the following ordering sequence:

a. First, check if your network has been started (or if your network redirector is loaded and active).

b. Next, check if you support the requested operation.

c. If any network resources are specified, check whether you own such resources.

d. Validate the supplied parameters to your function call (if any).

Routing

The MPR cycles through all of the network providers listed in the Registry, until one of them accepts the request and processes it or until all of the available network providers have been invoked and none of them accepts the

request. If your network provider is invoked and you do not wish to process the request, return an appropriate error code (e.g., ERROR_BAD_NETPATH, ERROR_BAD_NET_NAME, ERROR_INVALID_PARAMETER, ERROR_INVALID_LEVEL). If, however, your network provider returns an error code that is a significant error code (e.g., ERROR_INVALID_PASSWORD) that indicates that the operation was processed unsuccessfully or if your network provider DLL returns a success code, the MPR DLL conveys the results back to the requesting application (and stops routing the request to any other network providers).

Return Values/Errors

Functions implemented in your network provider DLL can return either WN_SUCCESS or an appropriate error code. If returning an error, the function should also invoke the WNetSetLastError() or SetLastError() function calls to report the error. If you are returning a general error (such as insufficient memory), simply invoke the SetLastError() function; otherwise, use the WNetSetLastError() function:

```
VOID WNetSetLastError (
    DWORDerror,
    LPTSTRlpError,
    LPTSTRlpProvider)
```

where the arguments are as follows:

error
: The error code value. If this is a Windows-defined error code, lpError is ignored. Otherwise, you could set this to ERROR_EXTENDED_ERROR to indicate that a network-specific error is being reported.

lpError
: A string describing the network-specific error.

lpProvider
: A string naming the network provider that raised the error.

To report general errors, execute the following steps:

```
// error condition occurs that should be reported.
// error code is contained in providerError.
SetLastError(providerError);
return(providerError);
```

To report network-specific errors, do the following:

```
// IpErrorString contains the error to be reported.
WNetSetLastError(ERROR_EXTENDED_ERROR, IpErrorString, IpProviderName);
return(ERROR_EXTENDED_ERROR);
```

Note that the NtGetCaps () function does not return any error code value; rather, it returns a capabilities mask value.

Here are the possible status code values that can be returned (your provider, however, is free to return any Windows-defined error code):

```
#define WN_SUCCESS          00h // success
#define WN_NOT_SUPPORTED    01h // function not supported
#define WN_NET_ERROR        02h // miscellaneous network error
#define WN_MORE_DATA        03h // warning: buffer too small
#define WN_BAD_POINTER      04h // invalid pointer specified
#define WN_BAD_VALUE        05h // invalid numeric value specified
#define WN_BAD_PASSWORD     06h // incorrect password specified
#define WN_ACCESS_DENIED    07h // security violation
#define WN_FUNCTION_BUSY    08h // this function cannot be reentered and
                                // is currently being used, OR
                                // the provider is still initializing and
                                // is not ready to be called yet.
#define WN_WINDOWS_ERROR    09h //a required Windows function failed
#define WW_BAD_USER         0Ah // invalid user name specified
#define WN_OUT_OF_MEMORY    0Bh // out of memory
#define WN_NOT_CONNECTED    30h // device is not redirected
#define WN_OPEN_FILES       31h // connection could not be canceled
                                // because files are still open
#define WN_BAD_NETNAME      32h // network name is invalid
```

Note that the WN_FUNCTION_BUSY code value is also used to indicate that the network provider DLL is currently initializing itself. When this error code is returned to the application, it is possible that the application may decide to retry the operation.

NPGetCaps()

This function allows your network provider DLL to specify which functions are supported by your network from the set of functions specified by the caller. This function is defined as follows:

```
DWORD NPGetCaps(
    IN DWORD nlndex)
```

Parameters

nlndex

Capability set that the caller is interested in.

The return value is typically a bit mask, indicating which of the specified services are supported by your network provider DLL. If you return 0, the caller will take that to mean that none of the specified services are supported by your network. For certain nlndex values, however, you must return a constant value instead of a bit mask.

*Possible **nindex** values*

Version information

The nindex value will be set to WNNC_SPEC_VERSION (= 0x01).

Set the high word of the return code to 4 (indicating the major version number) and the low word to 0 (for the minor version number).

Network provider type

The nindex value will be set to WNNC_NET_TYPE (= 0x02).

The high word of the returned value should contain the provider type and the low word should contain the subtype (if any). The following types have been defined by Microsoft:*

```
#define WNNC_NET_NONE     0x00000
#define WNNC_NET_MSNET     0x10000
#define WNNC_NET_LANMAN   0x20000
#define WNNC_NET_NETWARE 0x30000
#define WNNC_NET_VINES    0x40000
```

Network provider version

The nindex value will be set to WNNC_DRIVER_VERSION (= 0x03).

Returns your driver version number.

User information

The nindex value will be set to WNNC_USER (= 0x04).

If you support this capability, return the WNNC_USR_GETUSER (= 0x01) constant value.

Connection manipulation

The nindex value will be set to WNNC_CONNECTION (= 0x06).

Set any of the following bit fields:

```
#define WMNC_CON_ADDCONNECTION        0x01
#define WNNC_CON_CANCELCONNECTION     0x02
#define WNNC_CON_GETCONNECTIONS       0x04
#define WNNC_CON_ADDCONNECTION3       0x08
```

Provider-specific dialogs

The nindex value will be set to WNNC_DIALOG (= 0x08).

Set any of the following bit fields:

```
#define WNNC_DLG_DEVICEMODE         0x01
#define WNNC_DLG_PROPERTYDIALOG    0x20 // PropertyText is also
                                        implied.
```

* You can request Microsoft to assign a provider type value for your use.

```
#define WNNC_DLG_SEARCHDIALOG          0x40
#define WNNC_DLG_FORMATNETWORKNAME     0x80
#define WNNC_DLG_PERMISSIONEDITOR      0x100
```

Administrative functionality

The nlndex value will be set to WNNC_ADMIN (= 0x09).

Set any of the following bit fields:

```
#define WNNC_ADM_GETDIRECTORYTYPE    0x01
#define WNNC_ADM_DIRECTORYNOTIFY     0x02
```

Enumeration

The nlndex value will be set to WNNC_ENUMERATION (= OxOB).

Set any of the following bit fields:

```
#define WNNC_ENUM_GLOBAL   0x01
#define WNNC_ENUM_LOCAL    0x02
```

Startup

The nlndex value will be set to WNNC_START (= OxOC).

The MPR issues this request to determine how long it should wait *(.timeout value)* for network providers to start. Therefore, if your network provider is not responding (or returns busy), the MPR may decide to retry an operation, depending upon the current timeout value and the elapsed time interval. The default value is set to 60 seconds for each network provider. The administrator could, however, change this default value by specifying the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\NetworProvider\RestoreTimeout value (type REG_DWORD), which determines the timeout in milliseconds for all network providers.

If you return 0, the MPR will assume that your network provider is disabled. If you return OxFFFFFFFF, the MPR interprets this to mean that you do not know how long it will take you to start and will wait for the current default timeout value (60 seconds or whatever is specified by the administrator).

NOTE There is a single timeout value used by the MPR for all network providers. If your network provider DLL returns a timeout value that is greater than the current MPR default timeout (whether specified by the administrator or not), the MPR will use your specified timeout value. Therefore, be judicious in determining the appropriate timeout value you decide to return.

Consult the SDK documentation, and the documentation supplied on the Microsoft Developers Library CD-ROM for more information on how to implement the other network provider APIs for the Windows NT operating system.I