Part III

Appendixes

Appendix A

Details of System Calls with Parameters

THE DOCUMENTATION OF MOST of the system services provided by Windows NT follows. Most of these system services are undocumented. Each entry includes a prototype of the system service, its parameters and descriptions of them, return codes, and in some cases, comments.

This documentation is useful for many reasons, among them:

+ To put a hook into a system service, you must know the parameters it expects so you can write a new hook service with the same prototype.
+ Few services have no corresponding Win32 API. These are truly undocumented services. To use these services, one must know the parameters they expect.
+ Although it seems to be Microsoft's position to keep these system services undocumented because they might change in future versions, we have observed that most of these system services are core and largely unchanged in versions of Windows NT to date. New system services are being added to this list with each new version of Windows NT.

```
NtLoadDriver
NTSTATUS
    NtLoadDn ver(
        IN PUNICODE_STRING DriverRegistryEntry
    ):
```

NtLoadDriver loads the device driver specified by the Registry key under HKLM\System\CurrentControlSet\Device. For example, the device driver named xxx has a Registry key "xxx" under HKLM\System\CurrentControlSet\Device.

PARAMETERS

DriverRegistryEntry	Points to the Unicode string containing the name of the Registry key for the driver where the configuration information for the driver is kept. The parameter is of the form HKLM\System\CurrentControlSet\Device\xxx, where xxx stands for device driver named xxx.

RETURN VALUE
Returns STATUS_SUCCESS on success and an appropriate error code on failure.

COMMENTS
Only users who have the privilege to load/unload device drivers can use this API.

EQUIVALENT WIN32 API
Service Control Manager APIs such as CreateService, StartService, and so on.

NtUnloadDriver

```
NTSTATUS
    NtUnloadDriver(
        IN PUNICODE_STRING DriverRegistryEntry
    );
```

NtUnLoadDriver unloads the device driver specified by the Registry key under HKLM\System\CurrentControlSet\Device. For example, the device driver named xxx has a Registry key "xxx" under HKLM\System\CurrentControlSet\Device.

PARAMETERS

DriverRegistryEntry	Points to the Unicode string containing the name of the Registry key for the driver where the configuration information for the driver is kept. The parameter is of the form HKLM\System\CurrentControlSet\Device\xxx, where xxx stands for the device driver named xxx.

RETURN VALUE
Returns STATUS_SUCCESS on success and an appropriate error code on failure.

COMMENTS
Only users who have the privilege to load/unload device drivers can use this API.

EQUIVALENT WIN32 API
Service Control Manager APIs such as StopService, DeleteService, and so on.

NtClose

```
NTSTATUS
    NtClose(
      - IN HANDLE Handle
    ):
```

NtClose closes the open handle to the executive object. This could be any handle, such as a handle to the mutex, semaphore, and so on.

PARAMETERS

Handle	Handle to the object.

RETURN VALUE

Returns STATUS_SUCCESS on success and an appropnate error code on failure.

COMMENTS

Every object has a handle count and a reference count associated with it. The *handle count* represents number of open handles to the object. The *reference count* represents the number of pointer references to the object. The object is removed from memory only when the object handle count and reference count reaches zero.

EQUIVALENT WIN32 API

CloseHandle

```
NtDuplicateObject
NTSTATUS
     NtDuplicateObject(
          IN HANDLE hSourceProcessHandle,
          IN HANDLE hSourceHandle,
          IN HANDLE hTargetProcessHandle,
          IN OUT PHANDLE hTargetHandle,
          IN ACCESS_MASK AccessMask,
          IN BOOLEAN bInheritHandle,
          IN ULONG dwOptions
);
```

NtDuplicateObject creates a new handle to the given object in arbitrary process's context.

PARAMETERS

hSourceProcessHandle	Handle to the process in which the handle to be duplicated resides.
hSourceHandle	Handle to the object to be duplicated.
hTargetProcessHandle	Handle to the process in which the handle is duplicated.
TargetHandle	Pointer to the variable where the duplicated handle is received.
AccessMask	Access requested for the new handle.

blnheritHandle	Boolean value describing whether the handle is inherited by child processes spawned by the process and represented by TargetProcessHandle.
dwOptions	Flags that affect the behavior of the system service. If DUPLICATE_SAME_ACCESS is specified, then the AccessMask parameter is ignored. If DUPLICATE_CLOSE_SOURCE is specified, then the source handle is closed after duplication.

RETURN VALUE
Returns STATUS_SUCCESS on success and an appropriate error code on failure.

COMMENTS
Console handles cannot be duplicated using this system service.

EQUIVALENT WIN 32 API
DuplicateHandle

NtCreateDirectoryObject

```
NTSTATUS
    NtCreateDirectoryObject(
        OUT PHANDLE hDirectory,
        IN ACCESS_MASK AccessMask,
        IN POBJECT_ATTRIBUTES ObjectAttributes
);
```

NtCreateDirectoryObject creates a new directory object.

PARAMETERS

hDirectory	Pointer to the variable that receives a handle to the directory object.
AccessMask	Type of access requested to the directory object. This can be a combination of any of the following flags: DIRECTORY_QUERY, DIRECTORY_TRAVERSE, DIRECTORY_CREATE_OBJECT, DIRECTORY_CREATE_SUBDIRECTORY, and DIRECTORY_ALL_ACCESS.
ObjectAttributes	Points to the OBJECT_ATTRIBUTES structure containing the information about the directory object to be created, such as name, parent directory, objectflags, and so on.

RETURN VALUE
Returns STATUS_SUCCESS on success and an appropriate error code on failure.

COMMENTS
None.

EQUIVALENT W1M32 API
None.

NtCreateSymbolicLinkObject

```
NTSTATUS
    NtCreateSymbolicLinkObject(
        OUT PHANDLE hSymbolicLink,
        IN ACCESS_MASK AccessMask,
        IN POBJECT_ATTRIBUTES ObjectAttributes,
        IN PUNICODE_STRING SymbolicLinkValue
);
```

NtCreateSymbolicLinkObject creates a new symbolic link.

PARAMETERS

hSymbolicLink	Pointer to the variable that receives handle to the SymbolicLink object.
AccessMask -	Type of access requested to the symbolic link object. This can be a combination of any of the following flags: SYMBOLIC_LINK_QUERY or SYMBOLIC_LINK_ALL_ACCESS.
ObjectAttributes	Points to the OBJECT_ATTRIBUTES structure containing the information about the symbolic link object to be created, such as name, parent directory, objectflags, and so on.
SymbolicLinkValue	Points to a Unicode string containing the object name this symbolic link refers to.

RETURN VALUE
Returns STATUS_SUCCESS on success and an appropriate error code on failure.

COMMENTS
The user-mode API call DefineDosDevice enables you to create a symbolic link object only under the object directory named "\??", whereas this system service enables you to create a symbolic link object anywhere in the object name space

provided you have permission. There is a symbolic link named "\DosDevices" which points to "\??" object directory.

EQUIVALENT WIN32 API

DefmeDosDevice (limited support)

NtMakeTemporaryObject

```
NTSTATUS
    NtMakeTemporaryObject(
        IN HANDLE hObject
);
```

NtMakeTemporaryObject converts the permanent object into a temporary object.

PARAMETERS

hObject A Handle to the permanent object.

RETURN VALUE

Returns STATUS_SUCCESS on success and an appropriate error code on failure.

COMMENTS

The objects created with the OBJ_PERMANENT attribute in ObjectAttributes's Attributes member can be converted into a temporary object using this function. Permanent objects with names are not deleted from the Object Manager name space even when the handle count reaches zero. This function can be used to delete permanent named objects with handle count zero from the Object Manager name space.

EQUIVALENT WIN32 API

None.

NtOpenDi rectoryObject

```
NTSTATUS
    NtOpenDirectoryObject(
        OUT PHANDLE hDirectory,
        IN ACCESS_MASK AccessMask,
        IN POBJECT_ATTRIBUTES ObjectAttnbutes
):
```

NtOpenDirectoryObject opens an existing directory object in the Object Manager name space.

PARAMETERS

hDirectory	Pointer to the variable that receives a handle to the directory object.
AccessMask	Type of access requested to the directory object. This can be a combination of any of the following flags: DIRECTORY_QUERY, DIRECTORY_TRAVERSE, DIRECTORY_CREATE_OBJECT, DIRECTORY_CREATE_SUBDIRECTORY, and DIRECTORY_ALL_ACCESS.
ObjectAttributes	Points to the OBJECT_ATTRIBUTES structure containing the information about the directory object to be opened, such as name, parent directory, objectflags, and so on.

RETURN VALUE
Returns STATUS_SUCCESS on success and an appropriate error code on failure.

COMMENTS
None.

EQUIVALENT WIN32 API
None.

NtQueryDirectoryObject

```
NTSTATUS
    NtQueryDirectoryObject(
        IN HANDLE hDirectory,
        OUT PVOID DirectoryEntryBuffer,
        IN ULONG DirectoryEntryBufferSize,
        IN BOOLEAN bOnlyFirstEntry,
        IN BOOLEAN bFirstEntry,
        OUT PULONG BytesReturned,
        OUT PULONG EntryIndex
);
```

NtQueryDirectoryObject returns individual object names in the given object directory along with the type of these objects.

PARAMETERS

hDirectory	Handle to a directory opened using NtOpenDirectory.

DirectoryEntryBuffer	Pointer to the buffer that receives the object names and object types in the given object directory.
DirectoryEntryBufferSize	Size of the buffer pointed to by DirectoryEntryBuffer.
bAllEntries	Flag indicating whether you are interested in all the entries in the given object directory.
bFirstEntry	Flag indicating that the search should start from the first entry in the directory.
BytesReturned	Pointer to the variable that receives the number of bytes copied into the buffer pointed to by DirectoryEntryBuffer.
Entrylndex	Pointer to the variable that returns the index corresponding to the object entry returned.

RETURN VALUE

Returns STATUS_SUCCESS on success and an appropriate error code on failure.

COMMENTS

To enumerate all the objects in a given object directory, you need to first call this function with bFirstEntry set to TRUE, and then call this function with bFirstEntry set to FALSE. The function returns STATUS_NO_MORE_ENTRIES when all the entries in a given object directory are over. bAllEntries should be set to TRUE in this case.

Data in DirectoryEntryBuffer is of variable length based on the object names and object types. The fixed portion of this data is as follows:

```
typedef struct DirectoryEntryBuffer_t {
      UNICODE_STRING ObjectName,
      UNICODE_STRING ObjectType
} DIRECTORY_ENTRY_BUFFER
```

This is followed by ObjectName and ObjectType in wide character format.

EQUIVALENT WIN32 API

None.

```
NtOpenSymbolicLinkObject
NTSTATUS
     NtOpenSymbolicLinkObject(
          OUT PHANDLE hSymbolicLink,
```