# Chapter 6
# NIOS APIs for MS Windows

## Get NIOS Windows 16-Bit Mode API

**Description**

Use the following steps to access the NIOS APIs available to 16-bit MS Windows applications. These APIs provide, among other things, a method to invoke most exported NLM APIs from a 16-bit Windows application.

Locate the NIOS 16-bit Windows application interfaces by issuing an Int 2Fh as shown below. If AX returns set to 0000h then the API entry points are available.

On entry:
*ax* 0D8C3h

On return:

| | |
|---|---|
| *ax* | 0000h (NIOS is present) |
| *bx* | Version of loaded NIOS module |
| *bh* | has major version, bl has minor version |
| *esi* | Sel:Off of NIOS far call handler (refer to **Win16NiosFarCallHandler** for more information) |
| *ecx* | Sel:Off of NIOS function used to invoke "C" callable NLM functions (refer to **Win16InvokeCNlmApiHandler** for more information) |
| *edx* | Sel:Off of NIOS function used to invoke register-based NLM functions (refer to **Win16InvokeRegNlmApiHandler** for more information) |

All other registers preserved

**See Also**

Win16NiosFarCallHandler
Win16InvokeRegNlmApiHandler
Win16InvokeCNlmApiHandler

**See Also**

## Win16InvokeRegNlmApiHandler

**Description**

This function is used by 16-bit Windows applications to call (invoke) exported NLM functions that use register-based calling conventions.

**Assumes**

*apiAddress* Pushed onto the stack
*eax,ebx,ecx,edx,esi,edi,ebp*
Set up as specified for the NLM API

**Returns**

General purpose registers set up as defined by NLM API
All segment registers are preserved
*apiAddress* is removed from stack

**Remarks**

Use the procedure outlined in "Get NIOS Windows 16-Bit Mode API" to get the **Win16InvokeRegNlmApiHandler** far call address.

Data pointer parameters passed to asynchronous NLM APIs must typically be page-locked by the application (for example, GlobalPageLock).

**See Also**

Get NIOS Windows 16-Bit Mode API Win16InvokeCNlmApiHandler

## Win16InvokeCNlmApiHandler

### Description

16-bit MS Windows applications use this function to call (invoke) exported NLM functions that use "C" calling conventions.

### Syntax

```
(*Win16InvokeCNlmApiHandler)(
    UINT32      apiAddress,
    UINT32      apiParmCount,
    ...);
```

### Parameters

*apiAddress* Address of NLM API to invoke. Use **PM16_NIOS_BEGIN_USE_API** to get this value.

*apiParmCount* Number of UINT32 stack parameters needed for call. This value defines the number of UINT32 values that need to be copied from the application's stack onto the Ring-0 protected-mode stack prior to invoking the specified NLM API.

... Parameters to NLM API.

### Returns

Defined by NLM API
UINT32 values are returned in registers DX:AX

### Remarks

Use the procedure outlined in "Get NIOS Windows 16-Bit Mode API" to get the **Win16InvokeCNlmApiHandler** far call address.

Data pointer parameters passed to asynchronous NLM APIs must typically be page-locked by the application (for example, GlobalPageLock).

### See Also

Get NIOS Windows 16-Bit Mode API
Win16InvokeRegNlmApiHandler

## Win16LoadModule

**Description**

Called by 16-bit MS Windows applications to load an NLM.

**Syntax**

```
UINT32
Win16LoadModule(
     UINT32          loadOptions,
     UINT8           FAR16  *modulePathSpec,
     UINT8           FAR16  *commandLine,
     UINT32          nlmFileOffset,
     modHandle       *retModHandle,
     void            (FAR16  *msgHandler)(
                               modHandle       module,
                               UINT8           *prefix,
                               UINT8           *msg) );
```

**Parameters**

*loadOptions* Bits defining load styles. All undefined bits must be set to zero.

LOPTION_DEBUG_INIT
Executes an Int 1 before the loader invokes the module's initialization routine.

LOPTION_ERROR_MSGS
Standard output error messages are enabled.

LOPTION_BANNER_MSGS
Standard output sign on messages are enabled.

*modulePathSpec* Module [path\]name to load (with extension).

*commandLine* Pointer to any parameters that will be passed to the loading module. This is an ASCIIZ string.

*nlmFileOffset* Offset from the start of the modulePathSpec file where the NLM image starts. This will typically be zero for straight NLM files.

*retModHandle* Pointer to a module handle that will be set to the newly loaded module's handle on success. If NULL, the module handle will not be returned.

*msgHandler* Pointer to function which will be called when a text message is displayed during the load process. Parameters to this function are flat linear addresses; therefore the handler must either map a selector to them or use the appropriate NIOS functions to copy the memory.

**Returns**

LOADER_SUCCESS
Module was loaded successfully

LOADER_NO_LOAD_FILE
Open load file failed

LOADER_IO_ERROR
IO file error during read

LOADER_INSUFFICIENT_MEMORY
Not enough memory to load module

LOADER_INVALID_MODULE
Invalid NLM module

LOADER_UNDEFINED_EXTERN
Referenced undefined external item

LOADER_DUPLICATE_PUBLIC
Exported public is already defined

LOADER_NO_MSG_FILE
Open message file failed

LOADER_INVALID_MSG_MODULE
Message file is malformed

LOADER_MODULE_ALREADY_LOADER
Module cannot be loaded more than once

LOADER_BAD_REENTRANT_MODULE
Reentrant load failed because the module is not the same version as the first module

LOADER_MODULE_INIT_FAILED

Module failed to initialize

LOADER_LOAD_REFUSED
A loaded NLM refuses to allow this NLM to load

**Remarks**

All pointer parameters are passed in as selector:offset.

Windows applications needing to load an NLM typically will use this function instead of **NiosLoadModule**, since they will want to obtain text output messages from the NLM and loader while the load is taking place.

It is possible to invoke **NiosLoadModule** with the LOPTION_ERROR_MSGS set to zero from an MS Windows application, since this causes a silent load to take place.

**See Also**

## Win16NiosFarCallHandler

**Description**

This function is invoked by 16-bit Windows applications using the address obtained using the procedure outlined in Get NIOS Windows 16-Bit Mode API.

**Syntax**

```
#include <nlmapi.h>

UINT32
(*Win16NiosFarCallHandler)(
    UINT32      function,
    ...);
```

**Parameters**

*function* One of the following values:

| | | |
|---|---|---|
| PM16_NIOS_BEGIN_USE_API | equ | 00000000h |
| PM16_NIOS_END_USE_API | equ | 00000001h |
| PM16_NIOS_COPY_MEM | equ | 00000002h |
| PM16_NIOS_COPY_STRING | equ | 00000003h |

... Other parameters as needed

**Returns**

Values specific to each function

0x80000000 Invalid function request value

**Remarks**

Note that 32-bit return values are returned in registers DX:AX.

**See Also**

## Win16UnloadModule

**Description**

Called by 16-bit MS Windows applications to unload an NLM.

**Syntax**

```
UINT32
Win16UnloadModule(
     modHandle      modHand,
     UINT32         unloadOptions,
     void           (FAR16  *msgHandler)(
                         modHandle       module,
                         UINT8           *prefix,
                         UINT8           *msg) );
```

**Parameters**

*modHand*  Handle of module to unload. This is a flat linear address of a module handle for the NLM to unload.

*unloadOptions*  Bits defining unload options. All undefined bits must be set to zero.

UOPTION_ERROR_MSGS
Standard output error messages

*msgHandler*  Pointer to function which will be called when a text message is displayed during the unload process. Parameters to this function are flat linear addresses; therefore the handler must either map a selector to them or use the appropriate NIOS functions to copy the memory.

**Returns**

UNLOAD_SUCCESS
Module was unloaded

UNLOAD_MODULE_FORBIDS_UNLOAD
Module does not allow unload

UNLOAD_MODULE_BEING_REFERENCED
Another module is using this module

UNLOAD_INVALID_MODULE_HANDLE
Module handle is invalid

UNLOAD_RESOURCES_NOT_FREED
Module did not free resources

UNLOAD_MODULE_CANT_UNLOAD_NOW
Module is temporarily unable to unload

UNLOAD_UNLOAD_REFUSED
A loaded NLM refuses to allow this NLM to load

**Remarks**

All pointer parameters are passed in as selector:offset.

**See Also**

## PM16_NIOS_BEGIN_USE_API

**Description**

Determines the 32-bit flat linear address of the specified NLM API name. The returned address can then be used with either the **Win16InvokeCNlmApiHandler** or the **Win16InvokeRegNlmApiHandler** far call handlers to actually invoke the NLM function from a 16-bit MS Windows application.

**Syntax**

```
UINT32
(*Win16NiosFarCallHandler)(
     UINT32      PM16_NIOS_BEGIN_USE_API,
     UINT8       FAR16 *apiName);
```

**Parameters**

*apiName* Name of the API you would like to call. This is a case-insensitive ASCIIZ string, for example, "CNWIpxSendPacket".

**Returns**

| | |
|---|---|
| 0 | API does not exist |
| !0 | Linear address of API |

**Remarks**

This function records a dependency for the NLM module in which the API function exists, so it is important that the MS Windows application use **PM16_NIOS_END_USE_API** before the application terminates.

**See Also**

PM16_NIOS_END_USE_API

## PM16_NIOS_COPY_MEM

**Description**

Copies *length* bytes of the memory at the specified protected-mode linear address into the specified 16-bit sel:off buffer.

**Syntax**

```
void
(*Win16NiosFarCallHandler)(
    UINT32    PM16_NIOS_COPY_MEM,
    void      FAR16 *destBuffer,
    UINT32    pmBuffer,
    UINT32    length);
```

**Parameters**

| | |
|---|---|
| *destBuffer* | Pointer to sel:off buffer to which to copy |
| *pmBuffer* | Linear address of protected-mode buffer from which to copy |
| *length* | Number of bytes to copy |

**Returns**

Nothing

**Remarks**

**See Also**

## PM16_NIOS_COPY_STRING

**Description**

Copies the string pointed to by *pmBuffer* into the specified 16-bit sel:off buffer.

**Syntax**

```
void
(*Win16NiosFarCallHandler)(
    UINT32      PM16_NIOS_STRING,
    void        FAR16 *destBuffer,
    UINT32      pmBuffer);
```

**Parameters**

| | |
|---|---|
| *destBuffer* | Pointer to sel:off buffer to copy to |
| *pmBuffer* | Linear address of string |

**Returns**

Nothing

**Remarks**

**See Also**

## PM16_NIOS_END_USE_API

**Description**

Signals that the MS Windows application is no longer going to use the specified NLM API function. This deletes the dependency that was previously created using PM16_NIOS_BEGIN_USE_API.

**Syntax**

```
void
(*Win16NiosFarCallHandler)(
    UINT32      PM16_NIOS_END_USE_API,
    UINT32      apiLinAddress);
```

**Parameters**

| | |
|---|---|
| *apiLinAddress* | Linear address of NLM API function |

**Returns**

Nothing

**Remarks**

**See Also**

PM16_NIOS_BEGIN_USE_API
NE_WIN_VM_SUSPEND