Chapter 8 HACB Type Zero Functions

This chapter gives descriptions of HACB Type Zero functions. HACB type zero functions are called for HACB requests where the **HACBType** field has a value of zero. The NWPA requires a HAM to implement as many of these functions as are applicable to its adapter type. **HACBType=0** requests ask for information about the HAM, the host adapter, attached devices, sets a device's queue state, or monitors asynchronous hardware events. The HAM receives requests of this type through the union to Host command block of the HACB. Currently, there are seven HACB type zero functions that HAMs are required to implement; however, Novell has reserved up to 256 of these functions for the future. For HAMs to determine which of the functions to execute along with any applicable input parameters, the Host command block of the HACB provides the following information:

- The desired function's number.
- Any applicable input parameters.

All data transfers must be handled as any other HACB request. This means that data is either placed/retrieved in/from the data buffer fields of the HACB (**VirtualAddress** or **PhysicalAddress**). Also, the buffer's size (in bytes) is specified in the HACB's **DataBufferLength** field by the process placing the data. The functional descriptions in the following sections detail how the above information should be interpreted by the HAM in order to complete the **HACBType=0** request. The functions are listed in the order of their respective function numbers.

HAM_Return_HAM_Info (Function 0)

HAM_Return_HAM_Info is responsible for supplying the NWPA with information about the HAM. The structure of the return information is defined by the **HAMInfoStruct**. *HAM_Return_HAM_Info* can be identified when the following information is in the fields of the HACB's host command block:

Function = 0
parameter0 = 0
parameter1 = 0
parameter2 = 0

As part of this function, the HAM should copy the HAM information into the buffer pointed at by the HACB's **VirtualAddress** field.

HAM_Scan_For_Devices (Function 1)

This function tells the HAM to scan (probe) for devices attached to a specified bus and build a list of detected devices on that bus. For each device in its list, the HAM is expected to fill out an instance of a **DeviceInfoStruct**, which includes a HAM-generated, unique access handle (**DeviceHandle**) to the device. The HAM will receive one scan request for each bus instance it registers using **HAI_Activate_Bus()**.

Note: The HACB indicating this function is received on a non-blocking thread. Since a bus scan could take milliseconds or even seconds to complete, the HAM should spawn (schedule) a separate, blocking thread to perform the physical scan. Blocking threads can be created using **NPA_Spawn_Thread()**.

HAM_Scan_For_Devices must handle four different scan cases according to the parameters given in the HACB's host command block. The NWPA classifies each scan case as either Case 0, Case 1, Case 2, or Case 3 corresponding to the value received in the parameter2 field of the host command block. Case 0 scans are issued by the OS, and Cases 1 - 3 scans are issued by a CDM.

These scan cases also tell the HAM whether to declare a device *public* or *private*. The next subsection, "Public Vs. Private Devices", defines this concept in detail.

The objectives of each scan case are as follows:

Case 0: This scan case tells the HAM to perform a typical "scan for new devices"[1] by probing all public target locations. These locations include LUN 0 (and any other LUNs as specified under Case 2) of all target IDs on the specified bus. This scan case is issued by the OS, and an object for each device is created and added to the Media Manager's device database.

Case 1: This scan case tells the HAM to probe a particular LUN on a particular target ID of the bus. If a device is detected, it is declared private, and information about the device is returned as defined by the NWPA's **DeviceInfoStruct**. A device object is not created for the Media Manager's device database. This scan case is issued by a CDM allowing it to probe LUNs other than zero of the specified target ID to detect additional devices that enhance the functionality of the LUN 0 device. An example would be a magazine, addressed at LUN 1, attached to a tape drive at LUN 0.

Case 2: This scan case has the same function and purpose as the Case 1 scan except that the target device is declared public, and a device object

[1] The command "scan for new devices" is a NetWare command that can be issued from any .NCF file or at the command line by the user. The purpose of "scan for new devices" is to make storage devices attached to the server visible to the NetWare OS.

is created and added to the Media Manager's device database.

Case 3: This scan tells the HAM to discard the device object, associated with the specified LUN and target ID, from its device list. This scan case is issued by a CDM.
The remaining subsections provide more details by describing the following:

- Public versus private devices
- Scan Case Definitions and Descriptions
- Scan completion codes.

Public vs. Private Devices

The HAM does not decide the public/private attribute of a device. This decision is made either by the OS (Case 0) or a CDM (Cases 1 and 2) according to the scan case issued. The HAM declares a device public by clearing the **Private_Public_Flag** in the **AttributeFlags** field of the device's **DeviceInfoStruct**. To declare a device private, the HAM sets the **Private_Public_Flag**. All devices detected by Case 0 and Case 2 scans should be declared public. All devices detected by a Case 1 scan should be declared private.

Following a Case 0 or Case 2 scan, the HAM's *HAM_Return_Device_Info* function gets called on the target bus, which reports the bus's device list to the NWPA. For all public devices in the list, the NWPA creates a corresponding device object and adds it into the Media Manager's device database. The Media Manager then makes these devices visible to the application layer. This is the criteria that defines a public device.

The criteria that defines a private device is that the device's **Private_Public_Flag** is set, and there is no corresponding object added to the Media Manager's device database. Thus, the device is not visible to the application layer. The NWPA makes private devices visible only to the CDM layer.

Scan Case Definitions and Descriptions

This subsection defines the HACB's host command block parameters associated with each scan case along with a paradigm description of the how the HAM should handle each case.

Case 0: Probe all LUN 0's and Make Detected Devices Public

INPUTS from HACB:
Field Values in Host Command Block:
Function = 1
parameter0 = Unit Number Mask
parameter1 = Target ID Mask (-1 will scan all target IDs)
parameter2 = 0

OUTPUTS to HACB:
hacbCompletion = Appropriate scan completion code.

A Case 0 scan request is issued by the NetWare OS following a "scan for new devices" command issued either in a .NCF file or at the command line by a user.

When the HAM receives a Case 0 scan request, it is expected to do the following:

1. Probe LUN 0 of all target IDs (and any other target IDs and LUNs specified under Case 2) on the target bus to detect devices.

2. For each device detected, create an object describing the device, including information defined by the NWPA's **DeviceInfoStruct**, and place it in a device list for that bus.

Note: An essential data member of the **DeviceInfoStruct** is the unique **DeviceHandle** the HAM assigns to the device. This **DeviceHandle** is the token that the NWPA and CDMs will use to route HACB I/O requests, through the HAM, to a device.

3. Make the device public by clearing the **Private_Public_Flag** in the **AttributeFlag** field of the device's **DeviceInfoStruct**.

4. After finishing the scan on the target bus, set the HACB's **hacbCompletion** field to Successful_Completion (0x00000000), complete the HACB using **HAI_Complete_HACB()**, and then return.

Case 1: Probe Specified Target ID and LUN, Return Info and Make Detected Device Private

INPUTS from HACB:

Field Values in Host Command Block:

Function = 1
parameter0 = Unit Number
parameter1 = Target ID
parameter2 = 1

Field Values in Main HACB:

DeviceHandle = -1 indicating a request for a first-time peek at the target ID and LUN. The calling CDM does not have ownership of the private device; therefore, execution restrictions apply to this request.

= A handle that maps to a device already existing in the HAM's device list. The HAM returned this device handle to the calling CDM in a previous Case 1 scan request, thereby giving the CDM exclusive ownership of the private device. Due to device ownership, no execution restrictions apply to this request.

OUTPUTS to HACB:

Control_Info = Sizeof(struct **DeviceInfoStruct**)
VirtualAddress = Pointer to the buffer where the HAM will copy the device's **DeviceInfoStruct** information.
hacbCompletion = Appropriate scan completion code.

A Case 1 scan request is issued by a CDM allowing it to inquire about the existence of a device on a specific target ID and LUN (other than LUN 0). The purpose of this scan case is to grant a CDM private access to a device attached to the target ID and LUN by providing the CDM with a DeviceHandle.

When the HAM receives a Case 1 scan request it is expected to do the following:

1. Probe the specified target ID (parameter1) and LUN (parameter0):
   A. If the HAM detects a device at this location and there is no corresponding object for this device already existing in the HAM's device list:
      1. Create an object describing the device, including information defined by the NWPA's **DeviceInfoStruct**, and place it in a device list for that bus.

Note: An essential data member of the **DeviceInfoStruct** is the unique **DeviceHandle** the HAM assigns to the device. This **DeviceHandle** is the token that the CDM will use to route HACB I/O requests, through the HAM, to a device.

      2. Make the device private by setting the **Private_Public_Flag** in the **AttributeFlag** field of the device's **DeviceInfoStruct**.

3. Copy the device's **DeviceInfoStruct** information into the buffer pointed at by the HACB's **VirtualAddress** field.

4. Place the size, in bytes, of the return buffer in step 3 in the HACB's **Control_Info** field.

5. Set the HACB's **hacbCompletion** field to Successful_Completion (0x00000000).

6. Complete the HACB using **HAI_Complete_HACB()**.

B. If the HAM detects a device at this location and a corresponding object for this device already exists in the HAM's device list:

1. If the value in the HACB's **DeviceHandle** field is equal to -1, set the HACB's **hacbCompletion** field to Target_In_Use (0x000A0003).

2. If the value in the HACB's **DeviceHandle** field matches a device handle assigned to an object in the HAM's device list, do steps 2 through 5 in part A above.

3. Complete the HACB using **HAI_Complete_HACB()**.

C. If the HAM does not detect a device at this location:

1. Set the HACB's **hacbCompletion** field to Device_Not_Found (0x000A0001).

2. If a device object exists in the HAM's device list, meaning that at one time a device did exist at the target ID and LUN but has now gone away, the HAM should remove the object from the list and free the memory using **NPA_Return_Memory()**.

3. Complete the HACB using **HAI_Complete_HACB()**.

2. Return 0.

Case 2: Probe Specified Target ID and LUN, Return Info, and Make Detected Device Public

INPUTS from HACB:

Field Values in Host Command Block:

Function = 1
parameter0 = LUN
parameter1 = Target ID
parameter2 = 2

Field Values in Main HACB:

DeviceHandle = -1 indicating a request for a first-time peek at the target ID and LUN. The calling CDM does not have ownership of the device if it is private; therefore, execution restrictions apply to this request.

= A handle that maps to a private device already existing in the HAM's device list. The HAM returned this device handle to the calling CDM in a previous Case 1 scan request, thereby giving the CDM exclusive ownership of the private device. Due to device ownership, no execution restrictions apply to this request.

OUTPUTS to HACB:

Control_Info = Sizeof(struct **DeviceInfoStruct**)
VirtualAddress = Pointer to the buffer where the HAM will copy the device's **DeviceInfoStruct** information.
hacbCompletion = Appropriate scan completion code.

A Case 2 scan request is issued by a CDM allowing it to inquire about the existence of a device on a specific target ID and LUN (other than LUN 0) and make the device public.

Note: Any device object added to the device list through a Case 2 scan should also be probed and updated in any subsequent Case 0 scans.

When the HAM receives a Case 2 scan request it is expected to do the following:

1. Probe the specified target ID (**parameter1**) and LUN (**parameter0**):
   A. If the HAM detects a device at this location and there is no corresponding object for this device already existing in the HAM's device list:
      1. Create an object describing the device, including information defined by the NWPA's **DeviceInfoStruct**, and place it in a device list for that bus.

Note: An essential data member of the **DeviceInfoStruct** is the unique **DeviceHandle** the HAM assigns to the device. This **DeviceHandle** is the token that the CDM will use to route HACB I/O requests, through the HAM, to a device.

2. Make the device public by clearing the **Private_Public_Flag** in the **AttributeFlag** field of the device's **DeviceInfoStruct**.

3. Copy the device's **DeviceInfoStruct** information into the buffer pointed at by the HACB's **VirtualAddress** field.

4. Place the size, in bytes, of the return buffer in step 3 in the HACB's **Control_Info** field.

5. Set the HACB's **hacbCompletion** field to Successful_Completion (0x00000000).

6. Complete the HACB using **HAI_Complete_HACB()**.

B. If the HAM detects a device at this location and a corresponding object for this device already exists in the HAM's device list:

1. If the value in the HACB's **DeviceHandle** field is equal to -1, set the HACB's **hacbCompletion** field to Target_In_Use (0x000A0003).

2. If the value in the HACB's **DeviceHandle** field matches a device handle assigned to an object in the HAM's device list, do steps 2 through 5 in part A above.

3. Complete the HACB using **HAI_Complete_HACB()**.

C. If the HAM does not detect a device at this location:

1. Set the HACB's **hacbCompletion** field to Device_Not_Found (0x000A0001).

2. If a device object exists in the HAM's device list, meaning that at one time a device did exist at the target ID and LUN but has now gone away, the HAM should remove the object from the list and free the memory using **NPA_Return_Memory()**.

3. Complete the HACB using **HAI_Complete_HACB()**.

2. Return 0.

Case 3: Remove Device Object from Device List

Field Values in Host Command Block:

Function	=	1
parameter0	=	LUN
parameter1	=	Target ID

parameter2 = 3

Field Values in Main HACB:

DeviceHandle = -1 indicating a request for a first-time peek at the target ID and LUN. The calling CDM does not have ownership of the device if it is private; therefore, execution restrictions apply to this request.

= A handle that maps to a private device already existing in the HAM's device list. The HAM returned this device handle to the calling CDM in a previous Case 1 or Case 2 scan request, thereby giving the CDM exclusive ownership of the private device. Due to device ownership, no execution restrictions apply to this request.

OUTPUTS to HACB:

hacbCompletion = Appropriate scan completion code.

A Case 3 scan request is issued by a CDM after it determines that a device object (found from a previous Case 1 or Case 2 scan request) is a phantom, or in other words, a reflection of the device at LUN 0. Its main purpose is to have the HAM remove the target object from its device list.

When the HAM receives a Case 3 scan request, it is expected to do the following:

1. If the target object does not exist, meaning a previous Case 1 or Case 2 scan request was never received for the target ID and LUN, set the HACB's **hacbCompletion** field to Object_Not_Found (0x000A0004).

2. If the target object exists, but the value in the HACB's **DeviceHandle** field is equal to -1, set the HACB's **hacbCompletion** field to Target_In_Use (0x000A0003).

3. If the target object exists and the value in the HACB's **DeviceHandle** field matches the target object's device handle:

   A. Remove the device object associated with the specified target ID (parameter1) and LUN (parameter0) from the device list and free the memory using **NPA_Return_Memory()**.

   B. Set the HACB's **hacbCompletion** field to Successful_Completion (0x00000000).

4. Complete the HACB using **HAI_Complete_HACB()**.

5. Return 0.

Scan Completion Codes

The following table summarizes the scan completion codes described in the specific cases above. The table also includes additional error-completion codes common to all scan cases. These completion codes get posted to the HACB's **hacbCompletion** field:

Upper WORD (16 bits)	Lower WORD (16 bits)	Description
0x0000	0x0000	Successful Completion: The current scan operation completed successfully.This completion code applies to all scan cases.For Case 1 and Case 2 scans, this completion code indicates that a device responded atthe specified Target ID and LUN, and the information returned in the HACB's data bufferis valid.
0x000A	0x0000	General Failure: Default scan-error category. The cause of the error is unknown, andany information contained in the HACB's data buffer is invalid.This completion code applies to all scan cases.
	0x0001	Device Not Found: No device responded at the specified Target ID and LUN. Anyinformation contained in the HACB's data buffer is invalid.This completion code applies to Case 1 and Case 2 scans
	0x0002	Bad Target ID/LUN: The Target ID and/or LUN specified in the HACB's host adaptercommand block was/were invalid. Any information contained in the HACB's data bufferis invalid.This completion code applies to all scan cases.
	0x0003	Target In Use: The target object is owned by another CDM. Therefore, the current scanrequest could not be executed.This completion code applies to Case 1, Case 2, and Case 3 scans.
	0x0004	Object Not Found: A CDM issued a Case 3 scan to remove a device object from theHAM's device list that does not exist. The object does not exist because no previous Case1 or Case 2 scan was issued on the specified Target ID and LUN to create it. Anyinformation contained in the HACB's data buffer is invalid.This completion code applies to Case 3 scans.
		Novell reserves the right to add additional completion codes.

HAM_Return_Device_Info (Function 2)

This routine must return a **DeviceInfoStruct** that gives information about a single device and how it works. This routine is identified by the following information in the HACB's command area:

Function = 2
parameter0 = -1 to begin find-first-find-next sequence or the handle into the HAM's device list from the last iteration of the sequence
parameter1 = Not used
parameter2 = Not used

If parameter0 = -1, it means that a new find-first-find-next sequence is to be started, and the HAM must return information about the first device in its list. The structure of the return information is defined by the **DeviceInfoStruct**, which the HAM copies into the buffer pointed at by the HACB's VirtualAddress field.[2] The HAM must perpetuate the sequence until information about all of its devices are returned.

The HAM perpetuates the sequence by doing the following:

1. Copy the device information of the first device in the HAM's device list into the return buffer.

2. Place 0x00000000 into the HACB's **hacbCompletion** field.

3. Complete the HACB by calling **HAI_Complete_HACB()**.

The sequence continues with the HAM receiving another *HAM_Return_Device_Info* request, this time, with *parameter0* equal to the device handle the HAM placed in the return buffer of the previous iteration of the sequence. The HAM then keys off this previous device handle to locate the next device in the list and returns its information. The HAM follows this paradigm until it returns information about all the devices in its list

After the HAM returns information about the last device in its list, it will receive one more *HAM_Return_Device_Info* request. The HAM ends the find-first-find-next sequence by placing a non-zero value in this last HACB's **hacbCompletion** field and completing it by calling **HAI_Complete_HACB()**.

---

[2] One key piece of information the HAM places in the buffer is the DeviceHandle. The HAM generated this DeviceHandle during the host bus scan, and now through *HAM_Return_Device_Info()*, the NWPA will pass it to the CDM. The CDM will then use this **DeviceHandle** to indicate a target device to the HAM for all subsequent device I/O.

HAM_Unfreeze_Queue (Function 3)

This function unfreezes the selected device queue and initiates execution of the next request in the queue. This function is identified by the following information in the HACB's command area:

Function = 3
parameter0 = none
parameter1 = none
parameter2 = none

This function must unfreeze the HAM's device queue if it is in the frozen state, but it should not affect the queue if the queue is in the normal, operational state.

HAM_Set_IDE_Device_Config (Function 4)

Although this function is a **HACBType=0** function, implementing it is optional. This function sets the IDE\ATA device configuration with the HAM allowing the CDM to use special commands as they appear in the IDE\ATA specification. If this function is never used to change the drive configuration, the default is used. The calling process is required to pass a pointer to the data structure containing the desired configuration information. This routine is identified by the following information in the HACB's command area:

Function = 4
parameter0 = not used
parameter1 = none
parameter2 = none

HAM_Queue_AEN_HACB (Function 5)

HAM_Queue_AEN_HACB directs the HAM to monitor asynchronous hardware events such as a bus or device reset. This function is identified by the following information in the HACB's command area

Function = 5
parameter0 = Bitmap indicating the type of events for which the CDM wants to be notified. Currently, the NWPA recognizes the following:

```
0x00000001   Bus reset
0x00000002   Device reset
0x00000004   Device attention
0x00000008   Adapter reset
0x00000010
   to Reserved
0x80000000
```

parameter1 = 0
parameter2 = 0

These requests must be issued on a per device basis, meaning that the CDM will provide the correct device handle for the device it wants monitored. The device handle is placed in the AEN HACB's **DeviceHandle** field.

The CDM builds the bitmap indicating the events it wants to be informed of, places the bitmap value in the parameter0 field of the AEN HACB, and executes the request by calling **CDI_Non_Blocking_Execute_HACB()**. This API requires the CDM to provide a pointer to a callback routine as an input parameter.

The HAM receives the AEN HACB through its *HAM_Queue_AEN_HACB* HAM function and maintains it in a local holding area associated with the target device until an event occurs. These AEN HACBs should not be placed in the device queue since they do not represent I/O requests that need device processing.

After an AEN event occurs, the HAM will check to see if the value in *parameter0* represents an event that a CDM wants to be notified of. If so, the HAM will freeze the device queue, set a bitmap value in the HACB's **Control_Info** field to indicate which event(s) occurred, place the AEN code (0x80080000) in the HACB's **hacbCompletion** field, and complete the AEN HACB by calling **HAI_Complete_HACB()**. The bit definitions for the return bitmap value are the same as those defined for the *parameter0* field.

Note: If no CDM has registered for a specific AEN event that occurs, the queue state will not change.

The HAM must be ready to accept multiples of these requests per device. When an event occurs, the HAM should complete all AEN HACBs registered for that event for the target device.

HAM_Tag_Queue_Synch/Asynch (Function 6)

HAM_Tag_Queue_Synch/Asynch directs the HAM to turn on either tag queuing or SCSI synchronous/asynchronous device negotiation. This function is identified by the following information in the HACB's command area:

Function = 6
parameter0 = Sub-function:
0x00000000 Tag Queuing

0x00000001 SCSI-II Synchronous/Asynchronous Negotiation

parameter1 = 0 Tag Queuing/SCSI-II Negotiation off (default)
1 Tag Queuing/SCSI-II Negotiation on

parameter2 = 0

These requests must be issued on a per device basis, meaning that the CDM will provide the correct device handle for the target device. The device handle is placed in the HACB's **DeviceHandle** field.

Device negotiation must adhere to that described in the *SCSI-II Standard, X3.131-199x*.

HAM_Recovery_Reset (Function 7)

HAM_Recovery_Reset directs the HAM to perform the reset specified function. This function is identified by the following information in the HACB's command area:

Function = 7
parameter0 = Sub-function:
0 = Adapter reset
1 = Bus reset
2 = Device reset
parameter1 = 0
parameter2 = 0

A CDM may issue this HAM function in an attempt to recover from a hung device. This function is called on a non-blocking thread, so the HAM should spawn a thread to perform the actual reset. In the case of an adapter or bus reset, the HAM should schedule this thread with about a half-second delay to let any current I/O on other devices have a chance to complete.

HAM_Deactivation_Notification(Function 8)

HAM_Deactivation_Notification allows the CDM to inform the HAM of a device it has deactivated. This function is identified by the following information in the HACB's command area:

Function = 8
parameter0 = 0
parameter1 = 0
parameter2 = 0