

Memory Allocators

An Integrated Interface for Allocating and Releasing Memory

Version 1.0.1

The contents of this document are strictly confidential and the document should be handled accordingly.

Table of Contents

1	Introduction	4
2	Initializing Memory Allocators	4
3	Memory Allocator Functions	5
4	Associating Unique Memory Management Libraries and Memory Allocators	5

Tables

Table 2-1 Functions to Initialize the Memory Allocators	4
Table 3-1 Functions for Allocating and Releasing Memory Blocks	5

Revision History

Version	Revision Date	Content of Revision
1.0.1	8/26/2005	Revised an incorrect function name NNS_FNDInitAllocatorForOSHeap → NNS_FndInitAllocatorForSDKHeap
1.0.0	01/066/2005	Revised the version number.
0.3.0	08/02/2004	Initial release.

1 Introduction

Some of the NITRO-System libraries allocate necessary memory within their own library. While such libraries require dynamic allocation of memory, any method of memory resource management can be used. Since it is never a good idea to depend on any one particular memory management library, the built-in memory allocation system was created and implemented.

The memory allocator is used in conjunction with a specific memory management library. The memory allocator provides only the functionality for allocating and releasing memory while the actual allocation and release operations are carried out by calling the memory management library functionality. The library receives the memory allocator, and through it, secures and releases memory. This process means the library is not dependent on a specific memory management library to function properly.

2 Initializing Memory Allocators

Before using a memory allocator, it must be initialized. The memory allocator holds the required information in a structure named `NNSFndAllocator`, and initializes by passing the pointer to this structure using the initialization function. There are four types of initialization functions: three support the NITRO System heaps and one supports the NITRO-SDK heap. The initialization functions are described below.

Table 2-1 Functions that Initialize the Memory Allocators

Function	Description
<code>NNS_FndInitAllocatorForExpHeap()</code>	Initializes the allocator so that memory can be allocated and released from the extended heap.
<code>NNS_FndInitAllocatorForFrmHeap()</code>	Initializes the allocator so that memory can be allocated and released from the frame heap.
<code>NNS_FndInitAllocatorForUnitHeap()</code>	Initializes the allocator so that memory can be allocated and released from the unit heap.
<code>NNS_FndInitAllocatorForSDKHeap()</code>	Initializes the allocator so that memory can be allocated and released from the NITRO-SDK heap.

Regardless of which initialization function is used, the corresponding heap must be created before the function is called.

3 Memory Allocator Functions

There are two memory allocator functions, one for allocating and the other for releasing memory. The actual behavior depends on the type of the memory management library associated with the function. The functions for allocating and releasing memory blocks are described in the table below.

Table 3-1 Functions for Allocating and Releasing Memory Blocks

Function	Description
<code>NNS_EndAllocFromAllocator()</code>	Allocates memory blocks from the allocator
<code>NNS_EndFreeToAllocator()</code>	Releases memory blocks from the allocator (returning them to the allocator)

4 Associating Unique Memory Management Libraries and Memory Allocators

Situations may arise where you want to use a unique memory management library with a memory allocator, or to change the behavior of a conventionally prepared memory allocator. You can do so by uniquely implementing the memory allocator. Refer to the memory allocator source for instructions on unique implementation.

© 2004-2005 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed or loaned in whole or in part without the prior approval of Nintendo Co. Ltd.