6.1. POOL_CREATE

Create a pool.

Synopsis

```
pool_create( name, addr, length, buff_size, options, pid )
```

Input Parameters

name	: string	user defined pool name
addr	: address	start address of pool
length	: integer	length of pool in bytes
buff_size	: integer	pool buffer size in bytes
options	: bit_field	pool create options

Output Parameters

pid	: pool_id	kernel defined pool identifier

Literal Values

options	+ GLOBAL	pool is global within the shared memory subsystem
	+ FORCED_DELETE	deletion will go ahead even if there are unrealeased buffers

Completion Status

OK	pool_create successful
ILLEGAL_USE	pool_create not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_BUFF_SIZE	buff_size not supported
INVALID_OPTIONS	invalid options value
TOO_MANY_OBJECTS	too many pools on the node or in the system
POOL_OVERLAP	area given overlaps an existing pool

Description

This operation declares an area of memory to be organized as a pool by the kernel. The process of formatting the memory to operate as a pool may require a memory overhead which may be taken from the new pool. It can never be assumed that all of the memory in the pool will be available for allocation. The overhead percentage will be implementation dependent.

The FORCED_DELETE option governs the deletion possibility of the pool (see 6.2 pool_delete).

6.2. POOL_DELETE

Delete a pool.

Synopsis

pool_delete(pid)

Input Parameters

pid	: pool_id	kernel defined pool identifier

Output Parameters

<none>

Completion Status

OK	pool_delete successful
ILLEGAL_USE	pool_delete not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	pool does not exist
OBJECT_DELETED	originally existing pool has been deleted before operation
POOL_IN_USE	buffers from this pool are still allocated
OBJECT_NOT_LOCAL	pool_delete not allowed on non-local pools

Description

Unless the FORCED_DELETE option was specified at creation, this operation first checks whether the pool has any buffers which have not been returned. If this is the case, then the POOL_IN_USE completion status is returned. If not, and in any case if FORCED_DELETE was specified, then the pool is deleted from the kernel data structure.

6.3. POOL_IDENT

Obtain the identifier of a pool on a given node with a given name.

Synopsis

```
pool_ident( name, nid, pid)
```

Input Parameters

name	: string	user defined pool name
nid	: node_id	node identifier

Output Parameters

pid	: pool_id	kernel defined pool identifier

Literal Values

nid	= LOCAL_NODE	the node containing the calling task
	= OTHER_NODES	all nodes in the system except the local node
	= ALL_NODES	all nodes in the system

Completion Status

OK	pool_ident successful
ILLEGAL_USE	pool_ident not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	node does not exist
NAME_NOT_FOUND	pool does not exist on node
NODE_NOT_REACHABLE	node is not reachable

Description

This operation searches the kernel data structure in the node(s) specified for a pool with the given name, and returns its identifier if found. If OTHER_NODES or ALL_NODES is specified, the node search order is implementation dependent. If there is more than one pool with the same name, then the pid of the first one found is passed back.

Observation:

This operation may return the pid of a GLOBAL pool that is not in the same shared memory subsystem as the node containing the calling task.

6.4. POOL_GET_BUFF

Get a buffer from a pool.

Synopsis

```
pool_get_buff( pid, buff_addr )
```

Input Parameters

pid	: pool_id	kernel defined pool identifier

Output Parameters

buff_addr	: address	address of obtained buffer

Completion Status

OK	pool_get_buff successful
ILLEGAL_USE	pool_get_buff not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	pool does not exist
OBJECT_DELETED	originally existing task has been deleted before operation
NO_MORE_MEMORY	no more buffers available in pool
POOL_NOT_SHARED	pool not in shared memory subsystem
NODE_NOT_REACHABLE	node on which pool resides is not reachable

Description

The pool_get_buff requests for a single buffer from the pool's free memory. If the kernel cannot immediately fulfil the request, it returns the completion status NO_MORE_MEMORY, otherwise the address of the allocated buffer is returned. The exact allocation algorithm is implementation dependent.

6.5. POOL_RET_BUFF

Return a buffer to its pool.

Synopsis

```
pool_ret_buff( pid, buff_addr )
```

Input Parameters

pid	: pool_id	kernel defined pool identifier
buff_addr	: address	address of buffer to be returned

Output Parameters

<none>

Completion Status

OK	pool_ret_buff successful
ILLEGAL_USE	pool_ret_buff not callable from ISR
INVALID_PARAMETER	a parameter refers to an invalid address
INVALID_ID	pool does not exist
OBJECT_DELETED	originally existing pool has been deleted before operation
POOL_NOT_SHARED	pool not in shared memory sybsystem
INVALID_BUFF	no buffer allocated from pool at buff_addr
NODE_NOT_REACHABLE	node on which pool resides is not reachable

Description

This operation returns the given buffer to the given pool's free space. The kernel checks that the buffer was previously allocated from the pool and returns INVALID_BUFF if it wasn't.