NAME

scsi_init_pkt – prepare a complete SCSI packet

SYNOPSIS

#include <sys/scsi/scsi.h>

struct scsi_pkt *scsi_init_pkt(struct scsi_address **ap*, **struct scsi_pkt** **pktp*,
struct buf **bp*, **int** *cmdlen*, **int** *statuslen*, **int** *privatelen*, **int** *flags*,
int (**callback* **)(caddr_t), caddr_t** *arg***);**

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

ARGUMENTS

ap	Pointer to a **scsi_address**(9S) structure.
pktp	A pointer to a **scsi_pkt**(9S) structure.
bp	Pointer to a **buf**(9S) structure.
cmdlen	The required length for the SCSI command descriptor block (CDB) in bytes.
statuslen	The required length for the SCSI status completion block (SCB) in bytes.
privatelen	The required length for the *pkt_private* area.
flags	Flags modifier.
callback	A pointer to a callback function, **NULL_FUNC**, or **SLEEP_FUNC**.
arg	The *callback* function argument.

DESCRIPTION

Target drivers use **scsi_init_pkt( )** to request the transport layer to allocate and initialize a packet for a SCSI command which possibly includes a data transfer. If *pktp* is **NULL,** a new **scsi_pkt**(9S) is allocated using the HBA driver's packet allocator. The *bp* is a pointer to a **buf**(9S) structure. If *bp* is non-**NULL** and contains a valid byte count, the **buf**(9S) structure is also set up for DMA transfer using the HBA driver DMA resources allocator. When *bp* is allocated by **scsi_alloc_consistent_buf**(9F), the **PKT_CONSISTENT** bit must be set in the *flags* argument to ensure proper operation. If *privatelen* is non-zero then additional space is allocated for the *pkt_private* area of the **scsi_pkt**(9S). On return *pkt_private* points to this additional space. Otherwise *pkt_private* is a pointer that is typically used to store the *bp* during execution of the command. In this case *pkt_private* is **NULL** on return.

The *flags* argument is a set of bit flags. Possible bits include:

PKT_CONSISTENT This must be set if the DMA buffer was allocated using **scsi_alloc_consistent_buf**(9F). In this case, the HBA driver will guarantee that the data transfer is properly synchronized before performing the target driver's command completion callback.

PKT_DMA_PARTIAL This may be set if the driver can accept a partial DMA mapping. If set, **scsi_init_pkt( )** will allocate DMA resources with the **DDI_DMA_PARTIAL** bit set in the **dmar_flag** element of the **ddi_dma_req**(9S) structure. The **pkt_resid** field of the **scsi_pkt**(9S) structure may be returned with a non-zero value, which indicates the number of bytes for which **scsi_init_pkt( )** was unable to allocate DMA resources. In this case, a subsequent call to **scsi_init_pkt( )** may be made for the same *pktp* and *bp* to adjust the DMA resources to the next portion of the transfer. This sequence should be repeated until the **pkt_resid** field is returned with a zero value, which indicates that with transport of this final portion the entire original request will have been satisfied.

When calling **scsi_init_pkt( )** to move already-allocated DMA resources, the *cmdlen*, *statuslen* and *privatelen* fields are ignored.

The last argument *arg* is supplied to the *callback* function when it is invoked.

callback indicates what the allocator routines should do when resources are not available:

NULL_FUNC Do not wait for resources. Return a **NULL** pointer.

SLEEP_FUNC Wait indefinitely for resources.

Other Values *callback* points to a function which is called when resources may have become available. *callback* **must** return either **0** (indicating that it attempted to allocate resources but again failed to do so), in which case it is put back on a list to be called again later, or **1** indicating either success in allocating resources or indicating that it no longer cares for a retry.

When allocating DMA resources, **scsi_init_pkt( )** returns the **scsi_pkt** field **pkt_resid** as the number of residual bytes for which the system was unable to allocate DMA resources. A **pkt_resid** of **0** means that all necessary DMA resources were allocated.

RETURN VALUES

scsi_init_pkt( ) returns **NULL** if the packet or dma resources could not be allocated. Otherwise, it returns a pointer to an initialized **scsi_pkt**(9S). If *pktp* was not **NULL** the return value will be *pktp* on successful initialization of the packet.

CONTEXT

If *callback* is **SLEEP_FUNC**, then this routine may only be called from user-level code. Otherwise, it may be called from either user or interrupt level. The *callback* function may not block or call routines that block.

EXAMPLES

To allocate a packet without DMA resources attached, use:

```
pkt = scsi_init_pkt(&devp->sd_address, NULL, NULL, CDB_GROUP1,
        STATUS_LEN, sizeof (struct my_pkt_private *), 0,
        sd_runout, sd_unit);
```

To allocate a packet with DMA resources attached use:

```
pkt = scsi_init_pkt(&devp->sd_address, NULL, bp, CDB_GROUP1,
        STATUS_LEN, 0, 0, NULL_FUNC, NULL);
```

To attach DMA resources to a preallocated packet, use:

```
pkt = scsi_init_pkt(&devp->sd_address, old_pkt, bp, 0,
        0, 0, 0, sd_runout, (caddr_t) sd_unit);
```

Since the packet is already allocated the *cmdlen*, *statuslen* and *privatelen* are **0**.

To allocate a packet with consistent DMA resources attached, use:

```
bp = scsi_alloc_consistent_buf(&devp->sd_address, NULL,
        SENSE_LENGTH, B_READ, SLEEP_FUNC, NULL);
pkt = scsi_init_pkt(&devp->sd_address, NULL, bp, CDB_GROUP0,
        STATUS_LEN, sizeof (struct my_pkt_private *), PKT_CONSISTENT,
        SLEEP_FUNC, NULL);
```

To allocate a packet with partial DMA resources attached, use:

```
my_pkt = scsi_init_pkt(&devp->sd_address, NULL, bp, CDB_GROUP0,
        STATUS_LEN, sizeof (struct buf *), PKT_DMA_PARTIAL,
        SLEEP_FUNC, NULL);
```

SEE ALSO

scsi_alloc_consistent_buf(9F), **scsi_destroy_pkt**(9F), **scsi_dmaget**(9F), **scsi_pktalloc**(9F), **buf**(9S), **ddi_dma_req**(9S), **scsi_address**(9S), **scsi_pkt**(9S)

Writing Device Drivers

NOTES

If a DMA allocation request fails with **DDI_DMA_NOMAPPING**, the **B_ERROR** flag will be set in *bp*, and the **b_error** field will be set to **EFAULT**.

If a DMA allocation request fails with **DDI_DMA_TOOBIG**, the **B_ERROR** flag will be set in *bp,* and the **b_error** field will be set to **EINVAL**.