Printer Working Group – 1394 Printing

PWG 1394 Transport Command Set Proposal Revision 0c

Gregory A. Shue, Hewlett-Packard

June 18, 1998

1 Scope

This document contains a proposal for an SBP-2 command set sufficient to meet the transport stack communication requirements outlined by the Printer Working Group – 1394 Printing sub-committee. This document is limited to information needed to fully describe the SBP-2 ORB and Status Block data structures, the Commands, Parameters, and policies necessary for providing the required transport functionality. This does not cover Configuration ROM information.

2 Device Model

The general device model is covered in the actual PWG document. The following information to follow is for context only. The PWG SBP-2 Transport Command Set is based on the SBP-2 protocol. Rather than using a single ordered queue maintained in shared memory space and using the ORDERED processing model as defined by SBP-2, this transport models the SBP-2 device has having two ordered execution queues maintained within the target. The initiator uses the Task List to queue tasks on the end of the internal ordered execution queues. In addition, the target indicates that it uses the UNORDERED execution model of SBP-2. Because the target maintains the queues internally, target resources limit the size of the task set which it can manage. This limit is available to the initiator through command-set dependent means.

In order to provide bi-directional data flow, queue 0 shall not support initiator-to-target data transfers, and queue 1 shall only support initiator-to-target data transfers.

TO BE DETERMINED – Should the usage of each queue be specified or left arbitrary?

2.1 Transaction error recovery

SBP-2 recommends how initiators and targets should recover from a few of the more common error scenarios. This specification requires the behaviors described in the appropriate section of SBP-2.

2.1.1 Status FIFO write request recovery

SPB-2 identifies that a target should take no error recovery actions when it detects a missing acknowledgement after a write to an initiator's status FIFO. It also states that an initiator is expected to discover this error by a higher-level mechanism and to initiate appropriate error recovery actions, though the details are beyond the scope of SPB-2.

When a target detects a missing acknowledgement after a write to an initiator's status FIFO, it does not know which of the following applies:

- The information was correctly received in the status FIFO (and the initiator knows that the command has been completed); or

- The information had a CRC or other error, has not been correctly received by the initiator and the command is still uncompleted.

To recover from this condition, the initiator must determine when to take error recovery actions, and the target must be able to tolerate executed, but uncompleted non-idempotent commands. (Idempotent commands have indistinguishable effect whether they are executed once or many times in succession.)

NOTE – Whatever mechanism is defined needs to be broad enough to cover standard commands, vendor-specific commands, and unsolicited status notifications.

When this condition occurs, the target shall proceed as follows:

a) It shall stop execution of all commands.

b) It shall remember the specific command and queue which had the error.

c) It shall release all resources allocated by the target which are associated with the ORB.

d) It shall initiate a Bus Reset event. This shall cause all the tasks in the target to be implicitly aborted and causes the initiator to requeue the commands following a successful reconnection.

NOTE – For this device model, this case is assumed to be extremely infrequent, so error recovery actions with a high impact on the bus environment are assumed to be tolerable.

Upon completion of a successful reconnection, the initiator must restore FETCH DATA commands and associated data to the task set in the order previously queued. The commands must be requeued to the same order as before, and the data buffers must retain the same content as before. No assumptions shall be made about ORB or buffer location remaining the same within the 1394 shared memory space, or about the same physical buffers being used.

In order for the target to distinguish an executed but uncompleted command from an identical one preceding it or succeeding it in the data stream, a sequence number shall be used in combination with the queue identifier.

TO BE DETERMINED – What is the scope of the sequence number? If it is global to the queue, then it's range must be larger than the maximum task set size. If it is queue specific, then it only needs to support three (3) values, but the issue comes in of whether to allow after a reconnect the insertion of either vendor-specific commands or parameter interaction commands into the front of the data stream. If it is command-specific, then the issue of mixing vendor-specific commands into the data stream needs to be examined.

Following a reconnection, the target shall determine whether the command and sequence number of the first initiator-to-target data transfer command and the previous initiator-to-target data transfer command match. In response, the target shall take one of the following actions.

- If no match was found and the previous command was executed completely, the target shall execute the new command.
- If no match was found and the previous command was not executed completely, the target shall report the protocol violation and transition to the dead state.
- If a match was found and the previous command was executed completely, the target shall report the protocol violation and transition to the dead state.
- If a match was found and the previous command was executed but the acknowledgement was lost on the write to the status FIFO, the target shall not execute the command but shall write completion notification to the status FIFO as if the command had been executed.

3 Data structures

There are three data structures described by this standard:

- SBP-2 Operation request blocks (ORBs);
- SBP-2 Status blocks;
- Command parameter encoding.

3.1 Operation request blocks (ORBs)

All ORB formats described in SBP-2 shall be supported. In addition to the management functions required by SBP-2, the target shall support the management function LOGICAL UNIT RESET.

TO BE DETERMINED – SBP-2 requires that for the management functions LOGICAL UNIT RESET and TARGET RESET, the target shall create a unit attention condition for all other logged-in initiators at that scope. This requires that a unit attention status model be defined. The unit attention condition is required to use the completion notification/unsolicited status mechanism defined in SBP-2.

3.1.1 Command block ORB

Command block ORBs are used to encapsulate data transfer or device control commands for transport to the target.

The general format of the command block ORB is illustrated by the following figure.

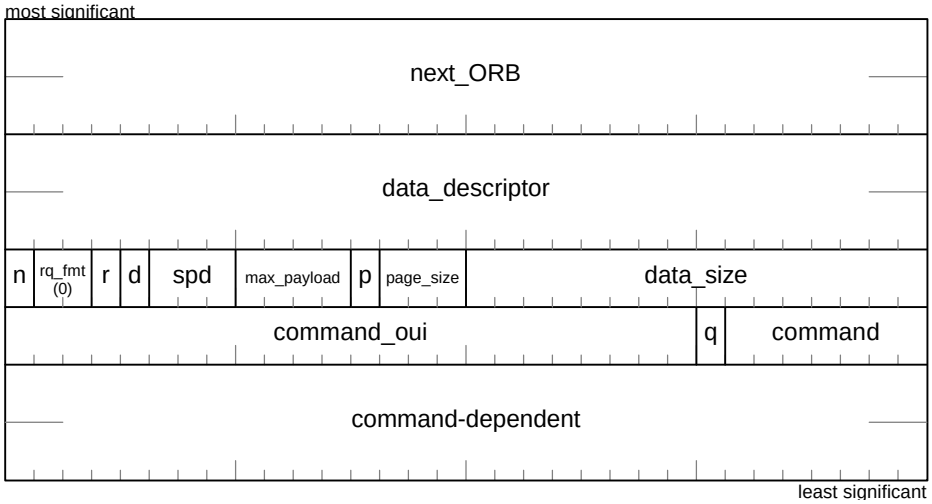

Figure 1 – Command block ORB

The *next_ORB*, *data_descriptor*, *rq_fmt*, *spd*, *max_payload*, *page_size* and *data_size* fields and the *notify*, *direction* and *page_table_present* bits (abbreviated as *n*, *d* and *p*, respectively, in the figure above) shall be as specified by SBP-2.

The *notify* bit shall contain a value of one (1).

TO BE DETERMINED – Since the device is modeled as having two ordered execution queues, whether or not the notify bit is set to one (1) for every command may be determined by the lost acknowledgement error recovery mechanism.

The *command_oui* field shall contain an Organizationally Unique Identifier (OUI) obtained from the IEEE Registration Authority Committee (RAC). The organization or company identified by *command_oui* specifies the interpretation of the *command* field. Unique identifiers for a company or organization may be obtained from:

Institute of Electrical and Electronic Engineers, Inc.
Registration Authority Committee
445 Hoes Lane
Piscataway, NJ 08855-1331

The *queue_ID* bit (abbreviated as *q* in the figure above) shall identify the ordered execution queue to which the ORB belongs. See X for more information on the target's management of the ordered execution queues.

The *command* field shall specify the command to be executed by the target. When *command_oui* has a value of XX $XXXX_{16}$, the *command* field shall be interpreted according to the table below.

Value	Command	Description
0	FETCH DATA	The target reads data from the buffer
1	STORE DATA	The target writes data to the buffer
2	GET PARAMETER	The target stores parameter information into the buffer
3	SET PARAMETER LIST	The target updates its current parameters according to information fetched from the buffer

The *command*-dependent field shall contain information determined by the value of *command*.

The precedence of error reporting shall be as follows:

1. SBP-2 errors
2. *notify* bit not set

   TO BE DETERMINED – According to SBP-2, the target may send a completion notification whether or not the *notify* bit is set. As a result, this document really needs to specify when a target shall send a completion notification. To avoid deadlock from holding on to memory resources, a target may end up having to send one on every STORE DATA command.

3. Data length not supported by protocol (>= 2^{31} bytes)
4. *command_oui* not supported
5. *command* not supported
6. Command-specific errors

3.1.2 Fetch Data Command ORB

The initiator shall use this command to request the target to read data from the buffer using the ORB format shown below.

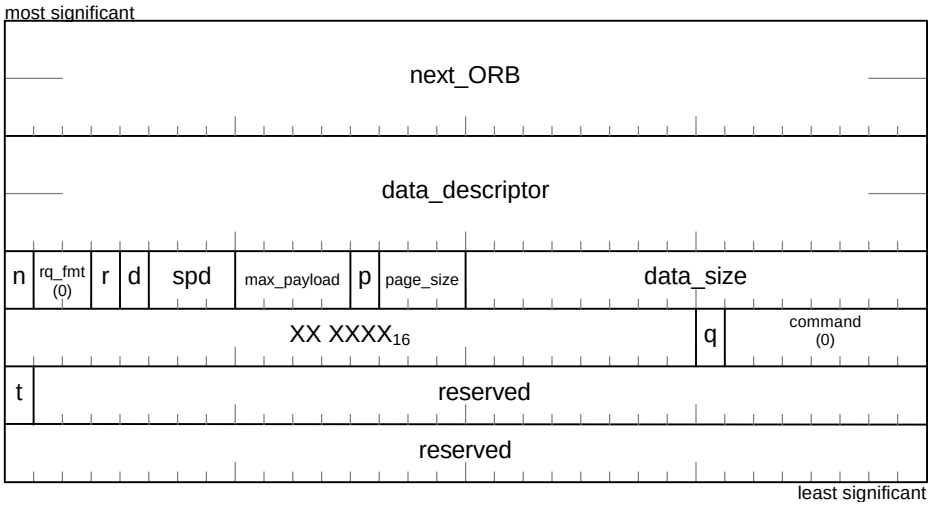

Figure 2 – Fetch Data Command ORB

XX $XXXX_{16}$ is the OUI obtained by XXXX from the IEEE/RAC. The value indicates that the meaning of this *command* field value is as defined in this standard.

Data lengths of zero (0) bytes are valid and shall complete without data being transferred.

If the command fails, the target shall retain no data referenced by this command.

The *tag* bit (abbreviated *t* in the figure above) shall specify whether or not this data is distinct from the rest of the data stream.

There are no command-specific errors specified.

3.1.2.1 Fetch Data Command Response

The command response shall conform to the format described in 3.2 Status block.

The *sfmt* field shall contain a value of zero (0).

The *status* field shall only be valid provided no SBP-2 errors are reported. When valid, this field shall indicate GOOD (0) upon successful completion of the command. If errors are detected during processing, this field shall be updated to reflect the error as specified in this standard.

The *tag* bit in the command response is reserved for this command.

The *residual* field is reserved for this command. Upon successful completion of the command, the target has fetched all the data.

3.1.3 Store Data Command ORB

The initiator shall use this command to request the target to store data to the buffer using the ORB format shown below. The command shall complete when either the requested amount of data becomes available, or the target reaches a transition between tagged and untagged data within the data stream. When the transport is being used for message-style communication, the target shall complete the command when a message boundary is reached. The target may complete the command prior to those events.

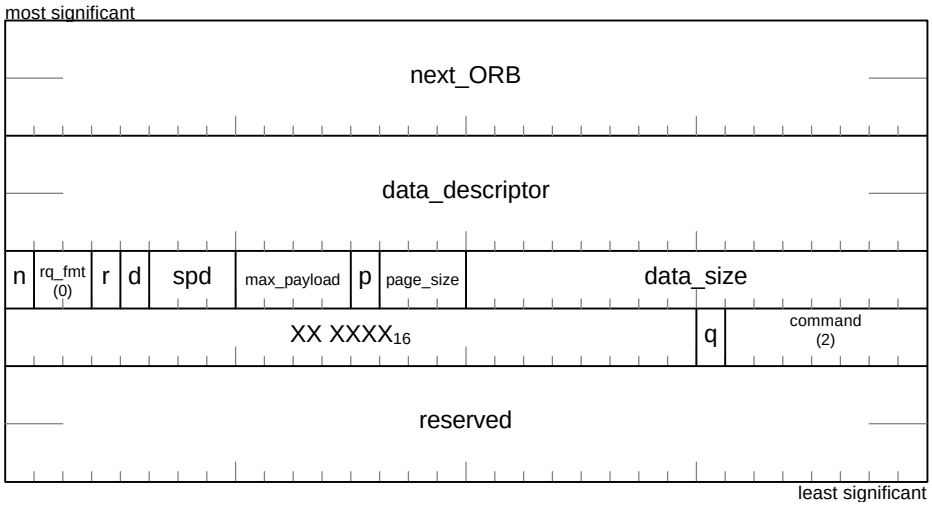

Figure 3 – Store Data Command ORB

XX XXXX$_{16}$ is the OUI obtained by xxxx from the IEEE/RAC. The value indicates that the meaning of this *command* field value is as defined in this standard.

A *data_size* value of zero (0) bytes is valid. In this case, the command shall not complete until some data becomes available for the initiator.

If the command fails, the target shall retain all pending data.

There are no command-specific errors specified.

3.1.3.1 Store Data Command Response

The command response shall conform to the format described in 3.2 Status block.

The *sfmt* field shall contain a value of zero (0).

The *status* field shall only be valid provided no SBP-2 errors are reported. When valid, this field shall indicate GOOD (0) upon successful completion of the command. If errors are detected during processing, this field shall be updated to reflect the error as specified in this standard.

The *tag* bit shall specify whether or not this data is distinct from the rest of the data stream. If no data is able to be returned due to a zero (0) length data block, this field is reserved.

The *residual* field shall conform to the description given in 3.2 Status block. For message-based communication, the residual field shall indicate either the amount of residual buffer space provided by

the command, or the additional amount of data within this message which could not be sent (using twos-complement notation). For stream-based communication, the initiator is required to an outstanding Store Data Command. In this case the *residual* field shall not indicate the presence of additional data for the initiator. It shall only indicate the amount of unused data space provided by the *data_descriptor*.

TO BE DETERMINED – When else should a target complete this command? (e.g. “Flush” requested from target’s transport client; the target can no longer retain the data for retransmission?)

3.1.4 Get Parameter Command ORB

The initiator shall use this command to request the target to store parameter information into the buffer. The initiator shall the ORB format shown below.

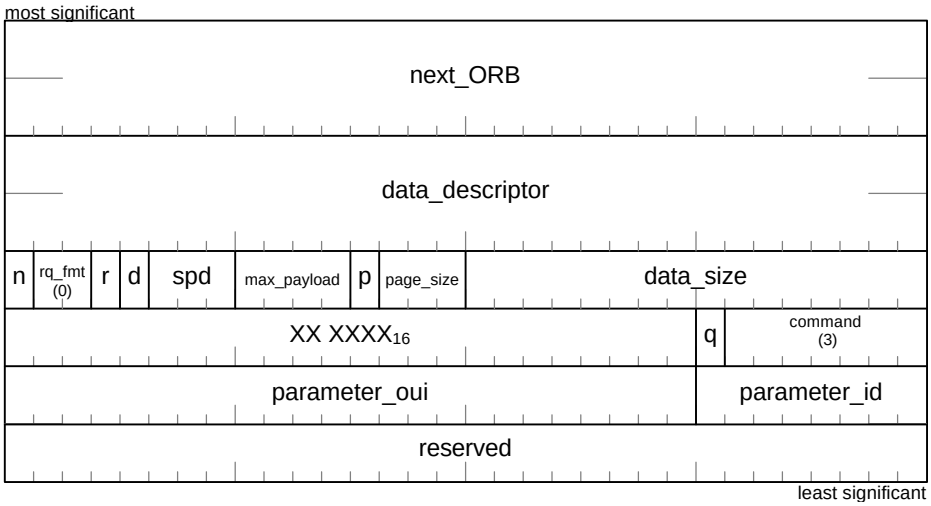

Figure 4 – Get Parameter Command ORB

XX $XXXX_{16}$ is the OUI obtained by xxxx from the IEEE/RAC. The value indicates that the meaning of this *command* field value is as defined in this standard.

The *parameter_oui* field shall contain an Organizationally Unique Identifier (OUI) obtained from the IEEE Registration Authority Committee (RAC). The organization or company identified by *parameter_oui* specifies the interpretation of the *parameter_id* field. This additional level of qualification allows vendor-specific parameters to be accessed with the standard commands.

The *parameter_id* field shall specify the parameter to be accessed by the target.

The *data_descriptor* field shall reference a memory block to be used to return the parameter information.

If an unsupported *parameter_oui* value is specified, or an unsupported *parameter_id* value is specified, the target shall return an empty parameter encoding. The parameter information shall be encoded as described in 3.3 Parameter Encoding.

There are no command-specific errors specified.

3.1.4.1 Get Parameter Command Response

The command response shall conform to the format described in 3.2 Status block.

The *sfmt* field shall contain a value of zero (0).

The *status* field shall only be valid provided no SBP-2 errors are reported. When valid, this field shall indicate GOOD (0) upon successful completion of the command.

The *residual* field shall conform to the description given in 3.2 Status block. If insufficient data buffer space is provided to store the entire data generated by the command, the data block shall be completely

written with valid information, and the residual field shall contain (as a negative number) the amount of data which could not be stored.

NOTE – Though the initiator can not use an incomplete encoding, the target must provide at least the first two quadlets of the incomplete encoding, if possible. This guarantees that the initiator will not interpret invalid (previous) data as a new encoding. In order for the residual calculation to be consistently applied, the target should transfer all partial data.

The buffer shall be filled with parameter information encoded as described in 3.3 Parameter Encoding.

3.1.5 Set Parameter List Command ORB

The initiator shall use this command to request the target to update the indicated parameters to the values in the buffer. The initiator shall use the ORB format shown below.

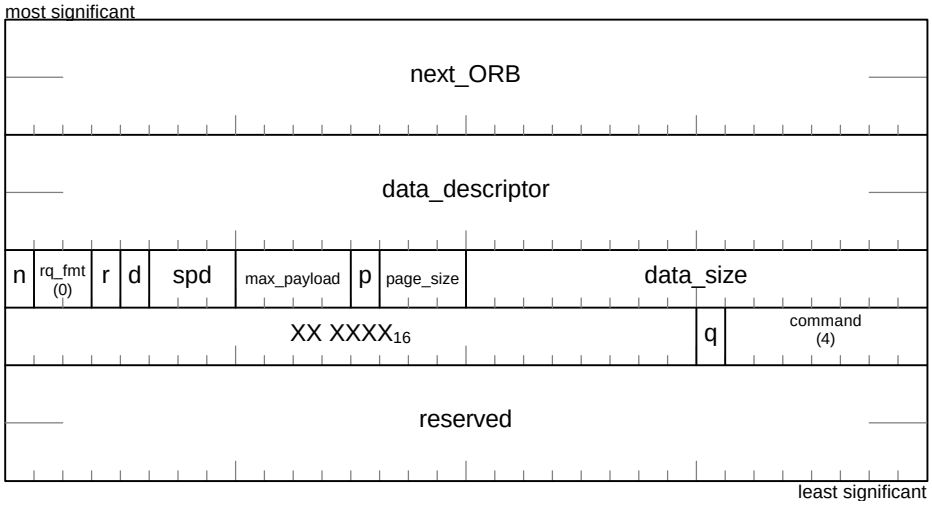

Figure 5 – Set Parameter List Command ORB

XX XXXX$_{16}$ is the OUI obtained by xxxx from the IEEE/RAC. The value indicates that the meaning of this *command* field value is as defined in this standard.

The *data_descriptor* field shall reference a memory block containing encoding(s), as described in 3.3 Parameter Encoding, for the parameter(s) to be set.

There are no command-specific errors specified. All behaviors regarding parameter encodings are described in 3.3 Parameter Encoding.

3.1.5.1 Set Parameter List Command Response

The command response shall conform to the format described in 3.2 Status block.

The *sfmt* field shall contain a value of zero (0).

The *status* field shall only be valid provided no SBP-2 errors are reported. When valid, this field shall indicate GOOD (0) upon successful completion of the command.

The *residual* field is reserved for this command.

3.2 Status block

A target shall store status at an initiator *status_FIFO* address when a request completes (successfully or in error). The *status_FIFO* address is obtained implicitly from the fetch agent context. Whenever the target has status to report, it shall store all or part of the status block shown below.

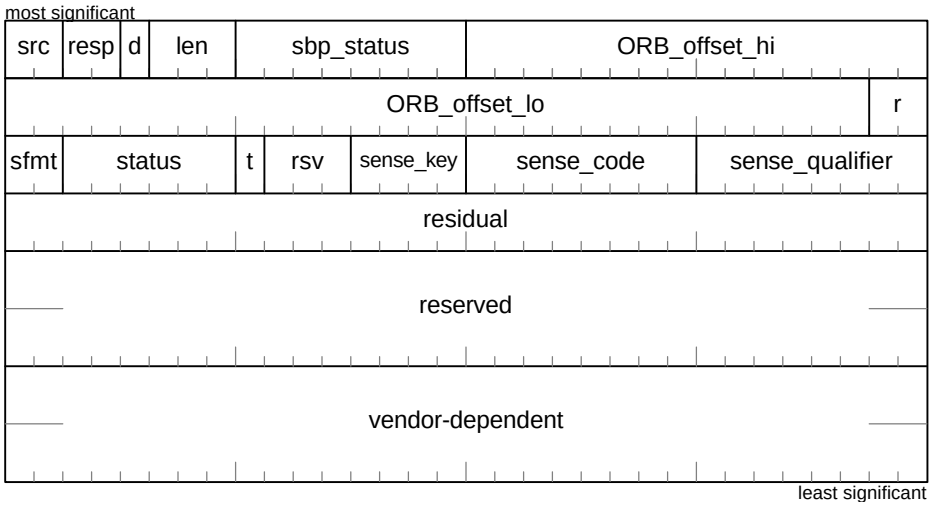

Figure 6 – Status block format

The target shall store a minimum of eight bytes of status information and may store up to the entire 32 bytes defined above so long as the amount of data stored is an integral number of quadlets. A truncated status block shall be interpreted as if the omitted fields had been stored as zeros. The target shall use a single Serial Bus block write transaction to store the status block at the *status_FIFO* address.

The *src*, *resp*, *len*, *sbp_status*, *ORB_offset_hi*, and *ORB_offset_lo* fields and the *dead* bit (abbreviated as *d* in the figure above) shall be as specified by SBP-2.

The *sfmt* field shall specify the format of the status block. The table below defines permissible values for *sfmt*.

Value	Description
0	Current error; status block format defined by this standard
1	Reserved for future standardization
2	Reserved for future standardization
3	Status block format vendor-dependent

The *status* field shall contain status information as defined by this standard. Status shall be sent from the target to the initiator when a command ends with a service response of TASK COMPLETE. The receipt of any status shall indicate that the associated task has ended. The following table defines permissible values for *status*.

Value	Description
0	GOOD
1	Notify Bit not set
2	Data length not supported
3	Command_oui not supported
4	Command not supported
5	MAX DATA SIZE PER FETCH DATA ORB exceeded
$3F_{16}$	Unspecified error
All other values	Reserved for future standardization

Definitions for each status code are given below:

GOOD. This status indicates that the target has successfully completed the task.

NOTIFY BIT NOT SET. This status indicates that the target has detected that the *notify* bit is not set to one as required by the protocol.

DATA LENGTH NOT SUPPORTED. This status indicates that the target has detected an unsupported data length.

COMMAND OUI NOT SUPPORTED. This status indicates that the target does not recognize the *command_oui* value within the ORB.

COMMAND NOT SUPPORTED. This status indicates that the target does not recognize the *command* value within the ORB.

MAX DATA SIZE PER FETCH DATA ORB EXCEEDED. This status indicates that the initiator requested more data to be transferred to the target than is supported by the target.

UNSPECIFIED ERROR. This status indicates that the target has detected an error condition not specified in this standard. (e.g. an internal error condition which prevents the target from continuing without a power cycle.)

The *tag* bit (abbreviated as *t* in the figure above) shall specify the association of the data returned in the data buffer (out-of-band). This field is reserved for FETCH DATA, GET PARAMETER, and SET PARAMETER LIST commands. When the *tag* bit is zero, the contents of the data buffer are part of the normal data stream. When the *tag* bit is one, the contents of the data buffer shall be understood to be distinct from but synchronous with the normal data stream.

The *sense_key, sense_code,* and *sense_qualifier* fields are reserved.

The contents of the *residual* field are unspecified if the *sfmt* field has a value of three. For *sfmt* values of one or two, the *residual* field is reserved. If the *sfmt* field has a value of zero, this contains the residue of the requested data transfer length minus the length of actual data to be transferred, in bytes, of the command. Negative values are indicated in two's complement notation.

The contents of the *vendor-dependent* field are command-dependent and are defined within the appropriate standard for the command. For *sfmt* field values of zero, one, and two, the field is reserved.

3.3 Parameter Encoding

All parameters passed by reference (whether to set or return values) shall be encoded using the format that follows. Parameter encodings may be packed together to form a list, which may be passed by reference, when supported by the command.

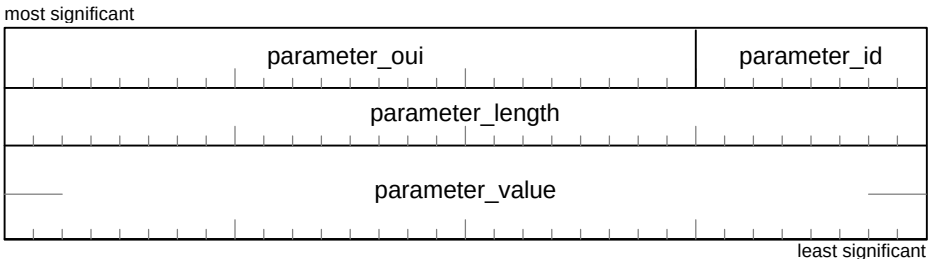

Figure 7 – Parameter ID and value format

The *parameter_oui* value shall indicate which organization has specified the meaning of the associated *parameter_id* value.

The *parameter_id* value shall indicate what information is encoded in the *parameter_value* field.

The *parameter_length* field shall indicate the number of bytes of valid data in the *parameter_value* field. A *parameter_length* value of zero on a GET PARAMETER command shall indicate an unsupported parameter. A *parameter_length* of zero on a SET PARAMETER command shall cause the parameter encoding to be discarded. All *parameter_length* values >= 2^{31} are invalid, shall cause the parameter encoding to be discarded.

The *parameter_value* field contains the actual data for the identified parameter. The data may be of an arbitrary byte length greater than zero.

If the parameter is of a fixed size, any data not encoded shall be interpreted as the bits being zero. The data transferred shall represent the least-significant bytes of data.

If the parameter is of a variable size, the target shall use the last n bytes of data as the value, where n is the maximum byte size supported by the target for that parameter. The target shall give no indication of data overflow to the initiator.

If the *parameter_value* field does not terminate on a quadlet boundary, the encoding shall be padded to the next quadlet boundary with reserved bytes. The *parameter_length* field shall not be adjusted to include this padding.

3.4 Standard Parameters

When *parameter_oui* has a value of xx $xxxx_{16}$, the *parameter_id* field shall be interpreted according to the table below. All unspecified values are reserved.

Value	Parameter	Access	Size	Description
0	SUPPORTED PARAMETERS LIST	RO	$< 2^{31}$ bytes	Self-describing list of all other supported parameters for the associated *parameter_oui* and their current values. Parameters shall be returned in numerically ascending *parameter_id* order.
1	MAX TASK SET SIZE	RO	16 bits	Returns the maximum number of pending commands across both command queues.
2	MAX DATA SIZE PER FETCH ORB	RO	31 bits	The maximum data size supported for FETCH DATA commands.
3	MAX DATA SIZE PER STORE ORB	RO	31 bits	(Optional) The maximum data size generated within STORE DATA commands.
4	COMMUNICATION MODE	RW	2 bits	Configures for bi-directional or send- or receive-only unidirectional communication.
FE_{16}	UNIFIED PARAMETER_OUI SET	RO	$< 2^{31}$ bytes	Set of all *parameter_oui* values with a supported parameter by this target. *parameter_oui* values shall be returned in numerically ascending order.

3.4.1 SUPPORTED PARAMETERS LIST (Required)

This parameter returns all the supported parameters associated with this OID. Each parameter is encoded as specified. The parameter list, excluding this parameter, is returned with parameters in ascending *parameter_id* order. If insufficient memory is provided to return the complete list, then the entire memory shall be filled and the additional space required shall be indicated in the *residual* field. This value shall be fixed for the duration of a Login.

3.4.2 MAX TASK SET SIZE (Required)

This parameter returns the maximum size of the active task set across all ordered queues supported by this connection. The initiator must avoid queuing more tasks than this on the Task List. All targets shall support at least one active task for each ordered queue. This value shall be fixed for the duration of a Login.

3.4.3 MAX DATA SIZE PER FETCH ORB (Required)

This parameter indicates the maximum amount of data that may be provided to the target in a FETCH DATA ORB. A value of zero (0) for this parameter indicates the target will not generate data for the initiator. This value shall be fixed for the duration of a Login.

NOTE – This size is limited by the target's ability to discard data within an active FETCH DATA command after being interrupted by a Bus Reset. Data sizes larger than this would prevent the target from

discarding all the data referenced by this ORB in the event that this task was aborted either explicitly or implicitly.

NOTE – Because the initiator is required to maintain in-order data flow, the integrity of the data stream must be maintained across command failures and Bus Reset events. For stream services, the target could maintain a fetch offset into the data stream referenced from the last released data. This would be valid no matter how the data stream was repackaged among an ORB sequence following a Bus Reset. Thus, this parameter is not necessary for stream services provided the requirement of having to 'discard all data associated with a failed command' can be relaxed.

NOTE – For message services, the parameter is not needed (because both transport clients using the service know the constraints on the message sizes), but the requirement to not retain (or process) any of the data contents must be maintained.

TO BE DETERMINED – Will this parameter be removed?

3.4.4 MAX DATA SIZE PER STORE ORB (Optional)

This optional parameter indicates the maximum amount of data that will be provided by the target in response to a Store Data command. This allows the initiator to adjust its memory usage to better match the target's capabilities. A value of zero (0) for this parameter indicates the target will not generate data for the initiator. This value shall be fixed for the duration of a Login.

NOTE – This size is limited by the target's ability to retransmit data within an active Store Data command after being interrupted by a Bus Reset.

TO BE DETERMINED – Because this parameter is not required for operation, it needs to be determined whether it shall be removed, left optional, or become required as part of the standard.

3.4.5 COMMUNICATION MODE (Required)

This parameter indicates the current communications mode of the connection. Upon Login, the connection is enabled for maximum supported functionality. At any point during the connection, the initiator may disable data transfers in either or both directions. Once a direction has been disabled, it may not be re-enabled. GET PARAMETER, SET PARAMETER, and vendor-dependent commands are always supported.

This parameter may have the following values as described in the table that follows. All other values for this parameter are reserved.

Value	Description
0	This connection is disabled for both initiator-to-target and target-to-initiator data transfers.
1	This connection is disabled for target-to-initiator data transfers.
2	This connection is disabled for initiator-to-target data transfers.
3	This connection is enabled for both initiator-to-target data and target-to-initiator data transfers.

TO BE DETERMINED – Exactly what should happen to FETCH DATA and STORE DATA commands which are in or added to the active task set when a data flow direction is disabled? What about vendor-specific commands?

Upon successful login, this parameter shall be in agreement with the sizes specified in the MAX DATA SIZE PER FETCH ORB and MAX DATA SIZE PER STORE ORB parameters.

3.4.6 UNIFIED PARAMETER_OUI SET (Required)

This parameter returns the unified set of *parameter_oui* values from all the supported vendor-dependent parameters. This list shall not contain the standard *parameter_oui* value of xx $xxxx_{16}$. The *parameter_oui* values shall be encoded as follows.

Figure 8 – Parameter_oui encoding format

One quadlet shall be used for each *parameter_oui* encoding. The byte length of the set of encodings shall be returned as the length of the parameter. The *parameter_oui* values shall be returned in numerically ascending order.

3.5 Vendor-Specific Parameters

If a target supports vendor-specific parameters, for each supported *parameter_oui* the target shall support a *parameter_id* value of zero. This value shall reference the SUPPORTED PARAMETERS LIST for the associated *parameter_oui* as described in 3.4 SUPPORTED PARAMETERS LIST.