1394 PRINTER WORKING GROUP

MODEL & COMMAND SET

*****PRELIMINARY DRAFT PROPOSAL *****

Revision 0.45 - January 15, 1999

Editor - Alan Berkema
8000 Foothills Blvd. MS #5558
Roseville Ca, 95746
916 785-5605
Fax 916 785-1195

1. Contents

Page

1. Contents 2

2. Figure 2

3. PWG Function/Service Model 3
   3.1. Initiator To Target Communication 5
   3.2. Target To Initiator Communication 6
   3.3. Data Transfer 7

4. Command Set 8
   4.1. Commands..8
   4.2. Actions 9
      4.2.1. I2T_Connect 9
      4.2.2. I2T_Connect_Reply 10
      4.2.3. I2T_Disconnect 10
      4.2.4. I2T_Disconnect_Reply 10
      4.2.5. T2I_Connect 10
      4.2.6. T2I_Connect_Reply 10
      4.2.7. T2I_Disconnect 10
      4.2.8. T2I_Disconnect_Reply 10
   4.3. Standard Parameters 11
      4.3.1. Task_slots 11
      4.3.2. Mode 11
      4.3.3. Service_id_string 11
      4.3.4. I2T_queue_no 12
      4.3.5. T2I_queue_no 12
      4.3.6. Result 12

5. Data structures 13
   5.1. Operation Request Blocks (ORBs) 13
      5.1.1. Command Block ORB Format 13
      5.1.2. Status Block 14
      Parameter Encoding..16
   5.2. Error Reporting Precedence 17

2. Figure

Page

Figure 1 – Command block ORB 13
Figure 2 – Status block format...14
Figure 3 – Parameter ID and value format 16

3. PWG Function/Service Model

Function (Instance) = unit directory with Logical Unit Number (LUN) 0

Service = Control protocol for Independently operable component of a Function

Queue = Ordered set of ORB’s that does not block with respect to other queues. ORBS on a queue shall be executed in order.

Connection = Set of Queues that affords access to a service

Channel = A special case of a connection which uses two queues to form a bi-directional connection.

Mixed Queue = A queue which may have ORBs for either direction. ORBs on a mixed queue shall be executed in order.

Command = Instructions defined in this specification that are delivered in an ORB

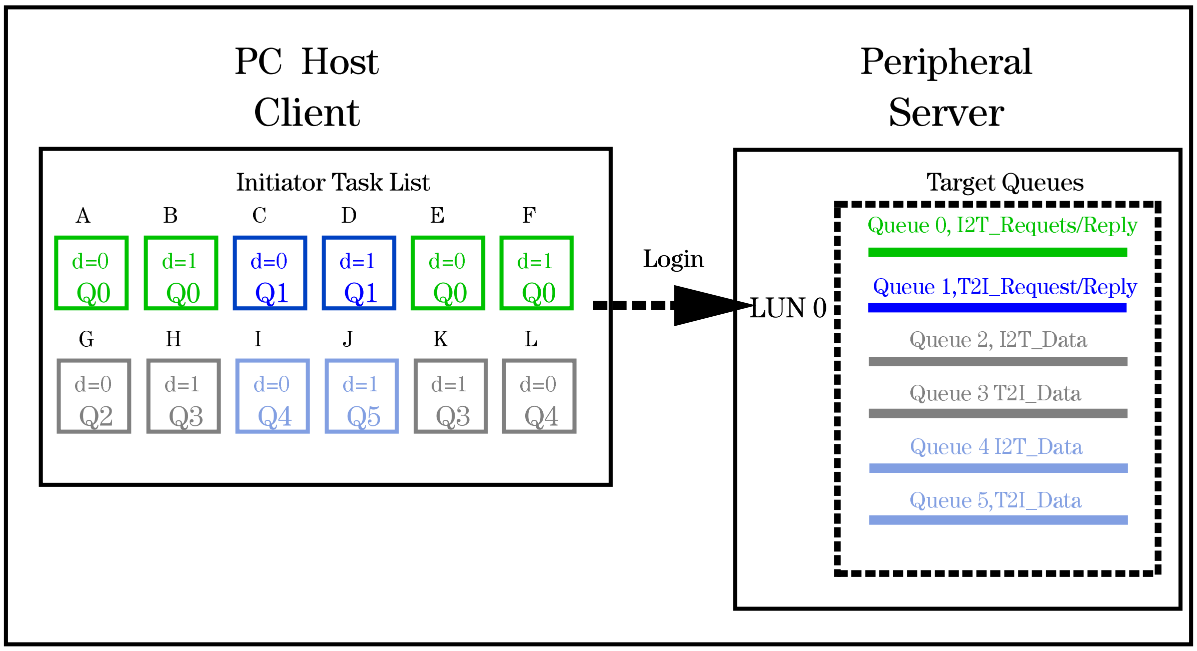
Action = Instructions defined in this specification that are delivered in the data buffer referenced by an ORB’s data_descriptor

The fundamental communication mechanism is a bi-directional non blocking channel. This channel is realized by ORB’s on an Initiator’s Task List which are fetched and organized into two queues by the Target. Uni-directional and mixed queues are also supported.

A unit directory represents the function and has exactly one Logical Unit Number, LUN zero. To establish communication to the Function, an Initiator shall Login to LUN zero. At successful Login, Queue zero is automatically established as the Initiator to Target Command Management queue. Data commands shall not be used on queue zero. Queue number one is automatically established as the Target to Initiator Command Management queue. Data commands shall not be used on queue zero. Data channels are created by using Connect Actions over queue zero or queue one.

Since the PWG Transport needs to concurrently process data transfers in both directions, the target implementing the PWG Transport must report itself to SBP-2 specifying the UNORDERED execution model. For bi-directional communication the PWG Transport target shall use two ordered execution queues. Each queue maintains an ordered execution sequence of Tasks for a single direction of data flow.

Queue numbers assigned

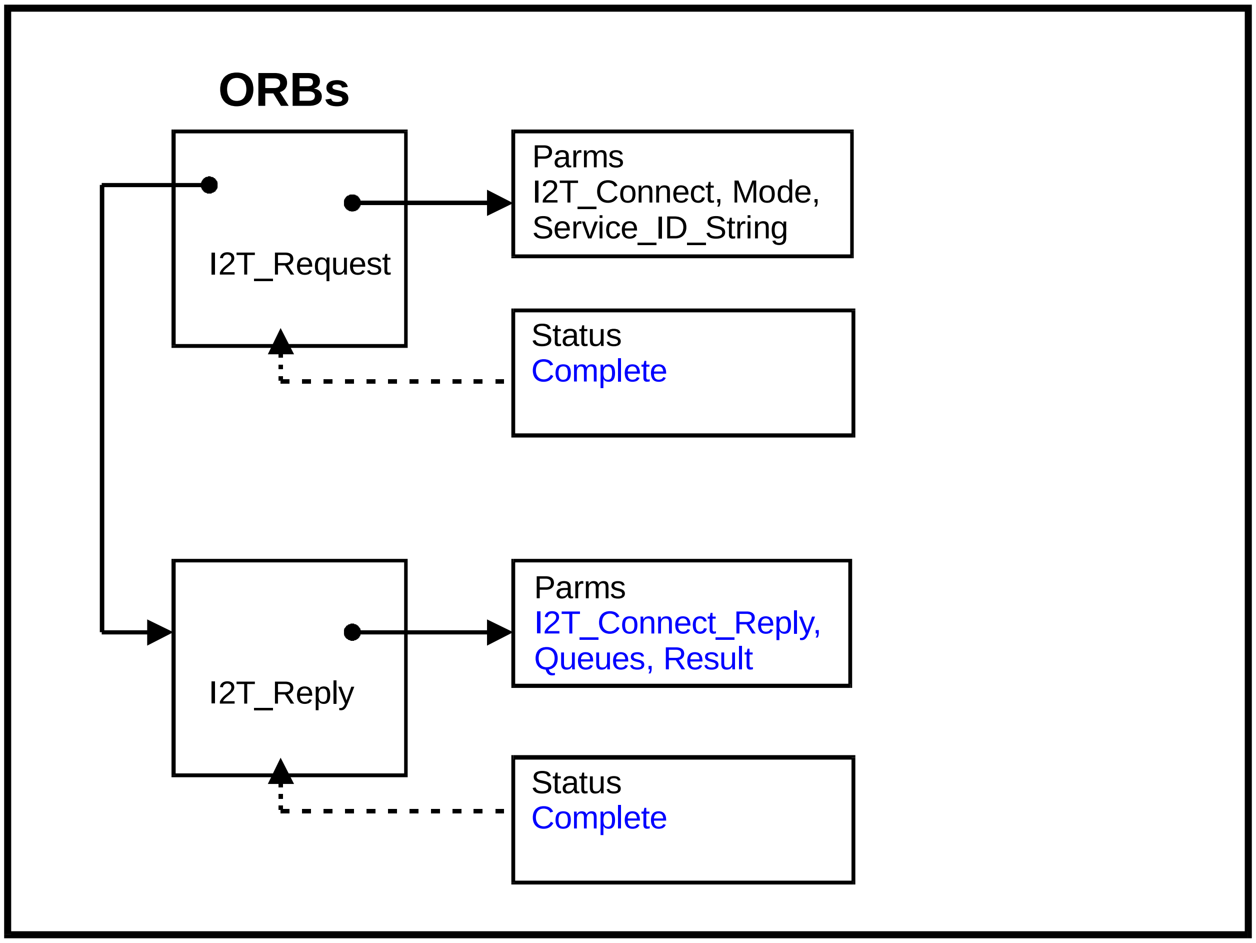


3.1. Initiator To Target Communication

Initiator to Target Commands and Actions are transferred using queue zero, which is a mixed queue. Commands use a request reply format that always uses two ORBs. The first ORB is used for the request and the second is used for the reply. The Actions also use the request reply format and are encoded as Parameters. Actions and parameters are transferred as part of the data associated with an ORB.

Commands are basically used to communicate the capabilities of the transport and to communicate a type of data transfer. Actions are used to establish and dissolve connections. Actions are located in the Data portion to also allow a Target to issue actions. The SBP-2 status block contains the outcome of the data transfer, it does not mean that any Actions were successfully accomplished. The outcome of an Action is communicated in the result parameter.

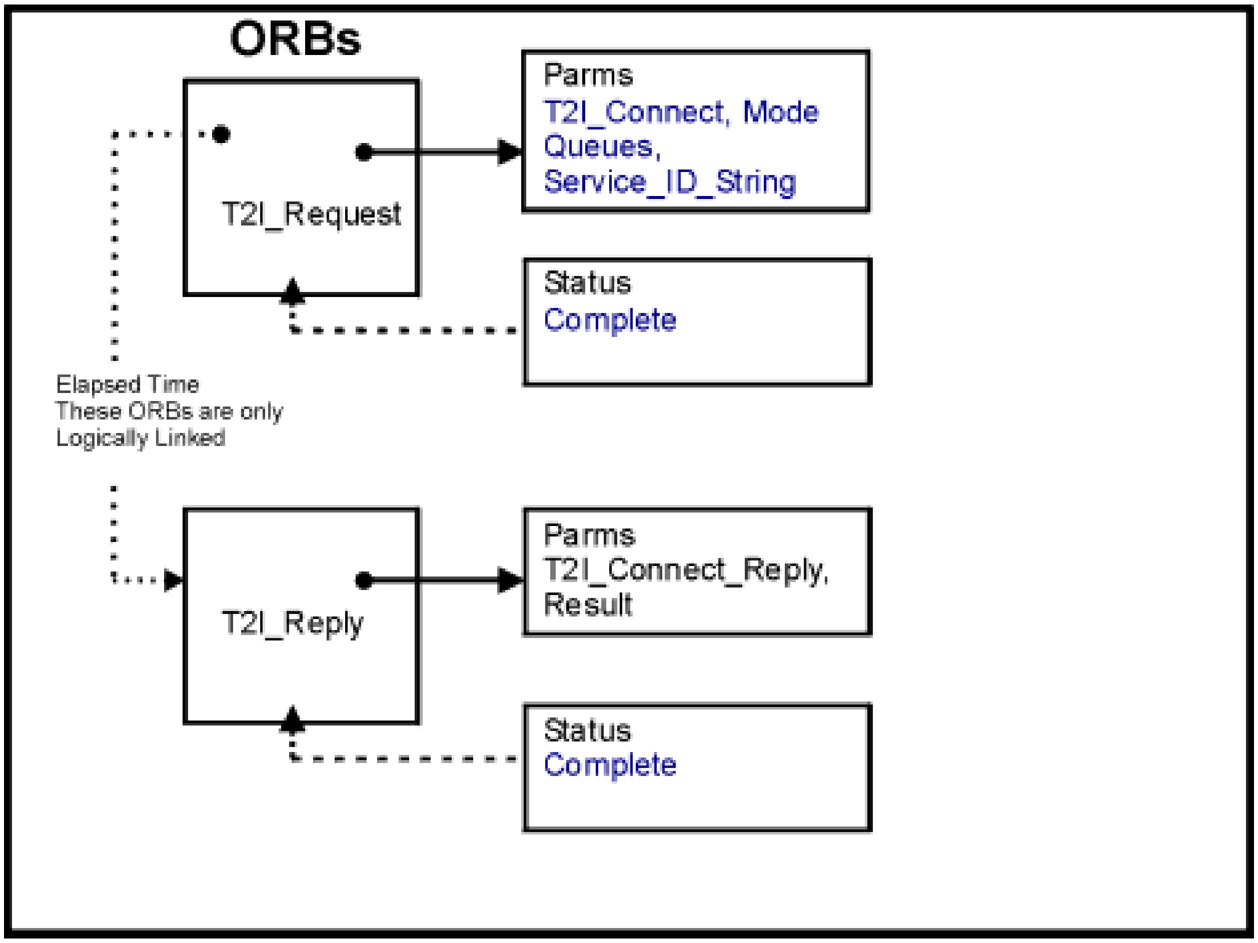
Example Commands and Actions to request a connection.



3.2. Target To Initiator Communication

Target to Initiator Commands and Actions are transferred using queue one, which is a mixed queue.

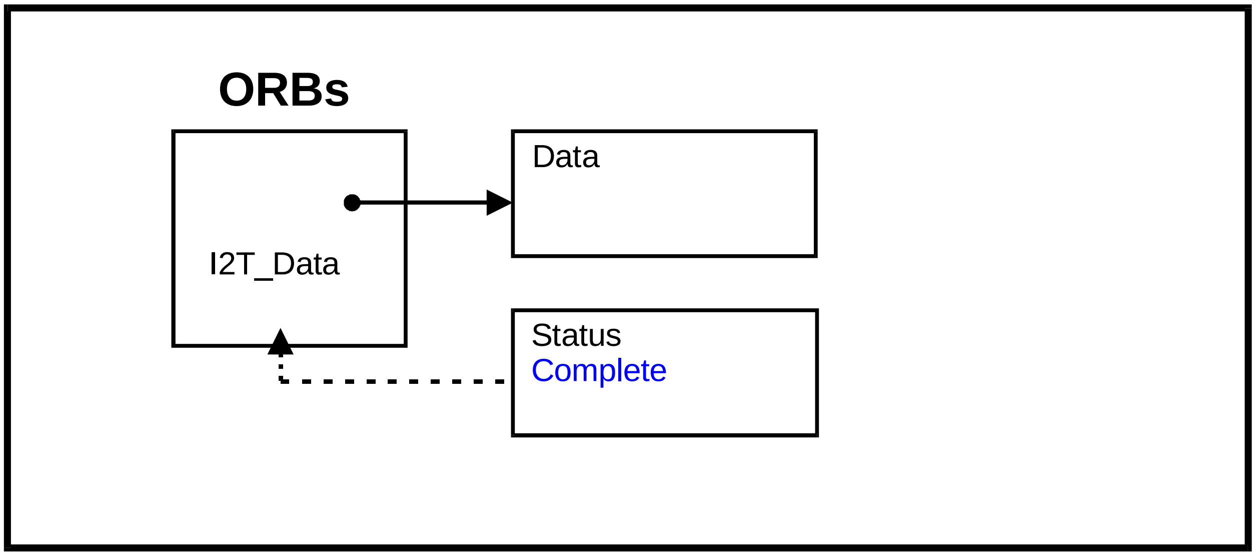
Example Commands and Actions to request a connection.

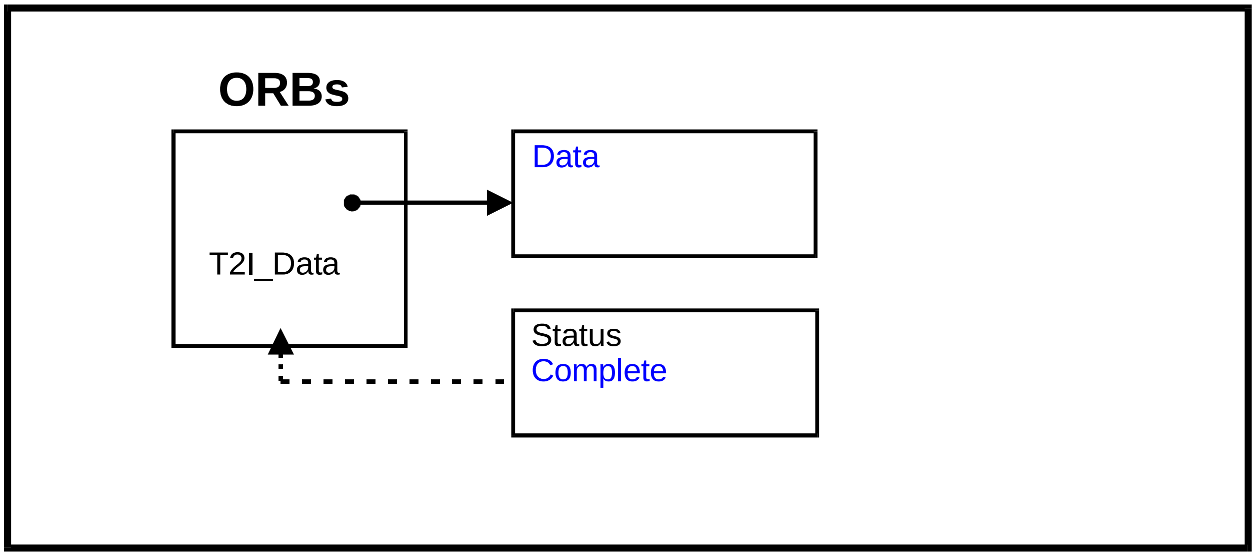


.

3.3. Data Transfer

Data transfer is standard SBP-2.





4. Command Set

4.1. Commands

Command	Value	Description
RESERVED	0	
TRANSPORT_CAPABILITIES	1	None
TRANSPORT_CAPABILITIES_REPLY	2	[??? Still need something like this?] Returns Max_task_set_size parameter
I2T_REQUEST	3	Request channel Actions and parameters from the Initiator to the Target
I2T_REPLY	4	Reply channel Actions and parameters from the Target to the Initiator
T2I_REQUEST	5	Request channel Actions and parameters from the Target to the Initiator
T2I_REPLY	6	Reply channel Actions and parameters from the Initiator to the Target
I2T_DATA	7	Request the target to transport the data in the buffer associated with the ORB to the target.
T2I_DATA	8	Request the target to transport, when available, any data from the target to the buffer associated with the ORB.

4.2. Actions

Action	Value	Parameters
RESERVED	0	
I2T_CONNECT	1	Mode, Task_slots, Service_id_string
I2T_CONNECT_REPLY	2	I2T_queue_no, T2I_queue_no, result
I2T_DISCONNECT	3	I2T_queue_no, T2I_queue_no
I2T_DISCONNECT_REPLY	4	result
T2I_CONNECT	5	Mode, Task_slots, I2T_queue_no, T2I_queue_no, Service_id_string
T2I_CONNECT_REPLY	6	result
T2I_DISCONNECT	7	I2T_queue_no, T2I_queue_no
T2I_DISCONNECT_REPLY	8	result

In order to setup the bi-directional communication queues, the Initiator and Target must complete the following ORB commands and actions in sequence:

1. TRANSPORT_CAPABILITIES – to retrieve the transport-layer capabilities provided by the target. This is necessary for the initiator to find the maximum task set size cacheable by the target.
2. TRANSPORT_CAPABILITIES_REPLY(Max_task_set_size) – Returns the maximum task set size.
3. I2T_REQUEST with Action = I2T_CONNECT(Mode, Service_id_string) – Request a connection with the target's transport client.
4. I2T_REPLY to Receive Action = I2T_CONNECT_REPLY(I2T_queue_no, T2I_queue_no, result) – Receive the connection reply and the channel queues.

Once the connection has been established, the initiator may issue data commands for data to be transferred, provided there is room on the active task list. All transport client data shall be referenced by the ORB's data_descriptor.

4.2.1. I2T_Connect

This action is used to establish a connection from the Initiator to the Target. The initiator uses the mode parameter to determine the type of connection. The Service_id_string is known to the application requesting the connection and is determined through driver dependent means based on the type of device which was discovered.

4.2.2. I2T_Connect_Reply

In response to the I2T_Connect the Target returns two queue parameters and the result.

4.2.3. I2T_Disconnect

It is recommended that this Action is sent “synchronously” after all the data has been transferred and acknowledged with status blocks. The Action should be sent on the Data Queue The Initiator and the Target shall wait for the reply before performing house keeping associated with the channel.

The Target will know that this is a “synchronous” disconnect since it was received on a data channel. At this point the Target should have no outstanding data transfer request for the Initiator on this channel.

This Action can also be sent on the Command Management queue as an Asynchronous disconnect.

4.2.4. I2T_Disconnect_Reply

After the Target sends the reply it is free to re-use the queue number(s) associated with the disconnected channel.

4.2.5. T2I_Connect

This action is used to establish a connection from the Target to the Initiator Target. The Target uses the mode parameter to determine the type of connection and provides the queue numbers. The Service_id_string is discovered or remembered? [Needs More explanation]

4.2.6. T2I_Connect_Reply

In response to the T2I_Connect the Initiator returns the outcome of the request in the result

4.2.7. T2I_Disconnect

It is recommended that this Action is sent “synchronously” after all the data has been transferred and acknowledged with status blocks. The Action should be sent on the Data Queue The Initiator and the Target shall wait for the reply before performing house keeping associated with the channel.

The Initiator will know that this is a “synchronous” disconnect since it was received on a data channel. At this point the Target should have no outstanding data transfer request for the Initiator on this channel. [The reverse channel will probably only have a T2I_Data Command hanging around, not a T2I_Request, so how is an explicit synchronous disconnect on the data channel possible?]

This Action can also be sent on the Command Management queue as an Asynchronous disconnect.

4.2.8. T2I_Disconnect_Reply

After the Initiator sends the reply the Target is free to re-use the queue number(s) associated with the disconnected channel.

4.3. Standard Parameters

4.3.1. Task_slots

This *Task_slots* parameter returns the maximum size of the active task set across supported by this connection. The initiator must avoid queuing a greater number ORBS than the value of Task_slots on the Task List. All targets shall support at least one active task for each ordered queue. This value will be established with a connect action.

ID	Parameter	Access	Size	Description
128	Task_slots	RO	14 bits	Returns the maximum number of pending ORBs for a connection.

4.3.2. Mode

The *mode* specifies the type of communication channel(s) requested.

ID	Parameter	Access	Size	Description
129	Mode	RO	8 bits	Specifies the type queue(s) requested for the Connection.

The *mode* is encoded according to the following table:

Mode	**Value**	**Description**
I2T-direction	0	this specifies a single queue for communication in the Initiator to Target direction
T2I-direction	1	this specifies a single queue for communication in the Target to Initiator direction
Bi-directional	2	this specifies two queues, one in each direction
Mixed	3	this specifies a single queue for request reply information.

4.3.3. Service_id_string

The *Service_id_string* is the name of the service requested. The Initiator or Target implicitly know what services a device offers based on its function and other information provided through discovery.

ID	Parameter	Access	Size	Description
130	Service_id_string	RO	variable Max??	Type service requested

4.3.4. I2T_queue_no

The *I2T_queue_no* is used in the *queue_field* of an ORB.

ID	Parameter	Access	Size	Description
131	I2T_queue_no	RO	8 bits	Queue number to be used for communication on this channel

4.3.5. T2I_queue_no

The *T2I_queue_no* is used in the *queue_field* of an ORB.

ID	Parameter	Access	Size	Description
132	T2I_queue_no	RO	8 bits	Queue number to be used for communication on this channel

4.3.6. Result

ID	Parameter	Access	Size	Description
133	result	RO	8bits	Outcome of the Action

Result	**Value**	**Description**
Success	0	The Action was accomplished and the return parameters are valid
Busy	1	Connection could not be established at this time.
No Service	2	The requested service does not exist
Refused	3	Connection not permitted
Invalid queue	4	Queue number in ORB dose not exist
Unspecified Error		An error occurred which is not specified in this table

5. Data structures

There are three data structures described by this standard:

– SBP-2 Operation request blocks (ORBs);

– SBP-2 Status blocks;

– Action and Parameter encoding.

5.1. Operation Request Blocks (ORBs)

All ORB formats described in SBP-2, rev 4, shall be supported.

5.1.1. Command Block ORB Format

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.

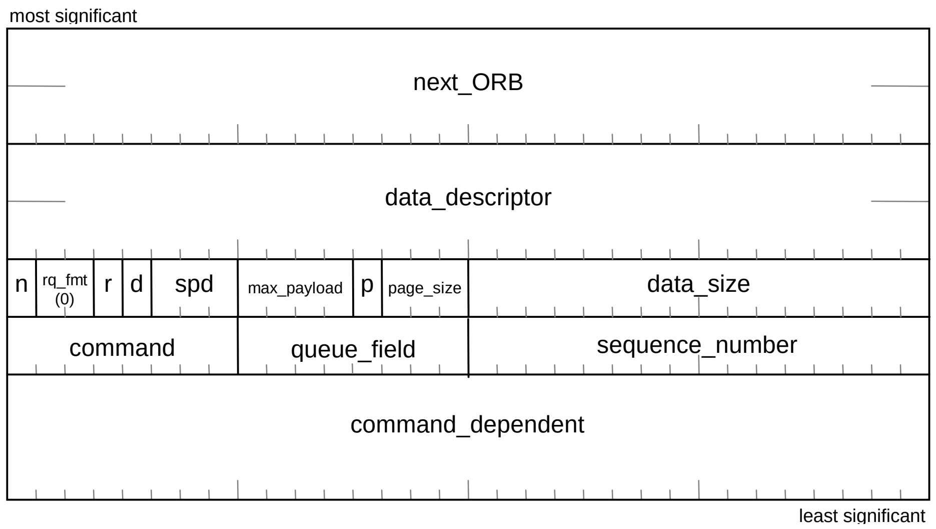


Figure 1 – Command block ORB

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 *command* field shall specify the command to be executed by the target. The *command* field shall be interpreted according to the table below. All unlisted values are reserved.

The *queue_field* shall identify the ordered execution queue to which the ORB belongs.

The *command* field shall specify the command to be executed by the target.

The *sequence_number* field carries the initiator-assigned sequence number for this command instance. The sequence numbers are private to each queue. Connection Parameter

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

5.1.2. Status Block

A target shall store status at an initiator *status_FIFO* address when a request completes and either the notification bit is set, an error occurred, or the completion notification contains other than zero bits beyond the first two quadlets. Status only pertains to the data transfer operation of the transport layer. It does not imply that any Action in the data portion completed successfully.

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.

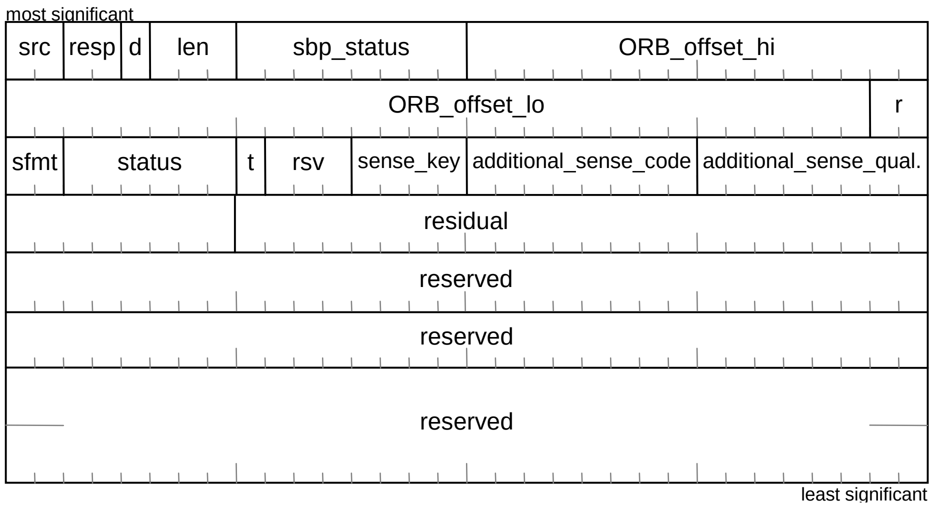


Figure 2 – Status block format

The target shall store a minimum of twelve 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, 2	Reserved for future standardization
3	Status block format vendor-dependent

The *status* field shall contain status information as defined by this standard. The status field shall only be value provided no SBP-2 errors are reported. The receipt of any status shall indicate that the associated task has ended. The following table defines permissible values for *status*.

***status* value**	**Description**
0	GOOD
2	CHECK CONDITION
$3F_{16}$	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.

CHECK CONDITION. This status indicates that the target has detected a condition that has stopped the Fetch Agent and implicitly aborted the active task set.

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 I2T_DATA, TRANSPORT_CAPABILITIES, and I2T_CONNECT 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 shall contain command completion information defined in this standard.

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.

A non zero residual value is not necessarily an error.

5.1.3. Parameter Encoding

All parameters passed in the buffer associated with the ORB (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.

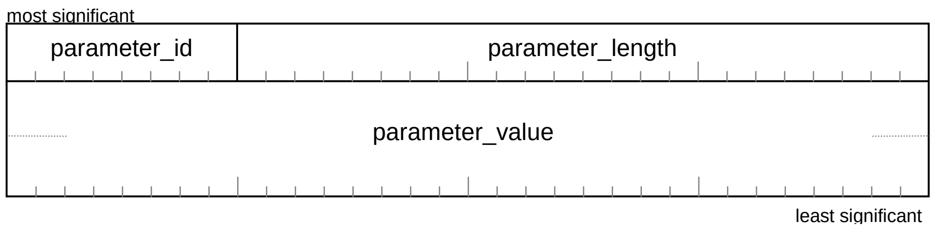


Figure 3 – Parameter ID and value format

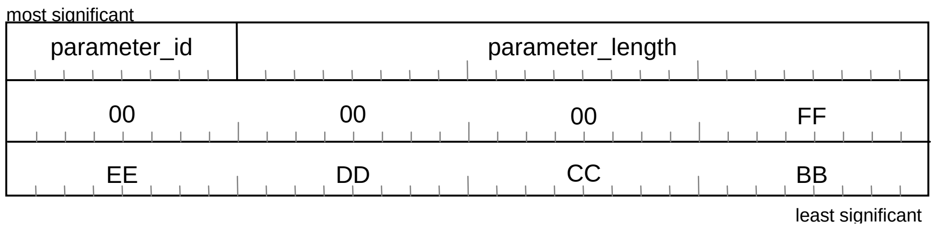
The *parameter_id* value shall indicate what type of information is encoded in the *parameter_value* field.

The *parameter_length* field shall indicate the size (in bytes) of valid data in the *parameter_value* field.

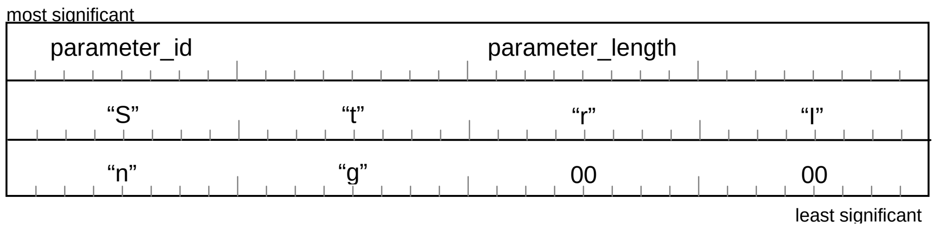
The *parameter_value* field contains the actual data for the identified parameter. The data may be of an arbitrary byte length greater than zero. The data shall be padded with reserved bits to the next quadlet boundary. The *parameter_length* field shall not be adjusted to include this padding.

Data shall be stored in the least significant bits of the *parameter_value* field. For padding sess section 3.2.2 SBP-2 revision 4 figure 3.

Example: the five byte number FFEEDDCCBB is encoded and padded as follows:



Example: the six character string “String” is encoded and padded as follows :



5.2. Error Reporting Precedence

The precedence of error reporting, and the reported values, shall be as follows:

1. SBP-2 errors

2. Unit Attention Condition. If a unit attention condition exists, the target shall report the following condition:

Resp	Sbp_status	Status	Sense key	ASC, ASQ
0 - REQ COMPLETE	0 - NO ADDL INFO	2 - CHECK COND	6 - UNIT ATTENTION	29,0 –RESET, OR DEVICE RESET OCCURRED

3. Data length not supported by protocol (>= 2^{31} bytes). For this error, the target shall report the following condition:

Resp	Sbp_status	Status	Sense key	ASC, ASQ
0 - REQ COMPLETE	0 - NO ADDL INFO	2 - CHECK COND	5 - ILLEG REQ	49.0 – INVALID PACKET SIZE

4. *command* not supported. For this error, the target shall report the following condition:

Resp	Sbp_status	Status	Sense key	ASC, ASQ
0 - REQ COMPLETE	0 - NO ADDL INFO	2 - CHECK COND	5 - ILLEG REQ	26,0 – INVALID COMMAND OPERATION CODE

5. If this command is issued to the wrong queue, then the target shall return status of

Resp	Sbp_status	Status	Sense key	ASC, ASQ
0 - REQ COMPLETE	0 - NO ADDL INFO	2 - CHECK COND	5 - ILLEG REQ	21.1 – INVALID ELEMENT ADDR

6. Command-specific errors

7. Unknown or unspecified errors

Resp	Sbp_status	Status	Sense key	ASC, ASQ
0 - REQ COMPLETE	0 - NO ADDL INFO	2 - CHECK COND	0 – NO SENSE	0,0 – NO ADDITIONAL SENSE INFORMATION