1394 PRINTER WORKING GROUP

IEEE 1394 HIGH SPEED BUS

IMAGING DEVICE

COMMUNICATIONS SPECIFICATION

***PRELIMINARY DRAFT PROPOSAL ***

Revision 0.52 - September 15, 1998

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

1. Contents

Page

2. Figure

Page

3. Scope

This document specifies the communications profile and device communications command set for imaging devices attached to the IEEE 1394 High Speed Serial Bus.

SBP-2 provides multiple, concurrent, independent connections which do not preclude concurrent operation of other protocol stacks; is data, application, and OS independent. In order to meet the requirements for imaging devices, SBP-2 must be supplemented by this a device profile.

Information supplemental to the IEEE 1394 and SBP-2 specifications are provided in this document.

- a policy to be used for maintaining device access across “transient” link interruptions
- a model of use for maintaining guaranteed, in-order data deliver across “transient” link interruptions
- a command set which supports
    - independent, bi-directional, half-duplex communication
    - data tagging of an associated data payload
    - ability for either end of a connection to close the connection at any time

This specification does not address:

- Isochronous communication
- Use with 1394.1 bridges.
- Security.

4. Purpose

The purpose of this document is to define the communications specification for IEEE 1394 printers, scanners, digital still cameras and other imaging devices. This specification will include traditional computer host communication to these devices as well as direct peer to peer communication.

The term “image device” is used throughout the remainder of this document to refer to image devices in general including any of the devices listed above.

The primary focus of this document is related to the SBP-2 protocol and how it can be used for image device communication. Requirements are specified to allow imaging device communication conformance to SBP-2. In all areas that concern transport protocol, SBP-2 should be followed. Where SBP-2 allows more than one choice of implementation, this profile defines the choice for imaging devices.

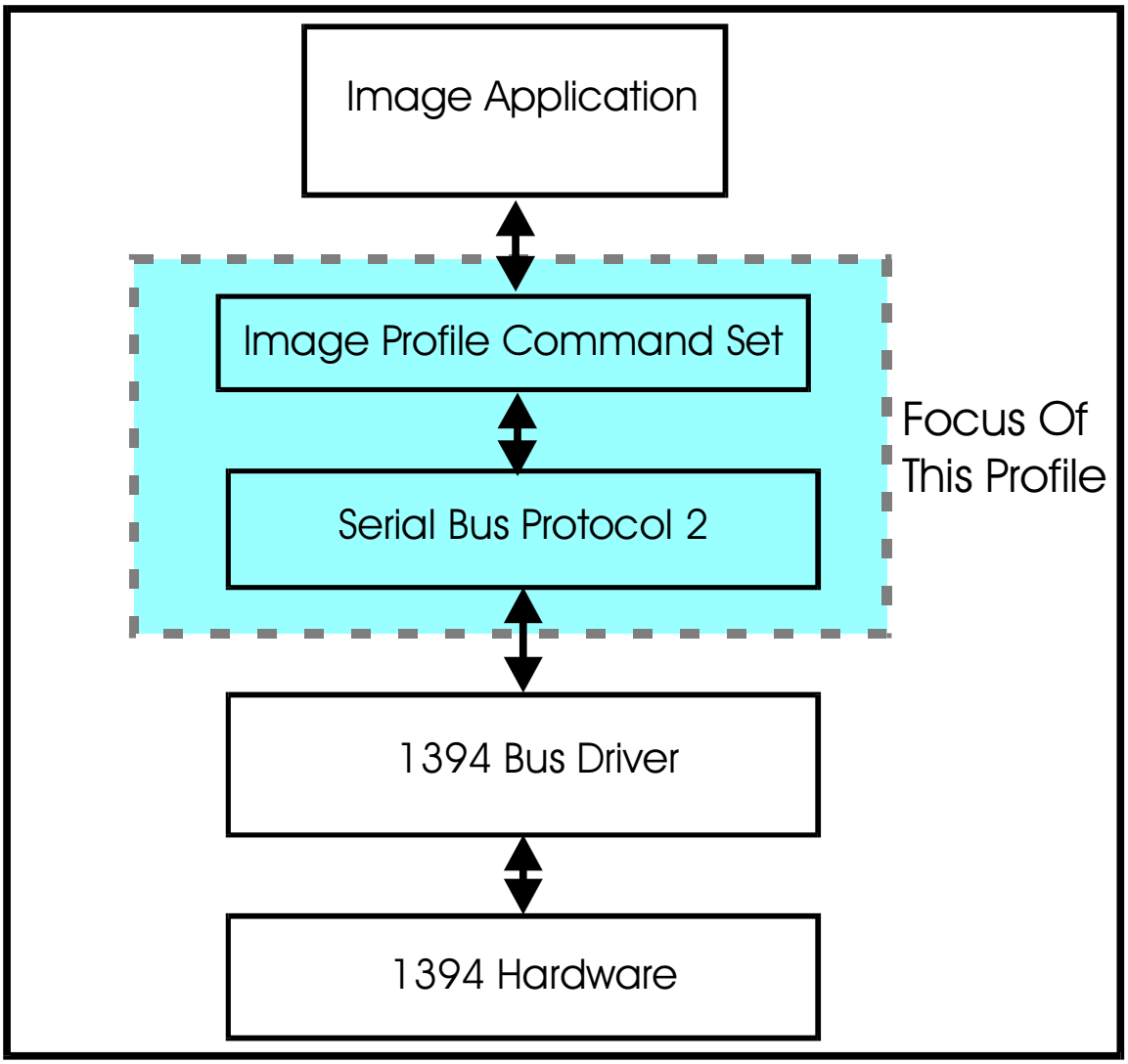


Figure 1 - Focus Block Diagram

5. References

This document makes reference to and contains excerpts from several industry standards. The revisions of those standards listed are current at the time of this document's release. However, each standard referenced is subject to change. More recent revisions may or may not support the information contained in this document:

1) ISO/IEC 13213 ANSI/IEEE 1212:1994, Control and Status Register Architecture for Microcomputer Buses.
2) IEEE Std 1394-1995, Standard for High Performance Serial Bus.
3) Serial Bus Protocol 2, Revision T10/1155x.
4) P1394a Draft Standard for a High Performance Serial Bus (Supplement).

5.1. Definitions

5.1.1. Function:

A capability of the node which is accessed by one or more units.

5.1.2. Queue:

A collection of ORBS that proceed or block independent of other queues. Orbs within a queue are processed in order.

5.1.3. Service:

A sub component of a unit that operates on a set of tasks independent of other services of the same unit. For SBP-2, there is a one-to-one correspondence between a service and a logical unit (LUN). For example, a printer (unit) might implement a PDL service responsible for data flow to the printer and a management service responsible for control and status.

5.1.4. Unit:

A unit is synonymous with a Unit Directory. A unit has a one to one correspondence with a device driver. From SBP-2: A component of a Serial Bus node that provides processing, memory, I/O or some other functionality. Once the node is initialized, the unit provides a CSR interface that is typically accessed by device driver software at an initiator. A node may have multiple units, which normally operate independently of each other. Within this standard, a unit is equivalent to a target.

5.2. Bit, Byte and Quadlet ordering

See IEEE std. 1394-1995.

5.3. Control & Status Registers (CSR)

All 1394 PWG devices shall implement the CSRs as defined in ISO 13213/IEEE 1212:1994 and IEEE Std 1394-1995. Required registers are the same as for SBP-2.

6. Configuration ROM

All 1394 PWG devices shall implement configuration ROM as defined in ISO 13213/IEEE 1212:1994 and IEEE Std 1394-1995. The ROM directory structure is a hierarchy of information blocks with the blocks higher in the structure pointing to the blocks beneath them. The locations of the initial blocks, *Bus_Info_Block* and *Root_Directory*, are fixed. The locations of the other entries are specified in the *Root_Directory* and its associated directories.

Reserved fields shall be set to zero.

Length values in the Configuration ROM specify the number of Quadlets.

There are two types of offsets specified by ISO 13213/IEEEE 1212.

1) Initial register space offset which is an offset in quadlets from the initial register space base address of 0xFFFF F000 0000. Value contained in the register multiplied by 4 plus base address.
2) Indirect space offset, which is an offset in quadlets from the current register address. Value contained in the register multiplied by 4 plus address of register.

Number 1 above has a key_type of 0x1. Number 2 above has a key_type of 0x2 or 0x3, see ISO 13213/IEEEE 1212 section 8.2.4 table 21 for all key_type definitions.

7. Discovery

The primary method for discovering devices on the Serial Bus is through information read from the Configuration ROM.

This section will be continued with the Function Discovery Service information supplied by the PWG and IEEE 1212r when this is specified.

8. Communication Model

This specification defines the basic communication path as a Login from an Initiator to a Target. For a given Login, the Initiator provides a single linked list of ORBs called the task list and the Target fetches ORBs from this task list.

Why?For bi-directional communication, devices may use the out of order ORB processing model described in the next section.

8.1. Uni-Directional Communication

The initiator may configure the Target for data transfer in a single direction. Either from the Initiator to the Target or from the Target to the Initiator. See the command set parameters for details.

8.2. Bi-Directional Communication Model

This specification recommends that the Initiator configure the Target for data transfer in both directions. From the Initiator to the Target and from the Target to the Initiator. See the command set parameters for details. This profile provides full duplex communication capability between an initiator device and a target device using an unordered ORB processing model. Data transfer within a specific direction is accomplished in order with respect to that direction. If the Initiator configures the device for bi-directional communication it shall insure that a sufficient number of ORBs are available for communication in each direction. The total number of ORBs on the task list which includes ORBs for each direction can be configured using the set parameters command with the max task set size parameter.

8.3. Target Model

Figure 2 illustrates an example block diagram of a command block agent for the target. The command block agent contains one command fetch agent, two command pre-fetch queues, called **Write queue** and **Read queue**, and two execution agents, called **Write execution agent** and **Read execution agent** connected to the Write queue and the Read queue respectively.

The command fetch agent fetches the normal command block ORB's in order. When the command fetch agent fetches the normal command block ORB, the command fetch agent examines the command specified in the *command_block* field of the command block ORB. The fetch agent dispatches the command block ORB to either the Write queue or Read queue according to the parameter. All Write commands are dispatched to the Write queue, and all Read commands are dispatched to the Read queue. The Write execution agent and Read execution agent, execute the commands queued in the Write queue and Read queue respectively.

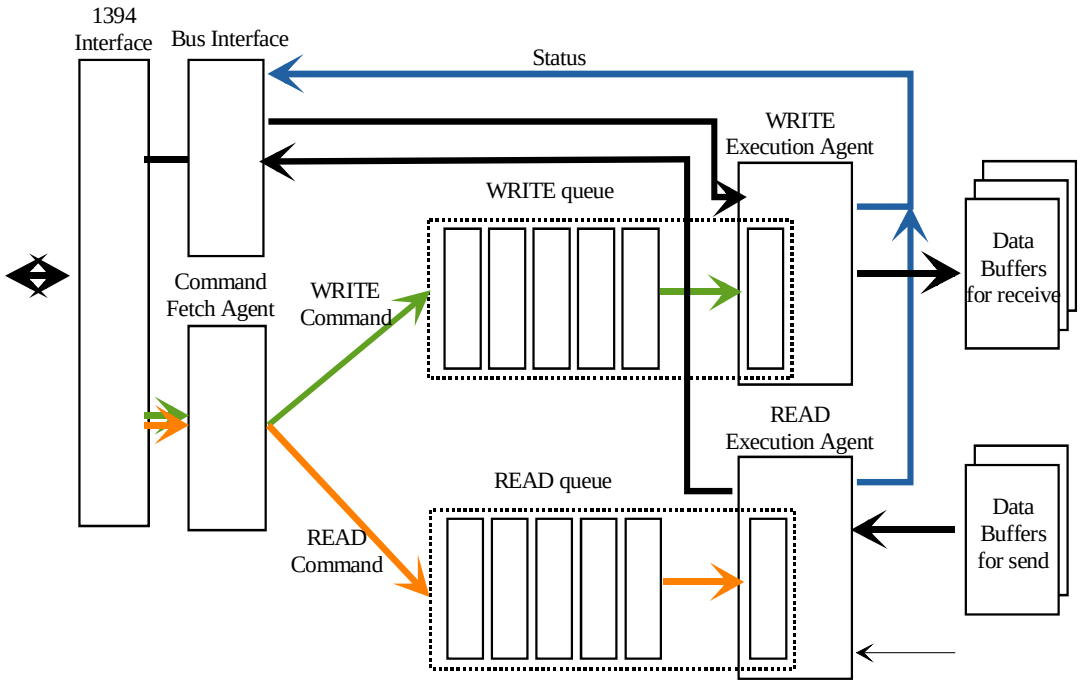


Figure 2 - Target Model

Each execution agent executes the dispatched command in the connected queue in order.

Both execution agents are independent of each other. Each execution agent executes the data transfer associated with the command according to the parameters specified in the command.

The target stores a status block in the initiator's memory according to the value of the *notify* bit of the command block ORB after executing the command as specified by SBP-2. Each execution agent shall store the status_block in order of execution for that particular agent.

While fetching ORBs the fetch agent is free to process the task list according to the values in the next_ORB field. After an ORB has been fetched and placed in the Read or Write queue, fetch agents (and execution agents) shall not refer to Initiator memory addressed by the *next_ORB* field. The ORBs addressed by the *next_ORB* field in the initiator's memory may not contain a valid pointer since the addressed ORB may already be completed due to unordered execution.

8.4. Initiator Model

The initiator may have two **i/o request queues** as illustrated below.

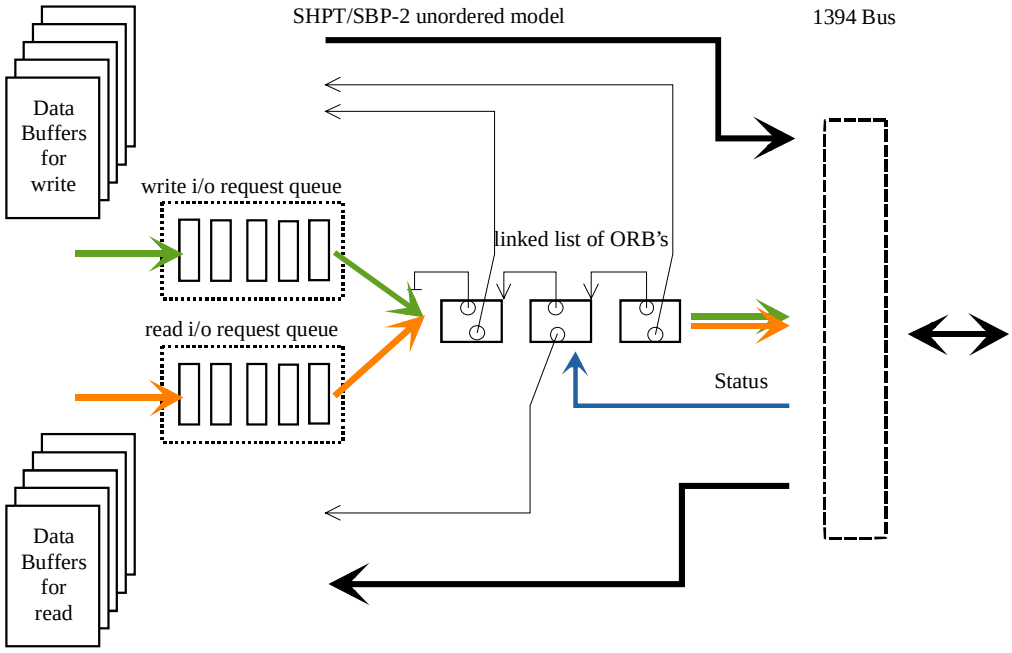


Figure 3 - Initiator Model

The **Write queue** and **Read queue** in the target queue the command ORB's destined to the **Write execution agent** and **Read execution agent** respectively. The initiator may only append a new **task** to a current **task list** when there is space in the list according to the combined depth count for the task list. In order to manage this constraint, the initiator retrieves the depth of the task list from the target before starting communication.

The SBP-2 status block notifies the Initiator that an item on the task list has been completed. The Initiator may release the memory associated with the ORB and the request can be removed from the appropriate queue according to the command.

NOTE – For targets that support the ordered model of task execution, the return of completion status for an ORB implicitly indicates that all preceding ORB's in the linked list have completed successfully, are no longer part of the task set and that the initiator may reuse or de-allocate their system memory.

NOTE: The initiator does not need to update the *next_ORB* field of the completed ORB in the current **task set**, since the target never refers to this field retroactively.

9. Multiple Host and/or Multiple Device

There is a need to provide fair access to devices on a 1394 bus. Each node is accessible from any of the other nodes on the bus and possibly nodes outside of the bus in a bridged environment. It is reasonable to expect that more than one initiator node may attempt to communicate with a target service at the same time.

SBP-2 provides a Login function. A successful Login creates a connection between two nodes. This creates the required instance data within the target memory. Devices that conform to this profile are required to support a minimum of a single Login. A specific implementation may support multiple logins and arbitrate between them.

10. Management ORBs

The following table specifies support for Management ORB functions.

Table 12 - Management ORB Functions Support List

Function Value$_{16}$	Management Function	Support Level
0	Login	Mandatory
1	Query Login	Mandatory
3	Reconnect	Mandatory
4	Set Password	Optional
7	Logout	Mandatory
B	Abort Task	Not supported
C	Abort Task Set	Mandatory
E	Logical Unit Reset	Not supported
F	Target Reset	Mandatory

11. Login & Login Response

The primary reasons for Login are, access control, unsolicited status and the simple SBP-2 reconnect scheme.

Writing “resources_unavailable”, in the sbp_status field of the status block, to the Login’s Status_FIFO address will reject a Login.

The *login_response_length* shall accommodate the size, in bytes, the Login Response specified in this standard.

12. Unsolicited Status

As of the May 1998 PWG meeting a need for Unsolicited Status has not been identified.

The reason for the handshake for the Unsolicited status is because of it's unsolicited nature. The initiator when preparing a FIFO to receive status knows how many ORB's it has given or will give to the target. The Initiator can allocate enough FIFO for those status reports. Since the Initiator does not know how many Unsolicited reports it may receive it is required to allocate at least one FIFO location and use the handshake with Unsolicited_Status_Enable when that one is available.

If a Target wants to send unsolicited status and the Unsolicited_Status_Enable regsiter is not set the Target shall wait at least a reconnect time out period before asserting a 1394 bus reset.

13. Command Set

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 as using the UNORDERED execution model. This allows the target to complete Task execution however it needs to. Within the PWG Transport target are two ordered execution queues. Each queue maintains an ordered execution sequence of Tasks for a single direction of data flow.

In order to setup the bi-directional communication queues, the initiator must complete each of the following commands in sequence:

a) 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.

b) TRANSPORT_OPEN – to configure the transport to use the reduced set of capabilities indicated by the initiator, to cause the target to allocate any necessary resources, and to attempt to establish a connection with the target's transport client.

Once the connection has been established, the initiator may issue the following 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.

- TRANSPORT_I2T_DATA – to request the target to transport the data in the buffer associated with the ORB to the target's client.
- TRANSPORT_T2I_DATA – to request the target to transport, when available, any data available from the target's client to the buffer associated with the ORB.

If the initiator wishes to stop accepting T2I data before disconnecting, it shall issue a TRANSPORT_CLOSE on the queue associated with TRANSPORT_T2I_DATA. This is necessary to allow the target to indicate the transport failure to the target's transport client, so that it may take appropriate action.

If the initiator wishes to stop generating I2T data before disconnecting, it shall issue a TRANSPORT_CLOSE on the queue associated with TRANSPORT_I2T_DATA. This is necessary to allow the target to indicate the End-Of-File to the target's transport client, so that it may take appropriate action.

When the initiator is ready to completely close the connection, it shall explicitly Logout. It may do this without having either direction closed. An initiator may not close both directions and re-open the connection without executing a Logout and Login. This is explicitly to prevent an initiator from monopolizing the target, which is a shared resource on the bus.

This device model does not support ABORT TASK or LOGICAL UNIT RESET management agent functions. Doing so requires unnecessary complexity in the management of the target's internal ordered-execution queues and opens issues with the in-order delivery of data.

13.1. Data structures

There are three data structures described by this standard:

- SBP-2 Operation request blocks (ORBs);
- SBP-2 Status blocks;

- Command parameter encoding.

13.1.1. Operation Request Blocks (ORBs)

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

13.1.1.1.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.

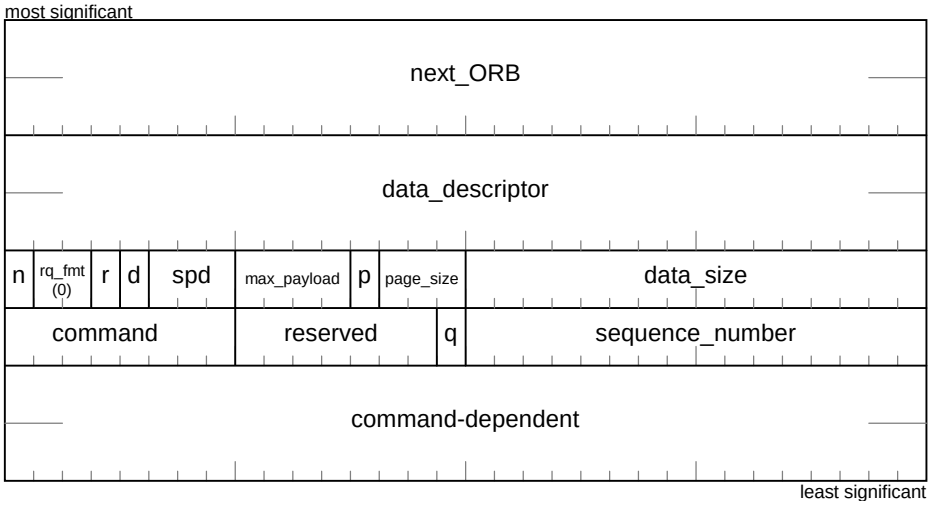


Figure 4 – Command block ORB

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 *queue_ID* bit (abbreviated as *q* in the figure above) shall identify the ordered execution queue to which the ORB belongs. See clause 13.6 Queue Management 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. The *command* field shall be interpreted according to the table below. All unlisted values are reserved

Value	Command	Queue ID	Description
0	TRANSPORT_CAPABILITIES	1	The target returns default setup information in the buffer
1	TRANSPORT_OPEN	0	The target updates its current parameters according to information fetched from the buffer and opens the connection
2	TRANSPORT_I2T_DATA	0	The target reads data from the buffer
3	TRANSPORT_T2I_DATA	1	The target writes data to the buffer
4	TRANSPORT_CLOSE	either	The target is aware that the initiator will queue no additional ORBs on the associated queue.

The *sequence_number* field carries the initiator-assigned sequence number for this command instance. The sequence numbers are private to each queue. See clause 13.7 Sequence Number Management for more information.

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

13.1.2. 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. 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.

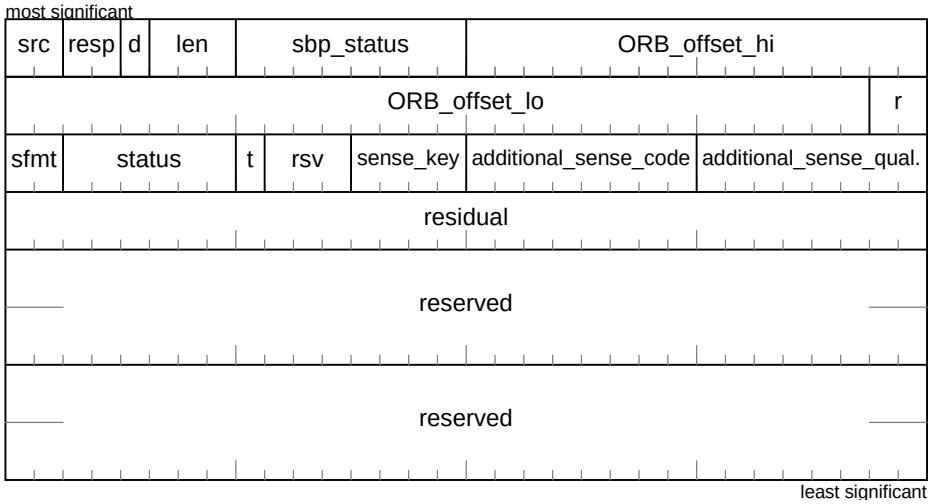


Figure 5 – Status block format

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, 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 TRANSPORT_I2T_DATA, TRANSPORT_CAPABILITIES, and TRANSPORT_OPEN 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.

13.1.3. 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.

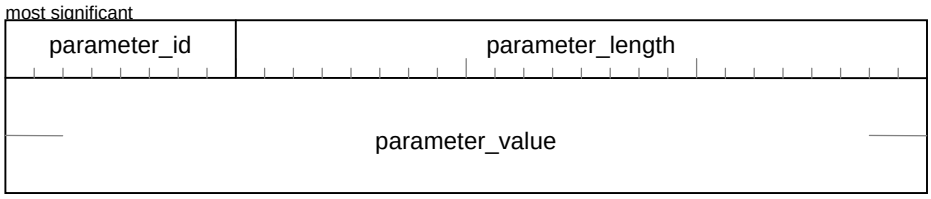


Figure 6 – Parameter ID and value format

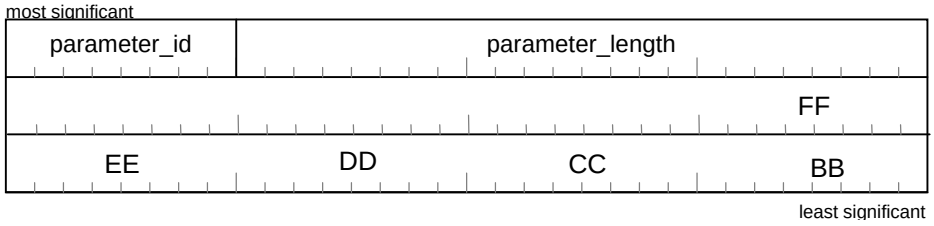
The *parameter_id* value shall indicate what type of information is encoded in the *parameter_value* field.

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 5 byte number FFEEDDCCBB is encoded as follows:



13.2. Standard Parameters

The following table describes all the standard transport parameters. All unspecified values are reserved. The target shall provide all parameters in the TRANSPORT_CAPABILITIES command response. The initiator may restrict the range specified by a parameter for this login by providing a reduced value with the TRANSPORT_OPEN command.

ID	Parameter	Access	Size	Description
1	MAX TASK SET SIZE	RO	14 bits	Returns the maximum number of pending commands across both command queues.
2	MAX I2T DATA SIZE	RW	31 bits	The maximum data size supported for TRANSPORT_I2T_DATA commands.
3	MAX T2I DATA SIZE	RW	31 bits	The maximum data size generated within TRANSPORT_T2I_DATA commands.

13.2.1. MAX TASK SET SIZE

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.

13.2.2. MAX I2T DATA SIZE

This parameter indicates the maximum amount of data that may be provided to the target in a TRANSPORT_I2T_DATA ORB. A value of zero (0) for this parameter indicates the target shall not generate data for the initiator.

NOTE – This size is limited by the target's ability to discard data within an active TRANSPORT_I2T_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. Thus, this parameter is not necessary for stream services.

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?

13.2.3. MAX T2I DATA SIZE

This parameter indicates the maximum amount of data that will be provided by the target in response to a TRANSPORT_T2I_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 TRANSPORT_T2I_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.

13.3. 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

13.4. Commands

13.4.1. TRANSPORT_CAPABILITIES Command

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

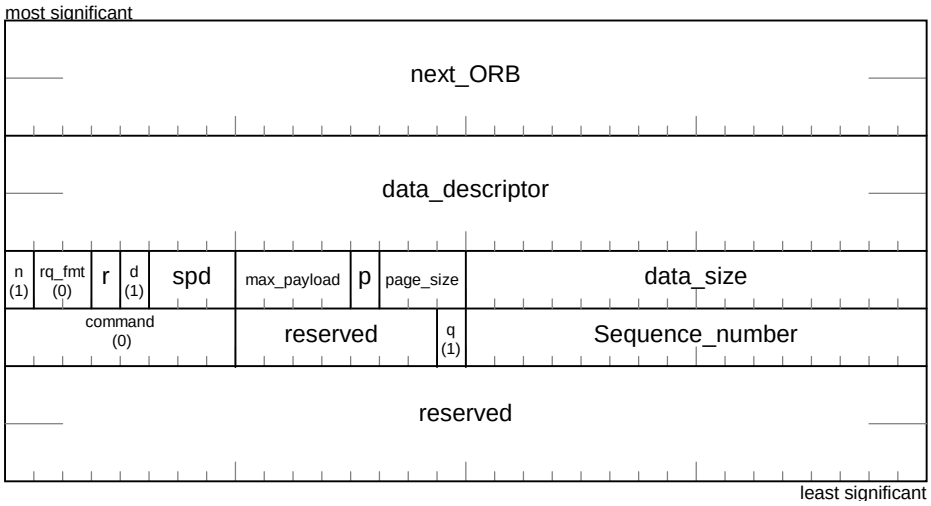


Figure 7 – TRANSPORT_CAPABILITIES Command ORB

All fields shall conform to the descriptions in 3 Command Block ORB Format.

The *notify* bit shall be one.

The *queue_ID* bit shall be one.

The *command* field shall indicate a TRANSPORT_CAPABILITIES command.

13.4.1.1.TRANSPORT_CAPABILITIES Command Response

The command response shall conform to the format described in clause 13.1.2.

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

The data associated with this command response will represent the values defined for the Target and will be presented according to the format described in the Parameter Encoding clause. See 13.1.3 for the encoding of the parameters associated with the target.

The data associated with this command response shall contain encoding(s), as described in clause 13.1.3. for the parameter(s) values of this Target. All behaviors regarding parameter encoding are described in clause 13.1.3

If this command completes successfully, then the target shall return status of:

Resp	Sbp_status	Status	Sense key	ASC, ASQ
0 - REQ COMPLETE	0 - NO ADDL INFO	0 - GOOD	0 - NO SENSE	0,0 – NO ADDITIONAL SENSE TO REPORT

Else if this command contains an invalid field in the CDB, 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	24,0 – INVALID FIELD IN CDB

Else if this command is issued after the connection is open, 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	4A,0 – COMMAND PHASE ERROR

Else if this command is issued with a NULL data descriptor, 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 ADDRESS

13.4.2. TRANSPORT_OPEN Command

The initiator shall use this command to request the target to open a connection to the target's transport client using the indicated transport parameters supplied in the buffer. The initiator shall use the ORB format shown below.

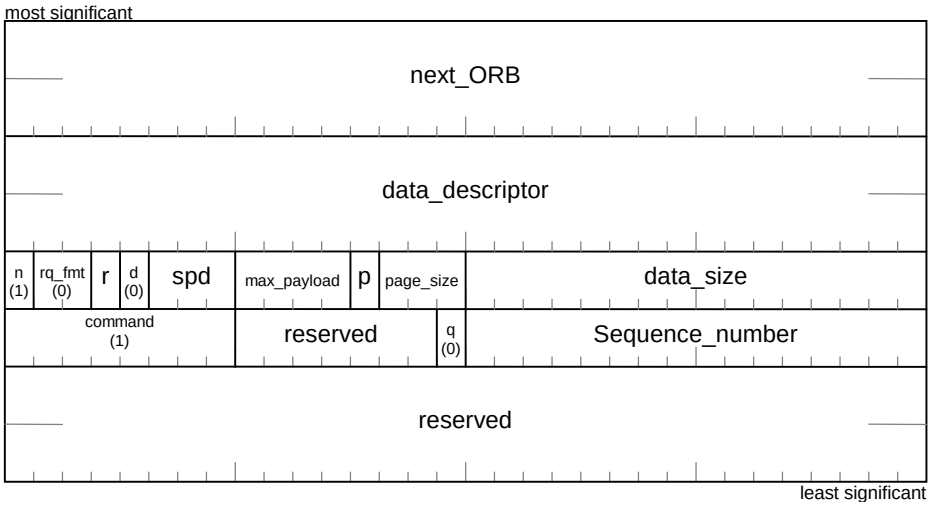


Figure 8 – TRANSPORT_OPEN Command ORB

All fields shall conform to the descriptions in the Command Block ORB Format clause

The *notify* bit shall be one.

The *queue_ID* bit shall be zero.

The *command* field shall indicate a TRANSPORT_OPEN command.

The data associated with this command shall contain encoding(s), as described in 13.1.3 Parameter Encoding, for the parameter(s) to be set. All behaviors regarding parameter encodings are described in 13.1.3 Parameter Encoding.

13.4.2.1.TRANSPORT_OPEN Command Response

The command response shall conform to the format described in 13.1.2 Status Block.

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

The *residual* field is reserved for this command.

If this command completes successfully, then the target shall return status of:

$Resp_{(16)}$	$Sbp\_status_{(16)}$	$Status_{(16)}$	Sense $key_{(16)}$	$ASC_{(16)}$, $ASQ_{(16)}$
0 - REQ COMPLETE	0 - NO ADDL INFO	0 - GOOD	0 - NO SENSE	0,0 – NO ADDITIONAL SENSE TO REPORT

Else if this command contains an invalid field in the CDB, 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	24,0 – INVALID FIELD IN CDB

Else if this command is issued before a TRANSPORT_CAPABILITIES request has succeeded, 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	2C,0 – COMMAND SEQUENCE ERROR

Else if this command contains an invalid parameter encoding, 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	1A,0 - PARAMETER LIST LENGTH ERROR

Else if this command contains an unsupported parameter id, 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	26,1 – PARAMETER NOT SUPPORTED

Else if this command contains an unsupported parameter value, 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	26,2 – PARAMETER VALUE INVALID

Else if this command cannot succeed due to target resource constraints, 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	55,0 – SYSTEM RESOURCE FAILURE

Else if this command is issued after the connection has been open, 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	4A,0 – COMMAND PHASE ERROR

13.4.3. TRANSPORT_I2T_DATA Command

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

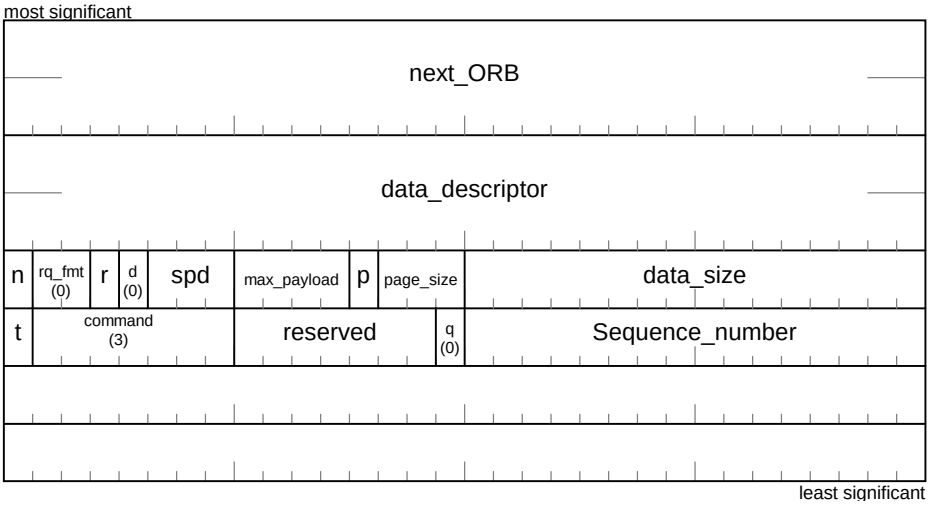


Figure 6 – TRANSPORT_I2T_DATA Command ORB

All fields shall conform to the descriptions in the Command Block ORB Format clause

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

The *queue_ID* bit shall be zero.

The *tag* bit (abbreviated *t* in the figure above) specifies that the data in the buffer pointed to by the data_descriptor field is to be treated separate from the rest of the data stream.

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

13.4.3.1.TRANSPORT_I2T_DATA Command Response

The command response shall conform to the format described in 13.1.2 Status Block.

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

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.

If this command completes successfully, then the target shall return status of:

Resp	Sbp_status	Status	Sense key	ASC, ASQ
0 - REQ COMPLETE	0 - NO ADDL INFO	0 - GOOD	0 - NO SENSE	0,0 – NO ADDITIONAL SENSE TO REPORT

Else if this command contains an invalid field in the CDB, 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	24,0 – INVALID FIELD IN CDB

Else if this command is issued before the connection is open, 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	4A,0 – COMMAND PHASE ERROR

Else if the direction is closed (by either the initiator or target), 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	0,2 – END OF PARTITION/MEDIUM DETECTED

13.4.4. TRANSPORT_T2I_DATA Command

The initiator shall use this command to request the target to TRANSPORT_T2I_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

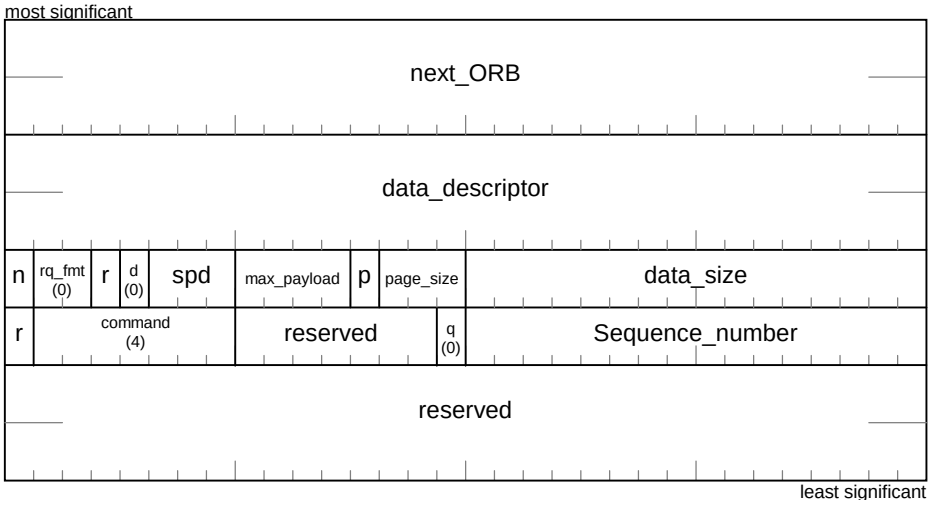


Figure7 – TRANSPORT_T2I_DATA Command ORB

All fields shall conform to the descriptions in the Command Block ORB Format clause

The *queue_ID* bit shall be zero.

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.

13.4.4.1.TRANSPORT_T2I_DATA Command Response

The command response shall conform to the format described in 13.1.2 Status Block.

The *tag* bit shall specify whether or not this data is distinct from the rest of the data stream. If the data block length is zero, this field is reserved.

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 *residual* field shall only indicate the amount of unused space in the buffer provided. The target shall not indicate the presence of additional data. Additional data shall be retained for a successive command. A target should complete the command as soon as data is available.

> 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?)

If this command completes successfully, then the target shall return status of:

Resp	Sbp_status	Status	Sense key	ASC, ASQ
0 - REQ COMPLETE	0 - NO ADDL INFO	0 - GOOD	0 - NO SENSE	0,0 – NO ADDITIONAL SENSE TO REPORT

Else if this command contains an invalid field in the CDB, 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	24,0 – INVALID FIELD IN CDB

Else if this command is issued before the connection is open, 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	4A,0 – COMMAND PHASE ERROR

Else if direction is closed (either by the target or the initiator), 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	0,5 – END-OF-DATA DETECTED

13.4.5. TRANSPORT_CLOSE Command

The initiator shall use this command to indicate to the target that no more commands will be enqueued on the associated queue. The initiator shall use the ORB format shown below.

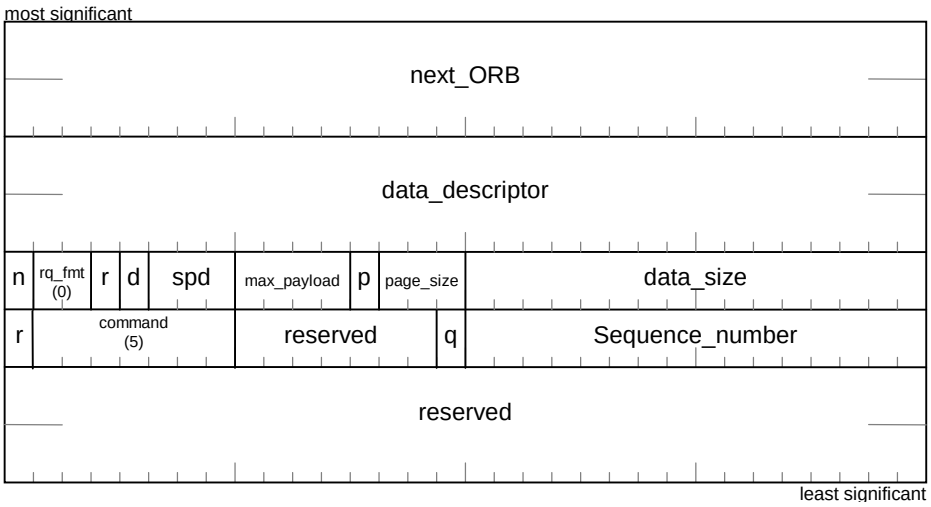


Figure 7 – TRANSPORT_CLOSE Command ORB

All fields shall conform to the descriptions in the Command Block ORB Format clause

13.4.5.1.TRANSPORT_CLOSE Command Response

The command response shall conform to the format described in 13.1.2 Status Block.

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

If this command completes successfully, then the target shall return status of:

Resp	Sbp_status	Status	Sense key	ASC, ASQ
REQ COMPLETE	NO ADDL INFO	GOOD	NO SENSE	0,0 – NO ADDITIONAL SENSE TO REPORT

Else if this command is issued before the connection is open, then the target shall return status of:

Resp	Sbp_status	Status	Sense key	ASC, ASQ
REQ COMPLETE	NO ADDL INFO	CHECK COND	ILLEG REQ	4A,0 – COMMAND PHASE ERROR

13.5. Status FIFO write request recovery

SBP-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 SBP-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 update the ORB_POINTER register to reference an offending ORB.

e) It shall transition the Fetch Agent to the DEAD state.

This allows the initiator to detect the erroneous condition by reading the AGENT_STATE register and recover from it by re-queuing the uncompleted task list.

Alternatively, the condition would automatically be corrected as a consequence of the next Bus Reset event. Upon completion of a successful reconnection, the initiator must restore TRANSPORT_I2T_DATA and TRANSPORT_T2I_DATA commands and associated data to the task set in the order previously queued within the target. No assumptions shall be made about ORB or buffer locations remaining the same within the 1394 shared memory space, or about the same physical buffers being used.

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. It is anticipated to be acceptable for the target's node to generate a Bus Reset when this condition occurs.

13.6. Queue Management

To simplify queue management and error recovery, the ABORT TASK and LOGICAL UNIT RESET management function shall not be supported.

For stream services, aborted ORBs must be requeued in the same order as previously existed, and must retain the sequence numbers previously assigned.

For message services, requeued ORBs must retain the sequence numbers previously assigned, and must be restored to the same queue order as before.

13.7. Sequence Number Management

In order for the target to distinguish between completed and acknowledged, executed but uncompleted, and partially or unexecuted commands within a data stream, each command shall contain a queue identifier and associated sequence number. The sequence number shall be unique to the queue.

The sequence number found in the first command placed on each queue shall be used as the initial sequence number value. At any point in time all data block sequence numbers on a queue must lie within one quarter of the size of the sequence number space. Sequence numbers are assigned in ascending order. Once a sequence number has been assigned to a data block, it may not be reused for a different data block until it falls outside the sliding window of usable sequence numbers. Commands, which do not transfer application data still implicitly have a (possibly empty) data block, and so still require a sequence number.

NOTE – For the following discussion Sna represents the sequence number of the last completed and acknowledged command on this queue. Sne represents the sequence number of the last fully executed, but uncompleted (or unacknowledged) command on this queue. Snl represents the sequence number of the command with a lost Ack on a status FIFO write. Sn represents the sequence number of the command being examined. The function dist(A,B) is evaluated as ((A + sequence_number_range – B) % sequence_number_range) and will produces only positive distances.

Whenever a target begins processing a command, it shall examine the sequence number. If the target is trying to recover from a lost status FIFO write Ack:

a) If (dist(Sn, Snl) >= sequence_number_range / 2) then the command had been successfully executed. If the notification bit is set, the command completion notification is immediately given.

b) Else if (dist(Sn, Snl) == 0) then the command has been requeued. The target shall only (re)send the previous completion notification. Upon receipt of the Ack on the status FIFO write, the target shall be synchronized with the initiator.

c) Else if (dist(Sn, Snl) > 0) then the initiator received the previously attempted (successful) completion notification, and the target is resynchronized with the initiator. The sequence number has been implicitly acknowledged.

When the target is not trying to recover from a lost status FIFO write Ack, it shall proceed as follows:

d) If (dist(Sn, Sna) == 0 || dist(Sn, Sna) >= sequence_number_range / 2), then the command has already been executed. If the notification bit is set, the previous completion notification is immediately given.

e) Else the command is executed.

When an initiator queues a command, it may only do so if:

a) The size of the active task set is less than the size of the Maximum active task set supported by the target

b) And, a unused sequence number is available < (sequence_number_range / 2) distance away from the last acknowledged sequence number.

In this case the initiator shall use the lowest available sequence number.

14. Issues

14.1. Login

Issue: Microsoft has implemented their SBP-2 driver to do a login on power up and not logout until the PC is shut down. Though this is provided for within the SBP-2 specification, it requires devices, which may be shared among multiple PCs to support multiple logins to a single service, even if the different PCs cannot simultaneously use the service.

Can imaging devices (which want to take advantage of the shared nature of 1394) tolerate the resource requirements to operate on a multiple Microsoft O/S host configuration?

Status: Closed, 8/17/98 - Toronto. Microsoft Implementations will not exhibit this behavior for image class devices.

14.2. Service Discovery

Issue: 8/17/98 - Toronto. Transport client command set information?

Status: Open.

14.3. Data packets vs. Data stream communications model?

Issue: 8/17/98 - Toronto. Brian to write up.

Status: Open.

14.4. Function Device Type Number in Unit Directory.

Issue: 8/17/98 - Toronto. Transport client class of service? (Printing vs. Scanning vs. ???)

Status: Open.

14.5. Maximum data size

Issue: Current proposal is 1 Mega byte. Discussion?

Status: Closed, 8/17/98 - Toronto. See command set in this document.

14.6. Unsolicited Status Register Enable

Issue: Need policy for Initiator to always do this ASAP.

Status: Closed, 8/17/98 - Toronto. Have not identified a need for Unsolicited status.

14.7. Target Logout

Issue: How does a Target Logout?

Status: In Progress, 8/17/98 - Toronto. Decided to send completion status to ORB in Target_2_Initiator direction indicating that the Target requests a Logout. If Initiator does not to perform Logout in XX time. Target will reset context for that Initiator effecting a Logout.

14.8. Timers

Issue: Need explanation of behavior at initialization and reset.

1) Reconnect Timer - This one is taken care of by SBP-2 as part of the Login process.

2) Management ORB Timer - When the Initiator writes a Management ORB address to the Management Agent Register, how long should it wait for a response to be written to the management ORBs status fifo? How is the timeout value determined and communicated?

3) Abort Task Set Timer - An Abort Task Set may issued by an Initiator in response to Target Inactivity. How is the timeout value determined and communicated?

Status: Open.

14.9. Plug & Play Support

Issue: Should the discovery section contain information about Plug-N-Play support?

Status: Open.

14.10. Non Symmetric Connection.

Issue: 8/17/98 - Toronto. A single transport_open command requires a two transport_close commands, one for each direction.

Status: Open?

14.11. Transport Capabilities Negotiation Resource Commitment.

Issue: 8/17/98 - Toronto The transport capabilities command requires target to commit resources that may not be immediately followed by a transport_open command. How long should the Target reserve these resources?

Status: Open.

14.12. Notify Status

Issue: In section 9.3 SBP-2 gives us the following:

NOTE - For targets that support the ordered model of task execution, the return of completion status for an ORB implicitly indicates that all preceding ORB's in the linked list have completed successfully, are no longer part of the task set and that the initiator may reuse or de-allocate their system memory.

Since we are using an un-ordered model does the implied status apply to all ORBS like in the ordered model or only to all the preceding ORBs in the same (Read or Write) direction?

Status: Open. Proposal: Only to those in the same direction.

14.13. Target Reset

Issue: One of the areas still requiring clarification is the mapping of transport and target transport clients into SBP-2 Units and Logical Units. This is an area where the SCSI roots of SBP-2 have really impacted our efforts.

Looking at the draft SCSI-3 ARCHITECTURE MODEL -2 (SAM-2) rev

0.3, I find that "Each SCSI-3 protocol standard shall specify the response to a target reset event including the conditions under which a target hard reset shall be executed.

Looking at SBP-2, rev 4, we find TARGET RESET affects all

logged-in initiators (to any logical unit within the Unit).

Some questions which need to be answered are:

a) How is TARGET RESET used by drivers n commercial operating systems? (Both non-embedded and embedded?)

b) Can we define how broadly (or narrowly) the TARGET RESET event affects a device using our transport?

If so, how should it be defined?

If not, does this mean that we must use one Unit directory for each possible instance of a service?

Status: Open.

14.14. Multiple Unit Directories

Issue: Multiple Unit Directories that are the same. How can we identify them to be unique? Example was after a bus reset.

1) What I can't recall is why we have these multiple identical udirs in the first place.

2) Also if we do have them, it seems like they would need to be unique independent of bus reset.

Status: Open.