Working Draft

Revision 0a
May 1, 1998
Author: Greg Shue, HP

IEEE Printer Working Group - 1394 Printing Subcommittee
SBP-2 Command Set for Stream and Message Communications

1. Scope and purpose

1.1 Scope

1.2 Purpose

To provide:
SBP-2 Command Set for PWG Stream/Message communications
SBP-2 Command Set for LUN Server Behavior

2. SBP-2 based Stream/Message Transport Model

Based on SBP-2 Unordered Execution Model. The SBP-2 Unordered Execution Model has the following characteristics:

- Task list is used to communicate command requests to the target and to identify sequential relationships between dependent commands. It does NOT mandate absolute execution order
- Target is responsible for maintaining references to all active tasks enqueued on the task list prior to all completed tasks.

2.1.1 Command Execution Model

For PWG proposal:

- Initiator shall not enqueue tasks on the Task List which cannot be cached by the target or executed immediately, as this would cause target cache overflow.
- Unrecognized commands shall be completed immediately with an unrecognized/unsupported result.
- Task caching characteristics of target must be found in a command-set dependent manner, and are constant within a Login.
- Only transport-level information, tags, and events shall be found in the CDB or command response.
- Both transport-level information and asynchronous application events or status may be communicated in the Unsolicited Event mechanism.

2.2 Unsolicited Status/Event Notification Model

Unsolicited Status shall always be enabled. If Initiator does not re-enable Unsolicited Status within an Initiator-configured interval, the target shall implicitly logout the initiator and release resources. This policy shall take effect as soon as the Unsolicited Status has been enabled for the Login. This parameter is only configurable before Unsolicited Status has been enabled.

Unsolicited Status is used for the following purposes:

- For the Target to detect the non-existence of the Initiator
- For the Target to inidicate a transition to a Data Available for Initiator condition when the initiator does not have at least one command enqueued to receive the data.
- For the Target to actively report an out-of-sync condition resulting from a lost ACK on a write to the Status FIFO.
- For the Target to report any condition which caused the Fetch Agent to transition to the Dead state?

2.3 Messaging/Stream Model

The transport provides a paired set of opposing unidirectional data flows for transferring information. All data is transferred and presented to the destination transport client layer in the order presented to the transport layer. Each particular request has the ability to be tagged by a flag bit to be used at the transport client's discretion. This feature means that very little difference exists between a Message communications model, and a Stream communications model. To facilitate either mechanism, and end-of-message flag is also available for marking data boundaries which should be communicated to the recipient transport client.

Because it is possible for the initiator and target to get out of sync with each other, each command format provides sequence number fields for helping the recipient to screen out duplicate commands or replies. These sequence numbers must be unique across all pending identical commands. Detection of a duplicate sequence number of the last processed command shall cause the command to become benign and immediately succeed. Detection of anything else other than the next number in sequence shall cause the Initiator to be notified of the error and the Execution Agent to stop processing commands until the Task Set has been aborted.

All buffer transfers shall be limited to (2^{32}-1) bytes. Buffer transfers of zero bytes are valid and potentially useful for indicating the end-of-message flag if the status was not previously available. Sending transport clients must present messages in their entirety. Messages which are larger than the maximum supported transfer size for the Login shall be segmented into successive buffers for transfer. The transport layer must fully reassembled the message before presenting any of it to the receiving transport client.

The message vs. Stream model of communication used by the transport client service shall be indicated by unique values for the ‘device_type’ field in the LUN_Characteristics Entry’.

LUN zero shall provide a direct access into the associated service. If multiple types of access are available for the service, or a dynamic LUN-server facility exists to gain access to multiple instances of the service from the same node, then they shall be indicated by other LUN_Characteristics entries in the Unit Directory and shall be distinguished by different ‘device_type’ values.

New entries are required in the Unit Directory for identifying the specific services and software required above the transport layer.

In order to maintain an extensible protocol, yet provide minimal support requirements, transport command set parameters are accessed individually and limited to 64-bits.

2.4 Target-initiated Logout

In order to provide true peer-to-peer type of service, it is necessary for the target to have the ability to logout an initiator at any time. Since this functionality is not described in SBP-2, it must be clarified here. Though it is possible for the target to simply drop the login, all other spontaneous disconnects are associated with well-recognized events (failed reconnect, reported errors, unsolicited status write enable timeout, invalid parameter). To verifiability of interactions, this paradigm shall be used also for target-initiated logout of the initiator.

When the target needs to logout the initiator, it shall:

- attempt to give an unsolicited event notification indicating the asynchronous logout.
- If the notification was successful, it shall then:
  - Disable the Unsolicited status write enable timeout
  - attempt to complete all queued ORBs with a *resp* equal to zero, REQUEST COMPLETE, and an sbp_status value equal to 10, Login ID not recognized.
- It shall then release all resources associated with the login.

If a Bus Reset occurs prior to completing the unsolicited event notification, the target must wait for either the *reconnect_hold* time to expire (so that it may drop the login), or the initiator to validly reconnect to the target and re-enable the Unsolicited status notification. The target shall not simply drop the login without sending notification across the bus.

Question: What do we do if the target wants to logout the initiator before the Unsolicited Status enable has taken place? Currently there is no timeout on how quickly that must be enabled, just a requirement that it must.

2.5 Single Login vs. Cross Login

2.6 LUN-server

SBP-2 inherently limits an initiator to one login per Logical Unit. In order to provide a way around the restriction, each Unit shall provide only one service, each login shall provide an independent session to the service, and a LUN server function shall reserve sessions and dynamically provide associated LUNs to be used to gain access. The sessions and LUNs are reserved only for the duration of the login to the LUN server. The LUN server is restricted to having an exclusive access interface.

Devices shall provide no more than one explicit LUN for each type of enumerated transport provided to the service. If more than one instance is to be provided to a single initiating node, then it must be made available only through the LUN server.

3. Command Set

The command set is shown in Table 1. Each command is mandatory.

Command name	Opcode (eight bits)	Reference
Reserved	**0x00**	
SEND_BUFFER	**0x01**	
GET_BUFFER	**0x02**	
SET_PARAMETER	**0x03**	
GET_PARAMETER	**0x04**	
CROSS_LOGIN	**0x06**	
Reserved	**0x07 - 0xFE**	
Reserved	**0xFF**	

3.1 SEND_BUFFER (Single-login only)

SEND_BUFFER transfers exactly one buffer from the initiator to the target.

Table ?? - SEND_BUFFER Command

Bit Byte	7	6	5	4	3	2	1	0
0	SEND_BUFFER (0x01)							
1	*send_buffer_seq_number*							
2	*tag*	*em*	*am*	r	r	r	r	r
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							

The *tag* field indicates a flag which applies to all the information within the buffer. The interpretation of this flag is beyond the scope of this document.

The *em* field (when set to one) indicates an end-of-message boundary following associated data. When the field is set to a value of zero, it indicates a continuation of a buffer within the following SEND_BUFFER command.

The *am* field indicates an abort message request by the sender. This causes the receiver to abort and discard this and all previous parts of the associated message.

The *send_buffer_seq_number* field carries the sequence number of successive SEND_BUFFER commands. The first SEND_BUFFER command sent following a Login shall have a sequence number of value zero. Each successive command shall cause the sequence number to be incremented by one.

The byte length of the buffer being sent is carried implicitly within the ORB holding this command.

The SEND_BUFFER response carries- the completion status of the indicated command.

Table 6 - SEND_BUFFER Response

Bit / Byte	7	6	5	4	3	2	1	0
0	*send_cmd_status*							
1	*send_buffer_seq_number*							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							
12	Reserved							

The *cmd_status* field indicates the results of the executed command. The following table indicates the supported results of this command

Value	cmd_status	Description
0x00	**Fully Completed**	
0x01	**Unexpected Sequence Number**	
0x03 - 0xFD	**Reserved**	
0xFE	**Unsupported Command**	
0xFF	**Invalid Command**	

The *send_buffer_seq_number* field shall contain the sequence number from the associated command.

3.2 GET_BUFFER

The GET_BUFFER transfers one buffer from the device server (target) to the application client (initiator). The response indicates the actual amount of data transferred, whether this information is tagged as out-of-band, and whether the end of this buffer represents a message boundary.

Table ?? - GET_BUFFER Command

Bit / Byte	7	6	5	4	3	2	1	0
0	GET_BUFFER (0x02)							
1	*get_buffer_seq_number*							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							
12	Reserved							

The *get_seq_number* field carries the sequence number of successive GET_BUFFER commands. The first GET_BUFFER command sent following a Login shall have a sequence number of value zero. Each successive command shall cause the sequence number to be incremented by one.

The byte length of the buffer space being provided is carried implicitly within the ORB holding this command.

The GET_BUFFER response (see Table ??) indicates the results of the data transfer from the target to the initiator.

Table ?? – GET_BUFFER response

Bit / Byte	7	6	5	4	3	2	1	0
0	*get_buffdr_cmd_status*							
1	*get_buffer_seq_number*							
2	*tag*	*em*	*am*	r	r	r	r	*da*
3	Reserved							
2	(MSB)							
3				*buffer_length* (32 bits)				
6								
7								(LSB)
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							

The *da* field indicates whether or not the target has additional data available without a queued GET_BUFFER command for receiving it. This field indicates the status at the time of the completion of the command. If the target transitions to a data available state without a GET_BUFFER command queued, then this indication will be given using the Unsolicited Status mechanism.

The *tag* field indicates a flag which applies to all the information within the buffer. The interpretation of this flag is beyond the scope of this document.

The *em* field indicates an end-of-message event which follows any valid data within the associated buffer when the field is set to a value of one. When the field is set to a value of zero, it indicates a continuation of a buffer within the following GET_BUFFER command.

The *am* field indicates an abort message request by the sender. This causes the receiver to abort and discard this and all previous parts of the associated message.

The *get_buffer_seq_number* field shall contain the sequence number from the associated GET_BUFFER command.

The *buffer_length* field indicates the valid byte-length of the associated buffer.

3.3 GET_PARAMETER

The GET_PARAMETER mechanism is used by the initiator to retrieve the specific current value of a parameter of the target (e.g. number of SEND_BUFFER commands that may be simultaneously queued).

Table 6 - GET_PARAMETER Command

Bit / Byte	7	6	5	4	3	2	1	0
0	GET_PARAMETER (0x03)							
1	*Get_param_seq_number*							
2	(MSB)			*parameter_id*				
3								(LSB)
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							

Table 6 – GET_PARAMETER Response

Bit / Byte	7	6	5	4	3	2	1	0
0	*Get_param_cmd_status*							
1	*get_param_seq_number*							
2	*Result*							
3	Reserved							
4	(MSB)							
5								
6				current Parameter Value				
7								
8								
9								
10								
11								(LSB)

Possible response codes incclude invalid message configuration size; remote service unavailable;

3.4 SET_PARAMETER

The SET_PARAMETER mechanism is used by the initiator to retrieve the specific current value of a parameter of the target (e.g. number of SEND_BUFFER commands which may be simultaneously queued).

Table 6 – GET_PARAMETER command

Bit / Byte	7	6	5	4	3	2	1	0
0	SET_PARAMETER (0x04)							
1	*set_param_seq_number*							
2	(MSB)			*parameter_id*				
3								(LSB)
4	(MSB)							
5								
6				Desired Parameter Value				
7								
8								
9								
10								
11								(LSB)

The SET_PARAMETER response message indicates the result of the connect request.

Table 6 – SET_PARAMETER Response

Bit Byte	7	6	5	4	3	2	1	0
0	*set_param_cmd_status*							
1	*set_param_seq_number*							
2	*Result*							
3	Reserved							
4	(MSB)							
5				New Value				
6								
7								
8								
9								
10								
11								(LSB)

??

3.5 GET_/SET_CONFIGURATION parameters

Parameter	Acces s	Size (bits)	Units	Default Value	Powerup value	Support
SEND_BUFFER cmd queue size	RO	6	cmds	N/A	N/A	Mandatory
GET_BUFFER cmd queue size	RO	6	cmds	N/A	N/A	Mandatory
Max SEND_BUFFER data block size	RO	32	bytes	N/A	N/A	Mandatory
Max GET_BUFFER data block size	RO	32	bytes	N/A	N/A	Mandatory
Unsolicited Status Enable write timeout	RW	8	0.1 sec	0 (invalid)	N/A	Mandatory

3.6 Unsolicited Status Indication

Table 6 - Unsolicited Status Indication

Bit / Byte	7	6	5	4	3	2	1	0
0	*da*	*ping*	*Lost_ ack*	Async logout	r	r	r	r
1	*Unsolicited_status_seq_number*							
2	Reserved							
3	Reserved							

The *da* bit indicates that the target has transitioned from having no data available for the initiator to having data available. The target must subsequently complete a GET_BUFFER command indicating the amount of data available even if the data transfer was cancelled within the target. If no data is available when the GET_BUFFER command was processed, then the buffer shall be complete with zero bytes transferred.

The *ping* bit is used by the target to verify the responsiveness of the initiator when it cannot infer this from other activity.

The *lost_ack* bit is used by the target to indicate as early as possible that the target may be out of synch with the initiator because the response to a write transaction was corrupted. (Either the ack was lost or ??).

This can happen on normal commands:

The target shall …

This can happen on Unsolicited Status transaction:

Target waits for Ustatus write enable timeout to expire. If the initiator issues a Ustatus write enable before the timeout expires, the target shall assume that it lost the ack on the transaction and shall resume normal operation.

If the timeout expires without the status being written, then the target shall assume that the transfer was corrupted and ignored by the initiator, and the target shall retry the transfer.

The Async_logout bit indicates that the target has asynchronously logged out the initiator. (Single-Login only)

3.7 Cross-Login Request (Cross-login only)

LOGIN_REQUEST transfers exactly one buffer from the initiator to the target.

Table ?? – LOGIN_REQUEST Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	LOGN_REQUEST (0x??)							
1	*Login_req_seq_number*							
2	(MSB)	Management Agent Offset w/in requesting node						
3								
4								
5								
6								
7								(LSB)
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							

The *send_buffer_seq_number* field carries the sequence number of successive SEND_BUFFER commands. The first SEND_BUFFER command sent following a Login shall have a sequence number of value zero. Each successive command shall cause the sequence number to be incremented by one.

The LOGIIN_REQ response carries- the completion status of the indicated command.

Table 6 – LOGIN_REQ Response

Bit / Byte	7	6	5	4	3	2	1	0
0	*Login_Req_cmd_status*							
1	*Login_req_seq_number*							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							
12	Reserved							

The *cmd_status* field indicates the results of the executed command. The following table indicates the supported results of this command

Value	cmd_status	Description
0x00	**Fully Completed**	
0x01	**Unexpected Sequence Number**	
0x03 - 0xFD	**Reserved**	
0xFE	**Unsupported Command**	
0xFF	**Invalid Command**	

The *send_buffer_seq_number* field shall contain the sequence number from the associated command.

4. Control and Status Registers

The Control and Status Registers (CSRs) implemented by a target shall conform to the requirements defined by SBP-2 and it's normative references.

5. Configuration ROM

The configuration ROM implemented by a target shall conform to the requirements defined by SBP-2 and it's normative references. A target shall implement entries defined by SBP-2 as described in the clauses that follow.

5.1 Unit Architecture

Because this document describes a generic communications service, the relationship of Unit Architecture and the associated type of service must be described. In order for the system to scale efficiently, the following requirements exist for this standard:

- A Unit Directory (and all subsequent branches) shall all be built upon SBP-2
- LUN 0 (zero) shall be directly mapped to the default service stack provided above SBP-2
- An initiator is not required to decode Logical Unit Directories, since they are optional. As a consequence, Command_Set_Spec_ID, Command_Set, and Command_Set_Revision entries shall be provided in the Unit Directory which describe the command set associated with LUN 0 (zero). Similarly, the Logical_Unit_Characteristics and Logical_Unit_Number entry which describe LUN 0 (zero) shall be provided in the Unit Directory
- For organizational clarity, all other Logical Units shall be described by Logical Unit Directories referenced from the Unit Directory. The Logical Unit Directories shall contain Command_Set_Spec_ID, Command_Set, Command_Set_Revision, Logical_Unit_Characteristics, and Logical_Unit_Number entries.
- The Command_Set entry shall only specify the SBP-2 command set used to provide transport functionality. Other entries are needed for subsequent layers. These include:

    - Protocol_Layer_Directory entry
    - Protocol_Layer_Command_Set_Spec_ID
    - Protocol_Layer_Command_Set
    - Protocol_Layer_Command_Set_Revision
    - other (Protocol_Layer_Command_Set_Spec_ID, Protocol_Layer_Command_Set) dependent entries.

- Textual_Descriptor entries shall immediately follow each of the listed entries in the Config ROM:
    - Module_Vendor_ID
    - Node_Unique_ID
    - ??

5.2 Command_Set_Spec_ID entry

The Command_Set_Spec_ID entry is an immediate entry formally defined in SBP-2. This specifies the organization responsible for the command set (CDB field) definition for the target. The contents shall be the value assigned by the IEEE/RAC to the organization responsible for this document.

5.3 Command_Set entry

The Command_Set entry is an immediate entry formally defined in SBP-2. This in combination with the *command_set_spec_id* specifies the command set implemented by the target. The value of *command_set* shall be specified by the owner of *command_set_spec_id* and shall indicate that the target supports the command set described in this document.

5.4 Command_Set_Revision

The Command_Set_Revision entry is an immediate entry formally defined in SBP-2. This specifies the revision level of the command set implemented by the target. The value of *command_set_revision* shall be specified by the owner of *command_set_spec_id*.

5.5 Logical_Unit_Number

The Logical_Unit_Number entry is an immediate entry formally defined in SBP-2. This specifies the “characteristics, peripheral device type and logical unit number” of a logical unit implemented by the target. For this command set, the following requirements exist:

The Logical_Unit_Number entry for LUN 0 (zero) shall be found in the Unit Directory. It shall be associated with a connection tied directly to the service provided by the unit. All other Logical_Unit_Number entries shall be located in Logcial Unit directories.

As defined in SBP-2, the *device_type* field indicates the peripheral device type implemented by the logical unit. The value is either command-set dependent or a global flag indicating that command-set dependent means are necessary to determine the peripheral device type. Since this document describes a command set which provides generic transport services, it is impossible to identify anything other than the type of transport service provided by this login. The possible types of services include:

Device _type	Ordered bit	Transport service type
0x00	**Zero**	**Message-style communications – Single Login**
0x01	**Zero**	**Stream-style communications – Single Login**
0x02	**Zero**	**Thin Session-equivalent services – Single Login**
0x03 – 0x0E	**Unspecified**	**Reserved – Single Login type of service**
0x0F	**One**	**LUN-server (provides ability for multiple logins from same initiator)**
0x10	**One**	**Message-style communications – Cross Login**
0x11	**One**	**Stream-style communications – Cross Login**
0x12	**One**	**Thin Session-equivalent services – Cross Login**
0x13 – 0x1E	**Unspecified**	**Reserved – Cross Login type of service**
0x1F	**Unspecified**	**As defined by SBP-2**