1394 PRINTER WORKING GROUP

IEEE 1394 HIGH SPEED BUS

IMAGING DEVICE

COMMUNICATIONS SPECIFICATION

***PRELIMINARY DRAFT PROPOSAL ***

Revision 0.40 - June 5, 1998

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

Contents

Figure

1 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 a device profile, which specifies:

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.

2 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.

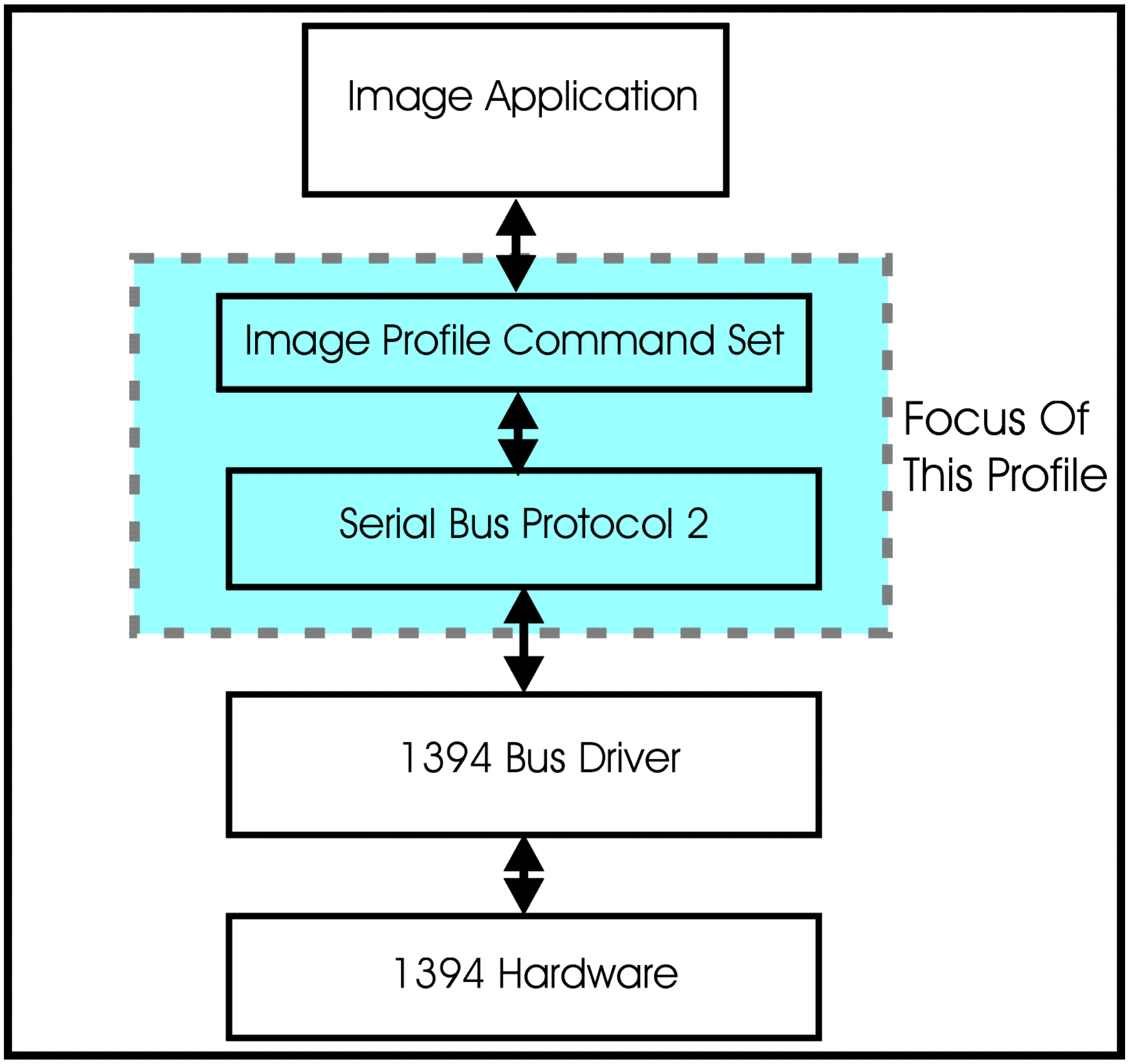


Figure 1 - Scope Block Diagram

3 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).

4 Definitions

4.1 Function:

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

4.2 Queue:

A collection of ORBS that proceed or block independent of other queues

4.3 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.

4.4 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 Bit, Byte and Quadlet ordering

See IEEE std. 1394-1995.

6 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

7 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.

The block diagram below illustrates device Configuration ROM relationships. Additional directories are defined in following sections.

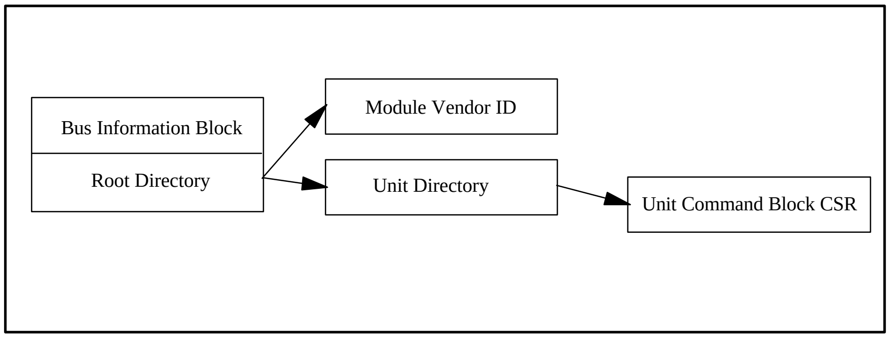


Figure 2 - Config ROM Block Diagram

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.

8 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.

9 ORB List Processing

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.

Each device is modeled as providing at least one image source or image sink service. A device may also provide a status service, a device control service, and zero or more applications which can use these same types of services on other devices.

Every command shall have the ORB notify bit set to one. The data transfer commands shall not be considered complete until successful completion notification has been given. This is necessary to avoid duplication of the commands in the data stream.

The initiator shall close a connection by an explicit logout command to the target's Management_Agent register. The initiator shall not retain access to the device when it is not actively using the resource.

Re-connect Timeout - 1394 PWG devices shall support the extended reconnect timeout mechanism defined by SBP-2.

Maximum Data Size per direction determines the largest amount data that may be transferred in each direction. A data_desciptor buffer shall be limited to the Maximum Data Size negotiated for a particular direction.

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

10 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.

11 Bi-Directional Communication Model

The initiator may also 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.

11.1 Target Model (Informative)

Figure 3 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.

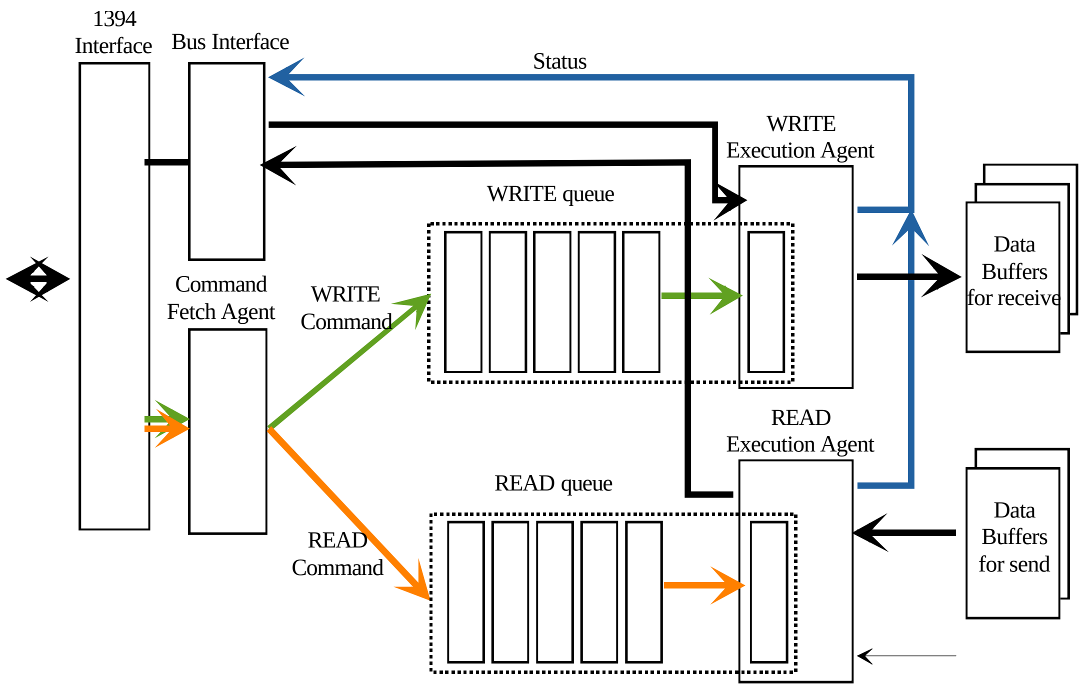


Figure 3 - Target Model

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.

Once an ORB has been fetched and placed in the pre-fetch queue, fetch agents (and execution agents) shall not refer to Initiator memory addressed by the *next_ORB* field.

NOTE – 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.

11.2 Initiator Model (Informative)

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

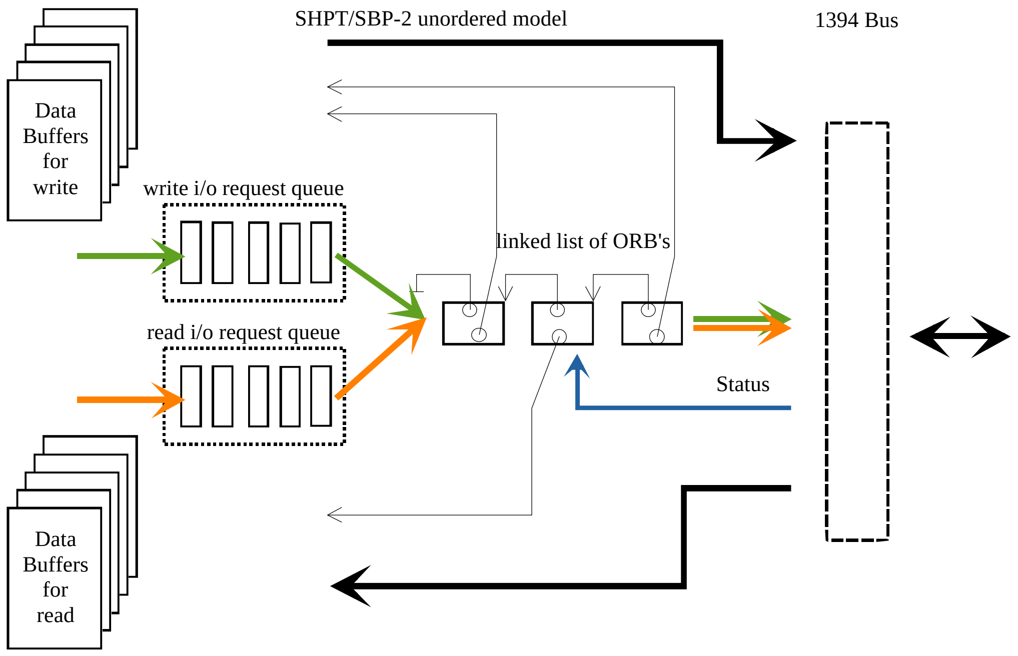


Figure 4 - Initiator Model

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 form the appropriate queue according to the command.

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.

11.3 Error Recovery

Avoid duplicate data

Avoid data loss

12 Multiple Host and/or Multiple Device

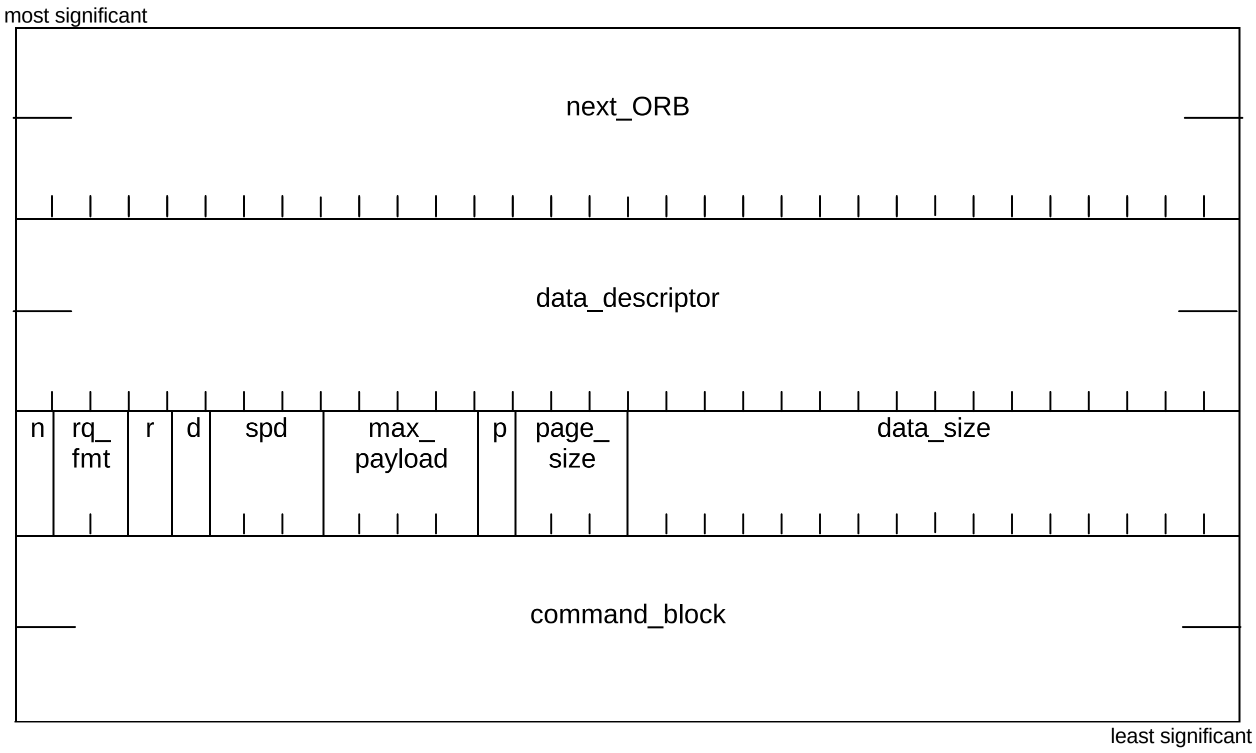
There is a need to provide fair access to devices on a 1394 bus. Each node is accessible from any of the other nodes on the bus and possibly nodes outside of the bus in a bridged environment. It is reasonable to expect that more than one initiator node may attempt to communicate with a target service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

13 Command Block ORBs

All devices that conform to this communications profile shall support the Normal Command Block ORB. The *command_block* field of the Normal Command Block ORB shall be exactly 12 bytes.. The format of the ORB shall follow the SBP-2 specification.

Command Block ORB



14 Management ORBs

All information within this section of the SBP-2 specification apply.

1394 asynchronous imaging devices shall implement the following Management ORB functions:

Table 12 - Management ORB Functions Support List

Function Value$_{16}$	**Management Function**	**Support Level**
0	Login	Required full support
1	Query Login	Required full support
2	Create Stream	Understood - unsupported functionality
3	Reconnect	Required full support
4	Set Password	Understood - unsupported functionality
5 - 6	Reserved	Required full support
7	Logout	Required full support
8 - 9	Reserved	Required full support
A	Not Supported	Required full support
B	Abort Task	Required full support
C	Abort Task Set	Required full support
D	Clear Task Set	Understood - unsupported functionality
E	Logical Unit Reset	Required full support
F	Target Reset	Required full support

15 Negotiated Maximum Task Data Payload Size

Because a Bus Reset implicitly aborts all tasks in the task list(s), and the data payloads are not idempotent (they hold state information), data block transfers between initiator and target must be completed and verified before being presented to the applications. This requires data payload sizes be limited to the buffering available on each end for each direction of data transfer. absolute buffer sizes shall be negotiated in units of 128 bytes.. The maximum byte size supported shall be limited to 2^{31} bytes.. Sizes must be negotiated per direction.

16 Login & Login Response

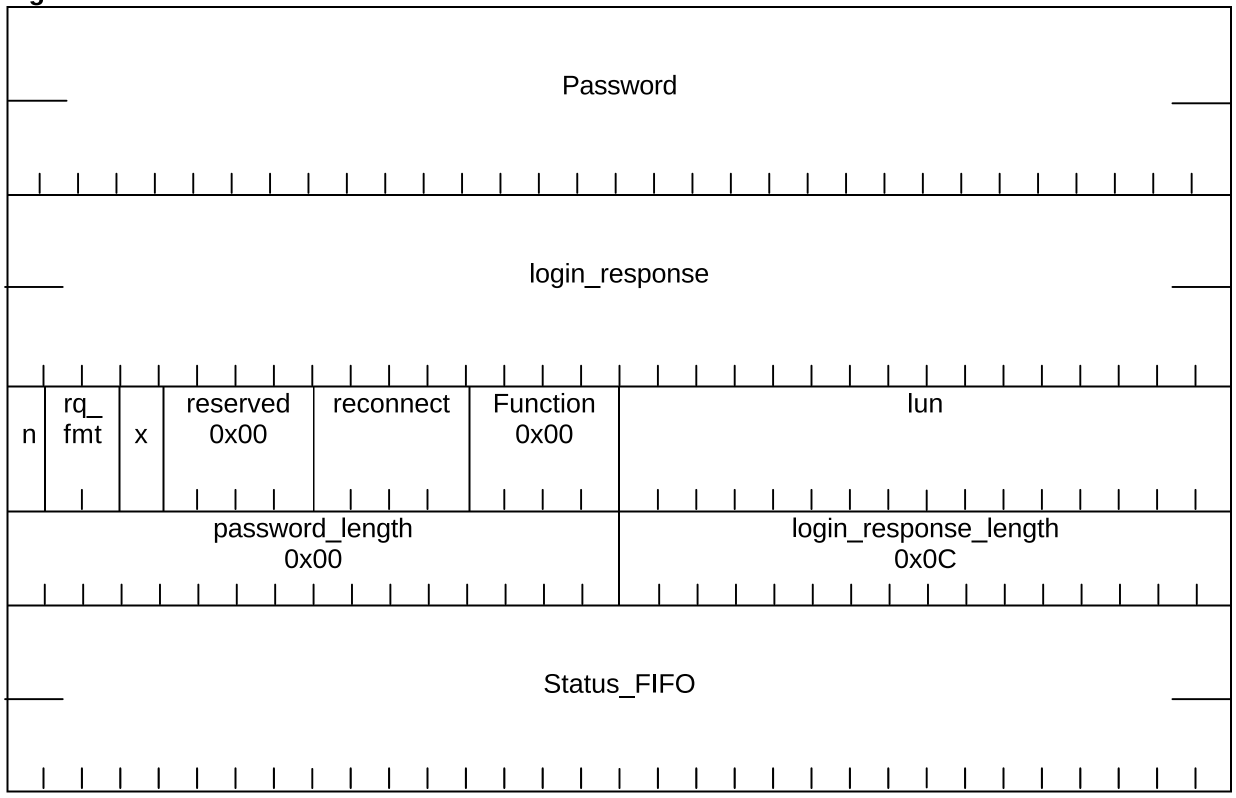
This section will specify the details of the Login Process.

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

16.1 The Login Process -

1) The Initiator will discover a device by reading the CSR & Configuration ROM space of all devices on the 1394 bus.
2) The Management_Agent_Register is also discovered at this time.
3) The Initiator will record the Target's GUID.

Login ORB



4) The Initiator will build the Login ORB with or without a password.
5) The login_response address is the temporary memory address that the Target will use to send its response to the Login.
6) The Status_FIFO address will remain static during the life of the Login.
7) The Initiator will write the address of the Login ORB to the Target's Management_Agent register. [Target shall monitor writes to this address or set up an Interrupt]
8) The 1394 speed (s100, s200, s400) of the write to the Management Agent register will determine the speed used for communication.
9) The Target will read the Login ORB.
10) The Target will read the Initiator's bus information block to discover the GUID.

11) The Target will validate this new Login by comparing the GUID against current login_descriptors. If this Initiator is already logged in the Login shall be rejected. If the Target only supports one Login and another device is logged in, the Login shall be rejected.

12) The Target will build a login_descriptor data structure that will be associated with this specific login.
13) The Target will store the Initiators GUID in the login_descriptor login_owner field.

Login Response ORB

Length 16	login_ID
command_block_agent 0xFFFF F001 0008 (example)	
	reconnect_hold

14) The Target will build the Login Response ORB and fill in the login_ID. The login_ID is like a connection identifier that is unique across active Logins.
15) The Target will store the login_ID in the login_descriptor.
16) The command_block_agent address points to the Unit Command_Block_Agent CSRs in the Configuration ROM.
17) Finally the Target writes the Login Response ORB to the login_response address.

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.

17 Logout

17.1 Initiator-Explicit Logout

Upon an explicit logout by the initiator, access shall be released immediately.

18 Unsolicited Status

As of the May 1998 PWG meeting a need for Unsolicited Status has not been identified. The remainder of this section has been left in for reference.

All unsolicited status generated by this specification shall reflect communication status only. No device-specific or service-specific information shall be carried in the unsolicited status message. The unsolicited status message shall be identified by setting the *src* field to a value of two (indicating the source is “device” status). When an unsolicited status block is stored as a result of initiator inactivity, the status structure shall indicate the nature of the request, and shall also indicate the current state of transport status.

1. The Target shall clear it’s Unsolicited_Status_Enable register after a successful Login.
2. The Initiator shall write a one to the Target’s Unsolicited_Status_Enable to allow Unsolicited Status.
3. The Target may send Unsolicited status, using a Status Block, to the Initiator using the Status_FIFO address that the Target received during Login.
4. The Target shall use its Unsolicited_Status_Enable register to handshake this status block.
5. The Target can only store status when the Unsolicited_Status_Enable register is set.
6. After writing the status, the Target shall clear this register.
7. The Initiator shall write a one to the Target’s Unsolicited_Status_Enable to allow subsequent Unsolicited Status.

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.

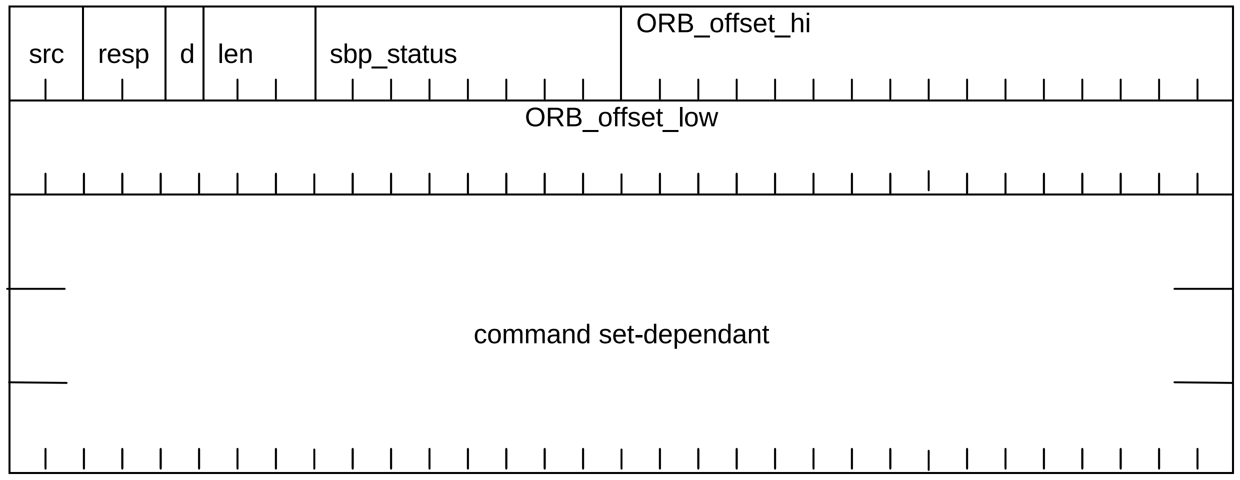
19 Status Block

The status block shall be a maximum of 5 quadlets in length. If no exception status is generated, only the first two quadlets shall be written to the initiator's Status_FIFO address.

The notify bit shall be set for all ORBs queued on the request list. If the target shall detect that the notify bit is not set on a task in the task list, then it shall abort the request with a *resp* value of ILLEGAL REQUEST. The *sbp_status* field shall be set to FF_{16}, as required by the SBP-2 protocol.. The fetch agent shall abort all tasks in the task list and shall transition to the DEAD state.

The first two quadlets are fully defined by the SBP-2 specification. If status is sent to the Status_FIFO in response to a management ORB the ORB_offset fields will contain the appropriate address. ORB_offsets for Unsolicited Status will be set to zero. The fields in the remaining quadlets and their values specific to this profile will be described in this profile.

Status Block



Implementations are not required to use all of the status information specified in the tables that follow. Could add information that states which codes are recommended in an appendix ?

src field

Value	Description
0	The status block pertains to an ORB identified by ORB_offset; at the time the ORB was most recently fetched by the target the next_ORB field did not contain a null pointer.
1	The status block pertains to an ORB identified by ORB_offset; at the time the ORB was most recently fetched by the target the next_ORB field was null.
2	The status block is unsolicited and contains device status information; the contents of the ORB_offset field shall be ignored.
3	The status block is unsolicited and contains Isochronous error report information as specified by 12.3.

resp field

Value	Name	Description
0	REQUEST COMPLETE	The request completed without transport protocol error (Either sbp_status or command set-dependent status information may indicate the success or failure of the request)
1	TRANSPORT FAILURE	The target detected a nonrecoverable transport failure that prevented the completion of the request
2	ILLEGAL REQUEST	There is an unsupported field or bit value in the ORB; the sbp_status field may provide additional information
3	VENDOR DEPENDENT	The meaning of sbp_status shall be specified by this specification

sbp_status field

Value	Description
0	No additional sense to report
1	Request type not supported
2	Speed not supported
3	Page size not supported
4	Access denied
5	Logical unit not supported
6	Maximum payload too small
7	Too many channels
8	Resources unavailable
9	Function rejected
10	Login ID not recognized
11	Dummy ORB completed
12	Request aborted
0xFF	Unspecified error

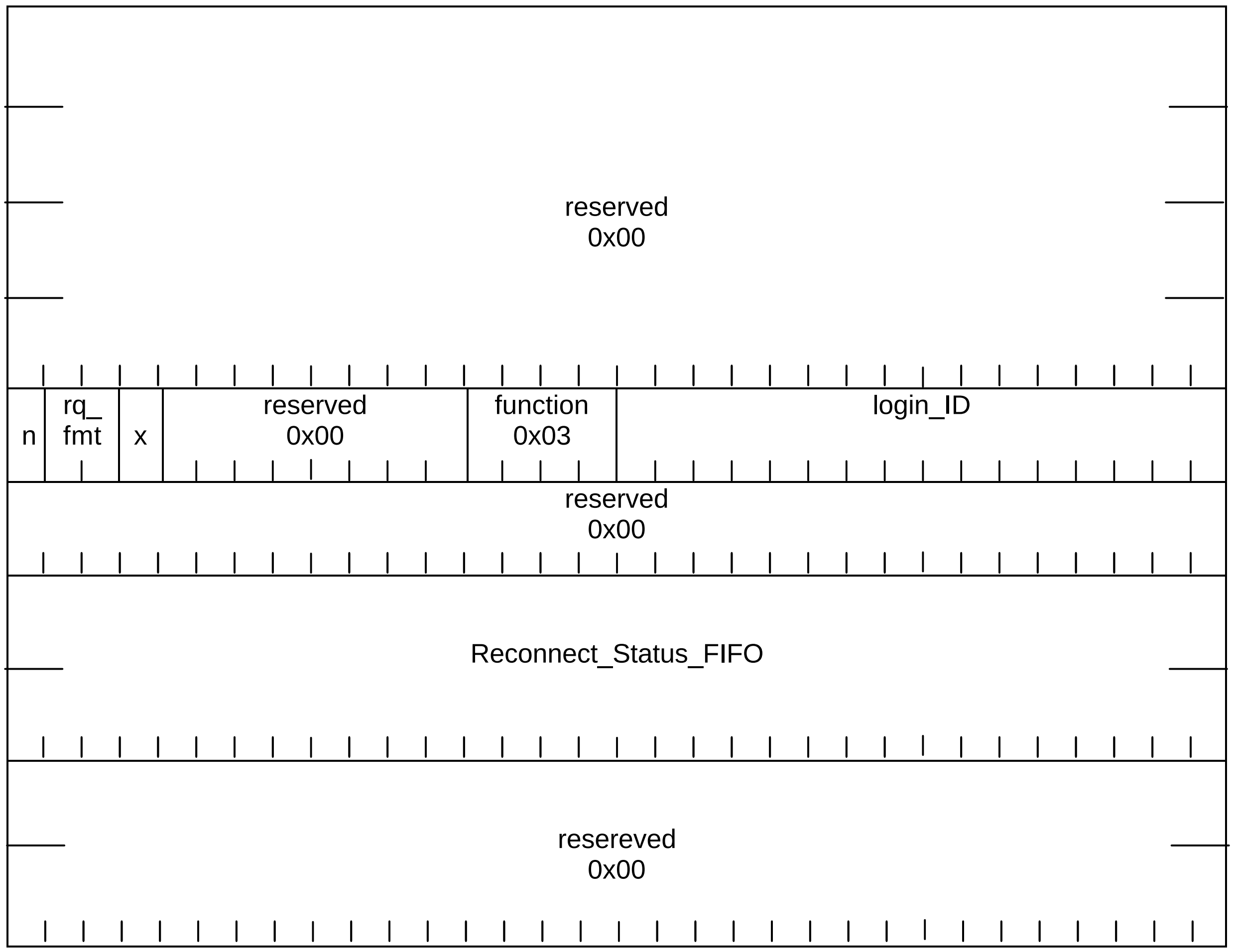
19.1 Command Set-Dependent Status

The command-dependent field meanings shall be specified by each command.

20 Reconnection

Reconnection after a Bus Reset will be accomplished using the Reconnect ORB.

Reconnect ORB



1. After a bus reset, the Initiator is required to re-discover the Target that it was Logged into by reading the Bus Information Blocks of nodes on the bus searching for a matching GUID.
2. After a bus reset, if the Target was connected by a Login the, Target will start a timer. The SBP-2 specification suggests 2 seconds, we may want to tune this value.
3. Once the Target is found the Initiator can write the address of the reconnect ORB to the Target's Management_Agent register.
4. The 1394 speed (s100, s200, s400) of this write will determine the speed used for communication.
5. The Initiator shall use the same login_ID that the Target provided at Login. The Target will fetch the reconnect ORB and read the Initiators Bus Information Block to verify that the Initiators GUID match the GUID established at Login.
6. The reconnect is completed when the Target writes status to the Reconnect_Status_FIFO. Note that this is a new separate Reconnect_Status_FIFO, which is not the same Status_FIFO established at Login.

7. After reconnect the 48 bits of the Login Status_FIFO is used for unsolicited status.
8. The Login Status_FIFO address may have to be patched with the Initiators new node number and bus number.
9. The Data_FIFO_Addresses may also have to be patched with the new Node number and Bus number.
10. If the Target's timer expires before a reconnect ORB is provided the Target will perform an automatic Logout. Logout consists of resetting the login descriptor variables to their initial values. The Unit Command_Block_Agent CSRs should also be reset to initial values.

A special case occurs when an active Login exists between an Initiator and a Target if the Initiator is power cycled independent of the Target. After the bus reset the Target is expecting a Reconnect and the Initiator will attempt a new Login. The Target shall refuse the Login if it happens before the Targets timer has expired. Initiators shall retry the Login after waiting a reconnect timeout period.

21 Query Login

22 1394 Bus Reset Behavior

Upon a Bus Reset, the reconnect timeout shall be reset and restarted to allow a window for the initiator to re-establish the connection.

22.1 After a Bus Reset:

During idle state - no data transactions pending

Reconnect as described in the preceding section.

During Asynchronous data transmission

Reconnect as described in the preceding section and continue data phase.

During Isochronous data transmission

Continue with Isochronous data transmission.

23 Issues

24 Login

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?

24.1 Unit Directory

How to encode the following information in the Unit Architecture and Directory Structure?

Transport client command set information?

Data packets vs. Data stream communications model?

transport client class of service? (Printing vs. Scanning vs. ???)

24.2 Maximum data size

Current proposal is 1 Mega byte. Discussion?

24.3 Unsolicited Status Register Enable

Need policy for Initiator to always do this ASAP.

25 Target Logout

How does a Target Logout?

26 Timers

Need explanation of behavior at initialization and reset

27 Plug & Play Support

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

Appendix A. SBP-2 Communication Protocol (Informative)

This section is provided to get a general idea of how SBP-2 works. SBP-2 defines a method to exchange commands, data, and status between devices connected to the Serial Bus.

The terms Initiator and Target have a specific meaning derived from SBP-2 and do not imply the direction of data transfer.

Initiator: Originates management & command functions such as Login, data transfer and Reconnect. Provides the shared memory space associated with the management & command functions.

Target: Responds to management & command functions and generates Unsolicited status.

SBP-2 requires that an Initiator login to a Target to begin communication. The basic building blocks of SBP-2 include Operation Request Block (ORB) data structures. The two main types of ORBs are the Command Block ORB and the Management ORB. SBP-2 describes the services that operate on these two types of ORBs as agents.

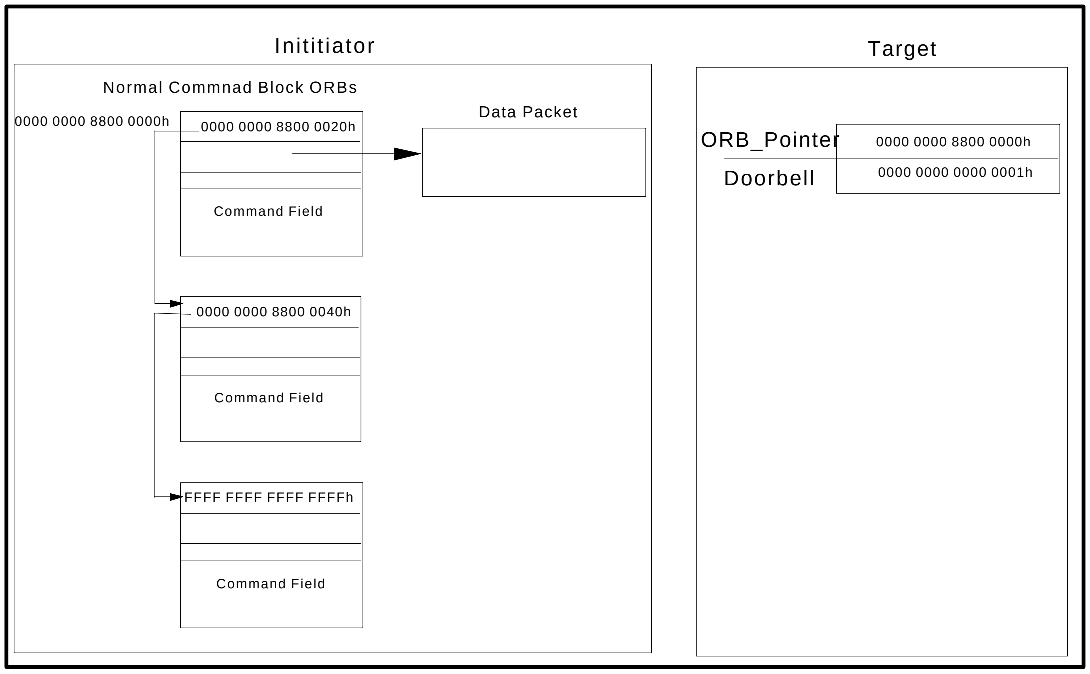
After power-on or bus reset, the Command_Agent and Management_Agent engines are in the Reset state.

The initiator reads the device's Configuration ROM data in order to determine 1394 capabilities, SBP-2 capabilities, EUI-64 (GUID) value, command set identifiers, software versions, and Management_Agent CSR address.

The initiator performs a Login operation prior to any request to the fetch agent interface. To perform a Login, the initiator writes its Login ORB address to the Management_Agent register.

The device returns the Login response to the bus address specified in the Login ORB. One field of the Login response contains the Command_Block_Agent CSR base address.

Prior to initiating command transfers, the initiator builds a list of Command_Block ORBs in system memory. The list may be as short as one ORB, but this example assumes a list length of more than one. The last ORB in the list contains a NULL Next_ORB pointer, which indicates the end of the list to the device's Command_Agent fetch engine. A NULL address has the n bit (most significant bit) set to a one.



To transition the Command_Agent state from Reset to Active the initiator writes the offset of the first ORB in the ORB list to the device's ORB_Pointer. The ORB_Pointer was discovered through the Command_Block_Agent CSR. This allows the Command_Agent fetch engine to begin fetching ORBs from initiator memory. If the initiator writes to the Doorbell CSR, the device will ignore the Doorbell at this time.

The device may optionally fetch ORBs until its ORB space is full or until an ORB containing a NULL Next_ORB pointer is fetched. Fetched ORBs are routed to the Execution engine. The Execution engine may reorder the execution of the commands contained in the ORBs as long as it can guarantee device data integrity.

The Direction bit (d) in the ORB determines the direction of data transfer from the Target's point of view. If the direction bit is zero the Target will use serial bus read transactions to fetch the data from the Initiator. If the direction bit is one the Target will use serial bus write transactions to transfer data to the Initiator. As each ORB is executed the device transfers data in the appropriate direction using serial bus block transactions.

Following the data transfer portion of each ORB the Target writes a Status Block to the Initiator's Status_FIFO address. The Status_FIFO address is the address that was obtained in the Login process. The status block contains SBP-2 specific status information as well as device-dependant status information.

If an ORB containing a Null Next_ORB pointer is fetched the Execution engine completes all fetched commands, including the one in the just fetched ORB, before the Command_Agent transitions to the Suspended state.

If additional commands are to be executed, the initiator creates a new list of Command_Block ORBs; changes the Next_ORB pointer in the last ORB of the old list from NULL to the offset of the first ORB in the new list and then writes to the device's Doorbell CSR address. This transitions the Command_Agent to the Active state.

The device fetches the new Next_ORB pointer value from the last ORB of the old list and begins fetching ORBS from the new list at that offset.

If the Command_Agent fetch engine has not reached the ORB containing a Null Next_ORB pointer, (and is still in the Active state) the device ignores any writes to the Doorbell CSR address.

This sequence may continue until the device is reset, power is removed, or an error occurs.