1394 PWG

1394 PRINTER WORKING GROUP

IEEE 1394 HIGH SPEED BUS IMAGING DEVICE COMMUNICATIONS SPECIFICATION

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

Revision 0.1c - January 22, 1998

Editors -
Alan Berkema
Greg Shue

1 Scope

This document specifies the communications profile for imaging devices attached to the IEEE 1394 High Speed Serial Bus supporting the imaging device communications command set within the SBP-2 serial bus protocol.

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

- 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

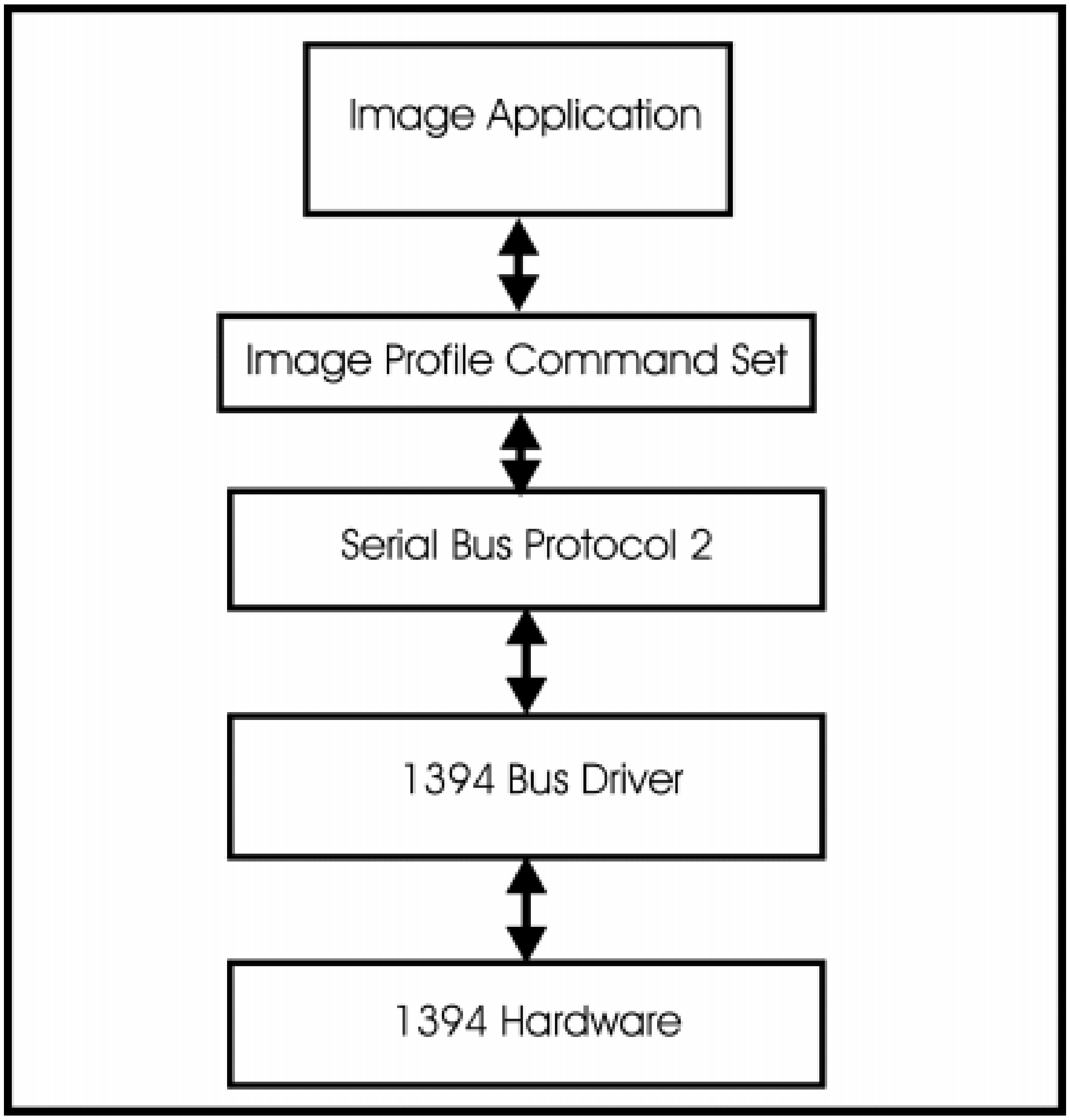
Information supplemental to the IEEE 1394 and SBP-2 specifications are provided in this document.

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 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.



3 IEEE 1394 Abstract

IEEE 1394-1995 was ratified in December of 1995. This standard describes a high speed serial bus that has a 64 bit address space, control registers, and read/write/lock operations that conform to the ISO 13213/IEEE 1212, Command and Status Register (CSR) standard.

In addition to the standard read/write/lock transactions used for Asynchronous communications the Serial Bus provides an Isochronous data transport that guarantees latency and bandwidth.

Data transmission uses two low-voltage differential signals to connect devices at 98.304 Mbit/s, 196.608 Mbit/s, and 393.216 Mbit/s speeds.

The cable medium allows up to 16 cable hops between any two device, each hop up to 4.5 meters long, giving a total cable distance of 72 meters. Bus management recognizes smaller configurations to optimize performance. The physical topology for the cable environment is a non-cyclic network (no loops) with the only limitation being the number of cable hops between any two devices (called “nodes”) and the length of a cable hop. The cables consist of 2 shielded twisted pairs for signals and one pair for power and ground. These cables connect to “ports” on the nodes. Each port consists of terminators, transceivers and simple logic. The cable and ports act as bus repeaters between the nodes to simulate a single logical bus. The 1394 Serial Bus also uses a fair bus access mechanism that guarantees all nodes equal access.
[5 Teener].

Plug and play is supported through automatic Node Identification without the need for switches or terminators.

The 1394 Serial Bus is not a network or an I/O channel, it is a shared memory architecture. The 64 bit address is divided into 16 bits for the Node ID (node number and bus number) and 48 bits (256 terabytes) for the memory space and CSR registers.

The 1394 standardization effort continues with p1394.a and p1394.b. P1394.a is focused on correcting ambiguities and problems in 1394-1995 as well as new enhancements such as Arbitrated Reset, Fly-By-Arbitration and Ack Acceleration. P1394.b is focused on higher speeds (800 Mbit/s) as well as long distance considerations using Plastic Optical Fiber.

4 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. Software Design for IEEE 1394 Peripherals, Peter Johansson.
5. New Technology in the IEEE P1394 Serial Bus, Michael Teener, March 29, 1994.
6. P1394a Draft Standard for a High Performance Serial Bus (Supplement).

5 Control & Status Registers (CSR)

The basic functionality of a Transaction Capable 1394 node includes fundamental PHY repeater operation as well as the implementation of certain core CSR's. The CSRs are defined within the initial register space beginning at offset 0xFFFF F000 0000. This allows the node to participate in Asynchronous read/write/lock transactions. The core CSRs are specified below:

Note:
Whether these registers are actually present as part of 1394 silicon or whether it is actually Random Access Memory is implementation dependent.

STATE_CLEAR and STATE_SET. All bits within these registers are optional but the registers themselves must be present.

NODE_IDS. Used to identify the 16-bit address of the node. In the cable environment, the LINK and PHY must together initialize this register during the self identify process that follows a bus reset.

RESET_START. This is a write-only register available to force a command reset of the node, as defined by ISO/IEC 13213:1994.

SPLIT_TIMEOUT In order to make requests, the node must implement a transaction time-out capability and make it configurable (or at least readable) through the SPLIT_TIMEOUT register.

6 Requirements For Isochronous Data Transmission

Serial Bus nodes that can participate in Isochronous operations, either as a talker or a listener, must have all of the features of transaction capable nodes. Additional requirements support the timing and detection of Isochronous operations. A key element in Serial Bus is that all Isochronous nodes share the same, coordinated time. Because Serial Bus is a distributed environment, each node must have its own 24.576 MHz cycle clock which runs freely when the node's link layer is active. This clock must be visible through the CYCLE_TIME register. Jitter in these clocks is eliminated every 125 usecs by the appearance of a cycle start packet on Serial Bus. A cycle start packet is essentially a write to the CYCLE_TIME register with a resynchronization value that eliminates jitter.

An Isochronous node must implement its resynchronization logic such that Serial Bus time, as observed by the values of the CYCLE_TIME register, can never give the appearance of moving backward. Although optional, Isochronous operations are expected to be inexpensive to implement within a node.

In addition to the requirements to talk or listen during Isochronous cycles, at least one node on a Serial Bus must be able to be the source of the synchronized cycle clock. This node is referred to as the **cycle master**. The cycle master must be able to use its 24.576 MHz clock to trigger cycle start events 8,000 times a second. As soon after a cycle start that the cycle master is able to arbitrate for the bus, it transmits a cycle start packet to resynchronize the clocks at all Isochronous nodes.

In addition to acting as the source for the Serial Bus cycle clock, a cycle master also has to implement the BUS_TIME register. This register is needed to extend the range of the Serial Bus clock from the limit permitted by the CYCLE_TIME register (about two minutes) to approximately 136 years.

An **Isochronous resource manager** is not actually an active manager of resources so much as it is an agreed upon location where all the Serial Bus nodes can cooperatively record their use of Isochronous resources, channel and bandwidth. BANDWIDTH_AVAILABLE is a register that stores the amount of Isochronous cycle time available to transmit data. Before any bandwidth is allocated, the register is initialized to a value that represents approximately 100 usecs. This permits some time to be implicitly reserved for asynchronous operations. CHANNELS_AVAILABLE is a register that is a bit map of all 64 possible Isochronous channels, free or in use. The Isochronous resource manager has another function, and that is to point to the bus manager, if any. Like the two CSR's just described, the BUS_MANAGER_ID register is a known location that is initialized by the node that assumes the role of the bus manager. It's important to note that a node that is Isochronous resource manager capable must be able to not only participate in the self identify process but to also analyze all of the self-ID packets observed. This is because if there are more than one Isochronous resource manager capable nodes on Serial Bus, only one becomes the Isochronous resource manager — the one with the largest 6-bit physical ID.
[4 Johansson]

It is recommended that all Isochronous nodes provide Isochronous Resource Manager functions. as well as cycle master capability.

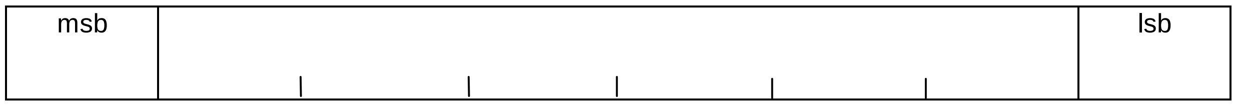
7 1394 Bus Management

The highest level of functionality available to a Serial Bus node is that of the **bus manager**. Bus managers need all of the capabilities discussed in the preceding sections plus additional intelligence to analyze the Serial Bus configuration and optimize it. It is likely that bus manager capabilities are implemented in firmware. They do not require additional hardware support as much as some sort of processor to perform the analysis. This does not, however, preclude the design of a bus manager entirely in logic. The key functions of a bus manager are: If Isochronous operations are desired but the current root node is not cycle master capable, the bus manager must perform a bus reset and insure that the new root can be the cycle master. The bus manager must collect and analyze the self-ID packets in order to make the TOPOLOGY_MAP and SPEED_MAP registers available. Serial Bus management applications need to communicate this information to system administrators. Serial Bus has a configurable parameter, the gap count, which is initially set to a default value. The bus manager can significantly improve Serial Bus performance by setting the gap count to a smaller value, the minimum which can be determined from maximum hop count of the current topology. Power management permits the bus manager to intelligently enable and disable selected nodes if the aggregate power demands are greater than the power available. An important distinction to remember about the bus manager is that it is an autonomous entity that has functions to perform even if no API is provided to a hypothetical bus management application. Also keep in mind that the costs associated with implementing a bus manager are not likely to be silicon costs, but testing costs. There can be a complex matrix of test cases to cover.
[4 Johansson]

8 Bit, Byte and Quadlet ordering

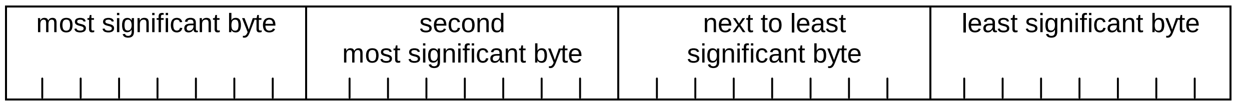
This profile defines the order and significance of bits within bytes, bytes within quadlets and quadlets within octlets in terms of their relative position and not their physically addressed position. Within a byte, the most significant bit, msb, is that which is transmitted first and the least significant bit, lsb, is that which is transmitted last on Serial Bus, as illustrated below. The significance of the interior bits uniformly decreases in progression from msb to lsb.

Bit ordering within a byte



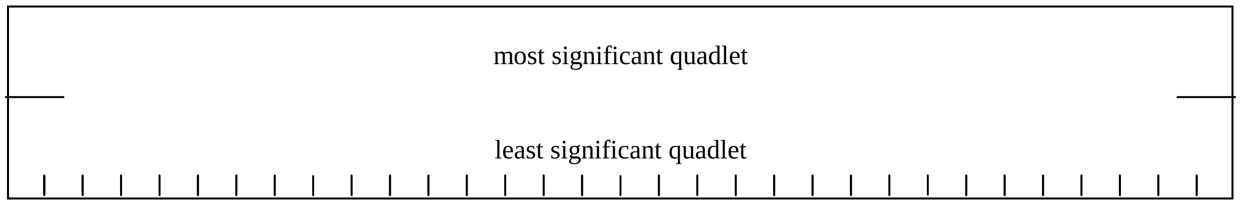
Byte ordering within a quadlet

Within a quadlet, the most significant byte is that which is transmitted first and the least significant byte is that which is transmitted last on Serial Bus, as shown below.



Quadlet ordering within an octlet

Within an octlet, which is frequently used to contain 64-bit Serial Bus addresses, the most significant quadlet is that which is transmitted first and the least significant quadlet is that which is transmitted last on Serial Bus, as the figure below indicates.



11. Control & Status Register (CSR) Summary

All 1394 PWG devices shall implement the CSRs as defined in ISO 13213/IEEE 1212:1994 and IEEE Std 1394-1995. Note that only the core and BUSY_TIMEOUT would be needed if Isochronous operations are not included. BUSY_TIMEOUT is needed for Asynchronous retry transactions.

Core Registers

Offset	Register	Initial Value
0x000	STATE_CLEAR	
0x004	STATE_SET	
0x008	NODE_IDS	
0c00C	RESET_START	
0x018-01C	SPLIT_TIME_OUT	

Serial Bus Dependent

Cycle Master

Offset	Register	Initial Value
0x200	CYCLE_TIME	
0x204	BUS_TIME	

Other Serial Bus Dependent

Offset	Register	Initial Value
0x210	BUSY_TIMEOUT	

Isochronous Resource Manager

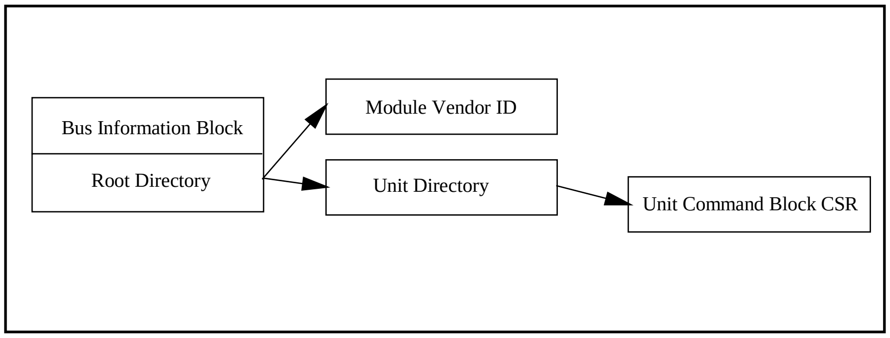
Offset	Register	Initial Value
0x21C	BUS_MANAGER_ID	
0x220	BANDWIDTH_AVAILABLE	4915
0x224-228	CHANNELS_AVAILABLE	All Ones

See the IEEE 1394-1995 specification for detailed information about each register.

9 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.



Note:
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.

First Quadlet
Offset: 0x400

bus_info_length	CRC_length	ROM_CRC_value
0x04	0x04	(calculated)

Bus Information Block
Offset: 0x404

0x31 "1"						0x33 "3"	0x39 "9"		0x34 "4"		
IRC	CMC	ISC	BMC	PMC	resv.	Cyc_Clk_Acc	Max_Rec	reserved 0x00	g	resv.	link_spd
Node_Vendor_ID									Chip_ID_High		
Chip_ID_Low											

Taken together the Node_Vendor_ID, Chip_ID_High and Chip_ID_Low are the EUI-64 also known as the Global Unique Identifier (GUID).

Upon detection of a bus reset the generate bit abbreviated as "g" in the bus info block shall be modified if any portion of the configuration ROM has changed since the prior bus reset. The CRC in the first quadlet will be recalculated each time the generate bit is modified.

Root Directory
Offset: 0x414

Directory Length 0x04	Directory CRC (calculated)
vendor ID key 0x03	Module_Vendor_ID (can be the same as Node_Vendor_ID)
Module_Vendor_ID _Key 0x81	Module_Vendor_ID_Textual_Descriptor_Offset (indirect offset) 0x03
Node_Capabilities_Key 0x0C	Node_Capabilities 0x0083E0
Unit_Directory_Key 0xD1	Unit_Directory_Offset (indirect offset)

Module_Vendor_ID_Textual_Descriptor
Offset: 0x428

Leaf Length		Leaf CRC (calculated)	
0x41 “A”	0x42 “B”	0x43 “C”	0x08 “ ”
string continued			
			0x00

Unit Directory
Offset: 0x43C

Unit Directory Length					Directory CRC (calculated)	
Unit_Spec_ID key 0x12	Unit_Spec_ID					
Unit_SW_Version key 0x13	Unit_SW_Version					
Cmd_Set_Spec_ID key 0x38	Cmd_Set_Spec_ID					
Command_Set key 0x39	Command_Set					
Command_Set_Rev key 0x3B	Command_Set_Rev ision					
Management_Agent key 0x54	Management_Agent_Offset (initial register space offset) 0x4000 (example)					
LU_Characteristics key 0x3A	q	o	I	reserved 0x00	Mgt_ORB_Timeout	ORB_size
Logical_Unit_Number 0x14	reserved 0x00			device_type	Logical_Unit_number 0x00	
Logical_Unit_Model_ID 0x17	Logical_Unit_Model_ID					
LU_Model_ID leaf 0x81	Logical_Unit_Model_ID_Textual_Descriptor Leaf offset (indirect offset)					

Leaf Length 0x04		Leaf CRC (calculated)	
0x41 “A”	0x42 “B”	0x43 “C”	0x08 “ ”
string continued			

Management_Agent_Register
Address: 0xFFFF F001 0000 (example)

reserved 0x00	ORB_offset_hi
ORB_offset_lo	

The Management Agent Register shall be written using a 1394 Block write with 8 bytes of payload. Even though SBP-2 uses the word offset as part of the ORB address names in ORB_offset_hi and ORB_offset_lo, these values combined with the node ID form an actual IEEE 1394 64-bit address.

Unit Command_Block_Agent CSRs (address is returned in the Login response)
Address: 0xFFFF F001 0008 (example for single Login device)

Relative Offset	Name	Description
0x00	Agent_State	Reports fetch Agent State
0x04	Agent_Reset	Resets fetch agent
0x08	ORB_Pointer	Address of ORB
0x10	Doorbell	Signals fetch agent to re-fetch an address pointer
0x14	Unsolicited_Status_Enable	Acknowledges the Initiator’s receipt of unsolicited status
0x18 - 0x1C		Reserved

The Unit Command_Block_Agent CSRs are provided in the Target’s memory space. The Address is returned in the Login Response.

The Isochronous_Plug_Register address shall directly follow the Command_Block_Agent CSR address.

Isochronous_Plug_Register
Address: 0xFFFF F001 0020 (example)

reserved 0x00	Isochronous Channel
Bandwidth (total number of bytes - includes 1394 header and payload CRC)	

10 Imaging Device Requirements and Desires

The requirements and desires specified by the range of imaging device client applications to the transport layer are intentionally brief, with definitions of terms following each specified item.

10.1 Requirements

Lack of provision of these items by a transport layer prohibits it's use by imaging class devices.

- Support multiple, concurrent, independent, symmetrical connections

  Connection - well-bounded communication path between two endpoints. The endpoints can be on the same device or on different devices.
  Multiple, concurrent - Allows for more than one connection at a time.
  Independent - Activity on one connection has no effect on other connections.
  Symmetrical - Either endpoint can open(?) and close the connection, and send data. *Question: Since some end has to be in control of the "job", the opposite end really behaves like a 'server' device. Does it really make sense to require that a 'server' can open a connection to a 'client'?.*

- Provide in-order, byte-stream and buffer (datagram?) services - Data is delivered to the receiving endpoint in the same order as it was presented by the sending endpoint.

  Byte-stream - Data is delivered as a stream of bytes. The stream of bytes is not guaranteed to be delivered to the receiving endpoint in the same form as it was presented by the sending endpoint. For example, a stream of 80 bytes of data may be presented as 4 20-byte buffers.
  Buffer (datagram? Packet?) - Data is guaranteed to be delivered to the receiving endpoint in the same form as it was presented by the sending endpoint. For example. If data is presented in a buffer of 30 bytes followed by a buffer of 10 bytes, it must be delivered in a buffer of 30 bytes followed by a buffer of 10 bytes.

- Provide a directory service

  Endpoints on a specific device may be referenced by their service name. This allows connections to be opened without any knowledge of the underlying layer's implementation of sockets, etc.

- Be data, application, and O/S independent

  The transport stack shall not put any requirements on the format of the data, nor shall it interpret the data in any way. The transport stack shall work with any application that correctly uses the appropriate interfaces. The transport shall be implementable under any operating system.

- Do not preclude concurrent operation of other protocol stacks

Devices may implement and use other protocol stacks concurrently with this transport stack.

- Transparently handle transient link interruptions

The transport stack shall handle transient link interruptions without affecting the endpoints. These link interruptions include: temporary cable disconnect, 1394 bus reset, etc. A “transient” link interruption is defined to be short and non-catastrophic with respect to the connection, the services provided, and human time. A temporary cable disconnect is explicitly defined to be longer than one second, since this is defined to be a human interaction.

10.2 Desires

Lack of provision of these items by a transport layer does not prohibits it’s use by imaging class devices, but provision of them allows for broader solutions. These desires are listed in no particular order and are of equal significance.

- Connectionless service

A non-bounded communication path between two endpoints. Data may be sent without “opening” a connection.

- Multi-casting

Simultaneously sending data from one endpoint to multiple endpoints. Does this need to be bi-directional? Does it need to be reliable?

- Bus-independent transport layer

The transport layer may be used on other busses.

- Data tagging

Data can be tagged as “special data” by the sending endpoint. The transport will indicate to the receiving endpoint that the data is tagged. This is also known as out-of-band data. The data is kept synchronous with the data flow.

- Provide endpoints with fair access to other endpoints

The transport will prevent endpoints from monopolizing the link and preventing other endpoints from access.

- Selectable quality of service

The ability to adjust various quality of service parameters, including:

Isochronous delivery
Priority
Propagation Delay
Rate of transfer (bandwidth)

11 Provisions of this Profile

The table below contains the feature set of this profile relative to the requirements and desires. The required features are listed in a bold typeface.

Requirement	Comments
Connections	Achieved by (an extended) access control to the SBP-2 interface via login/logout.
Multiple	Achieved by a separate login for each connection
Concurrent	Achieved by a separate login for each connection
Independent	Achieved by a separate login for each connection
Symmetrical	Achieved since after a connection has been established, either end may send data to the other end independent of the opposing data flow. Either end may close the connection at any time.
In-order data delivery	Achieved by In-order with respect to a given direction. Independent of the data flow of the opposing direction.
Byte-stream	Achieved by API
Buffer mode	Achieved by API
Directory service	**To Be Specified**. Should be able to be used with the proposal being developed by the PWG 1394 sub-committee.
Data independent	Achieved since transport client data payload is contained entirely within ORB data payload. Only transport commands and status are encoded within the command set.
Application independent	Achieved since no commands or policies are relied upon, implemented, or imposed which are beyond what is listed in the requirements or is limited in scope to the internals of the profile.
O/S independent	Achieved since no behaviors, features, or constructs are defined beyond which are observable across the 1394 bus and understandable within the framework of 1394, SBP-2, or an SBP-2 command set.
Concurrent existance	Achieved by the framework of 1394 and SBP-2. No additional 1394-level restrictions are imposed.
Transient link interruption tollerance	Achieved by this Profile

Desires	**Comments**
Connectionless	**Not Supported.**
Multi-casting	**Not Supported.**
Bus-independence	**Not Supported.**
Data tagging	Achieved since End-of-data-for-direction-for-connection tag is always valid. End-of-buffer tag is implicitly maintained for buffer-based connections. Additionally, transport allows an explicit tag indicating transport client provided out-of-band data is contained within the data payload, but no restrictions are provided on the contents of the data payload.
Fair access	**Not Supported.** Since fair access requires maintaining history of access requests, and device resources to store history could be exceeded, this has been left outside the scope of this profile.
Selectable quality of service parameters	**Not Supported.**
Isochronous	**Not Supported.**
Priority	**Not Supported.**
Propagation Delay	**Not Supported.**
Bandwidth	**Not Supported.**

11.1 Issues

Microsoft has implemented their SBP-2 driver to do a login on powerup 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 service cannot be simultaneously used by the different PCs.

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?

12 Discovery

The primary method for discovering devices on the Serial Bus is through information read from the Configuration ROM. Continued with information supplied by PWG and IEEE 1212 when this settles.

The Function Discovery Service is an initial attempt at performing high level 'Function' discovery.

[The Unit directory section should contain more info about the Service Discovery.]

Should this also contain information about Plug-N-Play support?

13 Why SBP-2

The 1394 PWG has examined several Protocols since its formation in early 1997 (see Appendix A) and measured them with respect to the requirements. From the beginning SBP-2 has been a leading contender and some of the primary reasons are listed below:

- Existing standard being used for other devices.
- SBP-2 is an efficient transport protocol and is optimized for 1394 DMA shared memory access.
- SBP-2 works well for normal print and scan data flow

Perceptions about its lack of bi-directional functionality caused concern. The specific test case, which SBP-2 does not address directly, occurs when a Target is obligated to send an undetermined quantity of data back to the Initiator. In this case the Initiator cannot predicate how much data the Target will send and therefore the Command Block ORBs required to receive this data may not have been built and provided in advance.

Several options to modify SBP-2 were proposed to work around this perceived deficiency. Some of these are listed below as an aid to understanding the events that lead to the current proposal.

- Multiple Fetch Agents In a Target
- Targitator - Dual Logins
- Abort Task List
- Fetch Ahead
- Bi-directional ORB (a.k.a. Request/Reply ORB)
- Single Command Block ORB

[Add: Explanation of each option above]

The main draw back to each of the options above is that they call for a significant modification of the SBP-2 spec or to use SBP-2 in a way that a standard driver might not support. The proposal, which follows in the next section attempts to enhance standard SBP-2, perhaps in a way which is unique to image devices, that could be supported by a standard SBP-2 driver.

14 SBP-2 Communication Protocol

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.

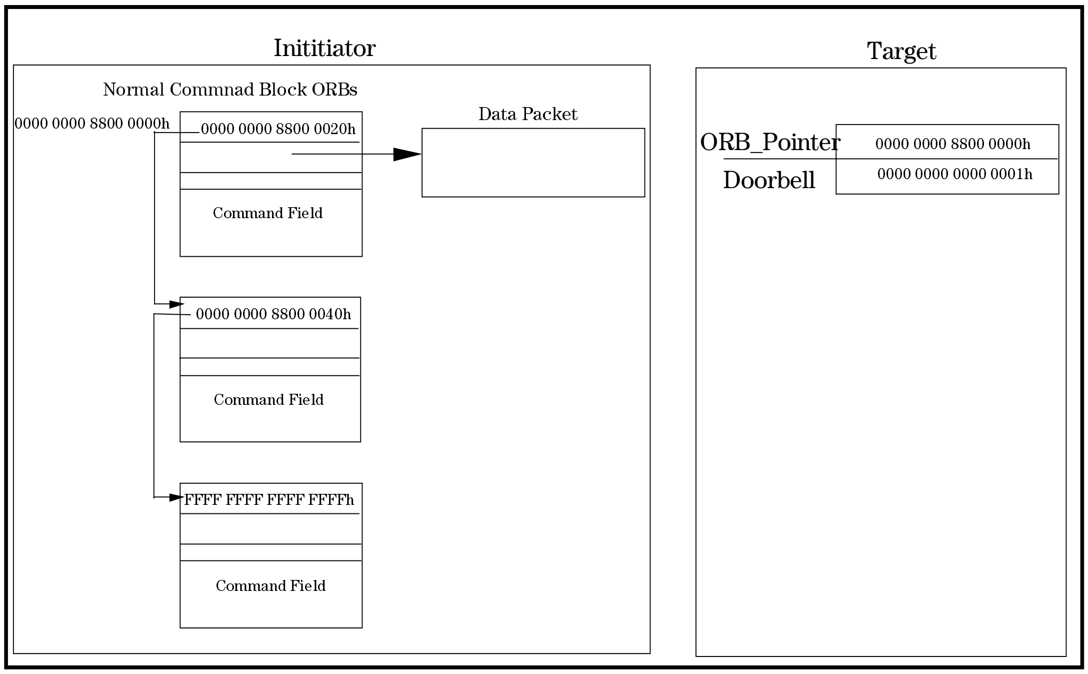
After power-on or bus reset, the Command_Agent and Management_Agent engines are in the Reset state.

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 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 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.

15 ORB List Processing

This specification defines the basic communication path as a Login from an Initiator to a Target. The service that an Initiator logs into is represented by a Unit Directory. An instance of that service is represented by a logical unit. A Target may have more than on Unit Directory. A specific Initiator is only allowed one Login per logical unit on a specific 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.

16 Out-of-order Request Completion clarification (informative)

The SBP-2 specification explicitly provides for out-of-order execution and completion of requests on a request list, but it does not give even a normative procedure for dynamically removing a completed request from the middle of an active request list. As such, all that is allowed by the specification is the recovery of all the resources associated with an ORB other than the memory and the 1394 Bus Address Range associated with the ORB header. The only information which is still associated with the ORB and is required by the target's active fetch agent is the Next_ORB field.

Presumably, a completed ORB may be removed from the middle of an active request list simply by updating the Next_ORB pointer of the previous ORB to contain the value of the Next_ORB field of the ORB being removed, and then writing to the Doorbell register of the fetch agent. Writing to this register will cause the fetch agent to immediately revalidate any ORBs cached or in progress before resuming operations.

Since this behavior is not command-set dependent, but central to the SBP-2 protocol, it cannot be relied upon without an amendment to the SBP-2 specification. This results in two profiles being described in the Annexes, one for an ordered execution model (which is compliant with the current SBP-2 specification), and another for an unordered execution model (which relies upon the capability presumed).

17 Model (informative)

This section describes the model of use of SBP-2 and the extensions provided by this profile. Though the features are application independent, this section describes the intended range of the imaging device class in order to better understand the communications provided by this profile.

17.1 Imaging Device Class Model

The imaging device class is intended to support devices which provide either image source or image sink capability. They are characterized by:

Requiring data integrity be maintained

Devices carry internal modifiable state information which is limited to an access session (i.e. job- or session-oriented)

An access session may span a long time (in human terms)

Physical device status exists which is outside the scope of the image source or image sink service but may reflect the devices (temporary or permanent) inability to complete the requested operation.

These devices generally include scanners, printers, and digital still cameras. Each device is modeled as providing at least one image source or image sink service, a device status service, a device control service, and zero or more applications which can use these same types of services on other devices.

18 Model Configuration

Each service is built upon this communications profile and is characterized by the following parameters:

Data packet vs. Data stream based communications model
image source, image sink, status, or control type service
transport client command set supported

Each unique combination of the above information represents a unique service provided by the device and is identified as a separate Unit within the device. Zero or more instances of a unit may exist within the device at any given time. All instances of a unit have identical functionality. Each instance of a unit is uniquely identified by a logical unit number assigned during login. LUNs are assigned upon successful login in numerically ascending order beginning with zero. A login request of a LUN which is already assigned shall return a response code of (4) access denied. A login request of a LUN which is unassigned, but unsuccessful due to resources shall return a response code of (8) Resource unavailable.

The Unit Directory shall not enumerate each LUN which may be supported.

Issue: 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. ???)

18.1 Model Operation

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

The Initiator performs a Login operation prior to any request to the device. This login process conforms to the SBP-2 specification, and is extended to determine the following information:

Access Resumption Timeout - an extended timeout on access reservation to the service interface. This is needed for support of transient link interruptions.

Transport Alive Timeout - used to determine that the initiator's transport layer has failed in such a way that no indication (e.g. Bus Reset) is generated. This is a response timeout for a "ping" type of request from the target.

Maximum Task Data Payload per direction - used to determine the largest data payload transferable in each direction. The transfer must be completed and acknowledged before being passed along to the receiving transport client. This is needed to provide data integrity across Bus Resets and transient link interruptions.

The initiator is required to enable unsolicited status from the target. The unsolicited status field is used by the device to indicate when new target to initiator data is available and no task was detected on the request list for receiving the data. This is used as an early indication that data is available. If the target finds itself in a deadlock situation and cannot continue processing the requests in order, the target shall abort all data transfer requests and let the initiator reissue the requests to receive data into the initiator. It is also used by the target to request the initiator to confirm that the initiator's transport stack is still responsive.

All data transfers within a specific direction between the initiator and the target are limited to the Maximum Task Data Payload negotiated for that direction. In Buffer mode, a data buffer shall be limited to the Maximum Task Data Payload negotiated for that direction. Each data payload shall contain exactly one buffer.

All data payloads for a specific direction are processed in the order queued on the task list. Since out-of-order completion and removal of ORBs from the middle of an active task list is not addressed by the SBP-2 specification, the out-of-order task list execution model is too vague and too limited to be an option. So, all task list processing shall use the ordered execution model.

Every command shall have the 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 target shall report Data Available to the initiator whenever data exists to go to the initiator. If the target reaches a deadlock condition where it cannot process the data transfer into the target until it can send data to the initiator, the target shall abort all tasks on the task list after reporting Data Available to the initiator. Upon a Bus Reset, all tasks on the task list are implicitly aborted. Because no data is sent to the receiving transport client until the task has been successfully completed, neither a Bus Reset nor a task set abort can corrupt the data. All requests for data from target by the initiator shall be the Maximum Task Data Payload size negotiated for target to initiator data.

Upon a Bus Reset, the SBP-2 reconnection process shall be followed. A successful reconnection shall be transparent to this profile. Upon the timeout of the reconnection window, the extended access control policy shall take effect and reserve the logical unit number and service interface for the initiator until the extended timeout expires. During this time the internal server state shall be retained. Upon expiration of the extended access control policy, the access reservation ID shall become invalid for another extended access control timeout interval.

The communications profile command set consists minimally of:

Receive from Initiator - transfer data payload to the target

The command carries a tag for indicating end of data for connection, and a tag for indicating the data payload carries out-of-band data.

Send to Initiator - transfer data payload from the target

The command carries a tag for indicating end of data for connection, and a tag for indicating out-of-band data, and a field for indicating amount of valid data sent to the initiator.

Reserved - reserved for future extensions

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.

The process used by the target to close a connection shall be by aborting the current task set with a status of (10) Login ID not recognized. If no task set exists, and credit exists to allow an unsolicited status message to be sent, then an unsolicited status message shall indicate that a spontaneous logout was performed. If neither condition exists when the target wishes to terminate the logout, then the initiator must infer the connection being closed upon the next interaction with the target. The target shall not wait for confirmation of the logout before releasing resources.

19 PEER to PEER

The peer to peer requirement states that any device can initiate communication. Since SBP-2 is architected around an Initiator and a Target model, true peer to peer is not a natural fit. If peer to peer is really needed it can be accomplished.

The Imaging Device Profile provides a mechanism for two nodes to communicate with one another in a peer to peer fashion. The mechanism requires each node to implement target and initiator functionality as defined in the SBP-2 specification. The command set used by such a device must be able to adapt to the differences of being either the initiator or target as defined by SBP-2. The details of such an implementation are beyond the scope of this specification.

20 Multiple Host and/or Multiple Device

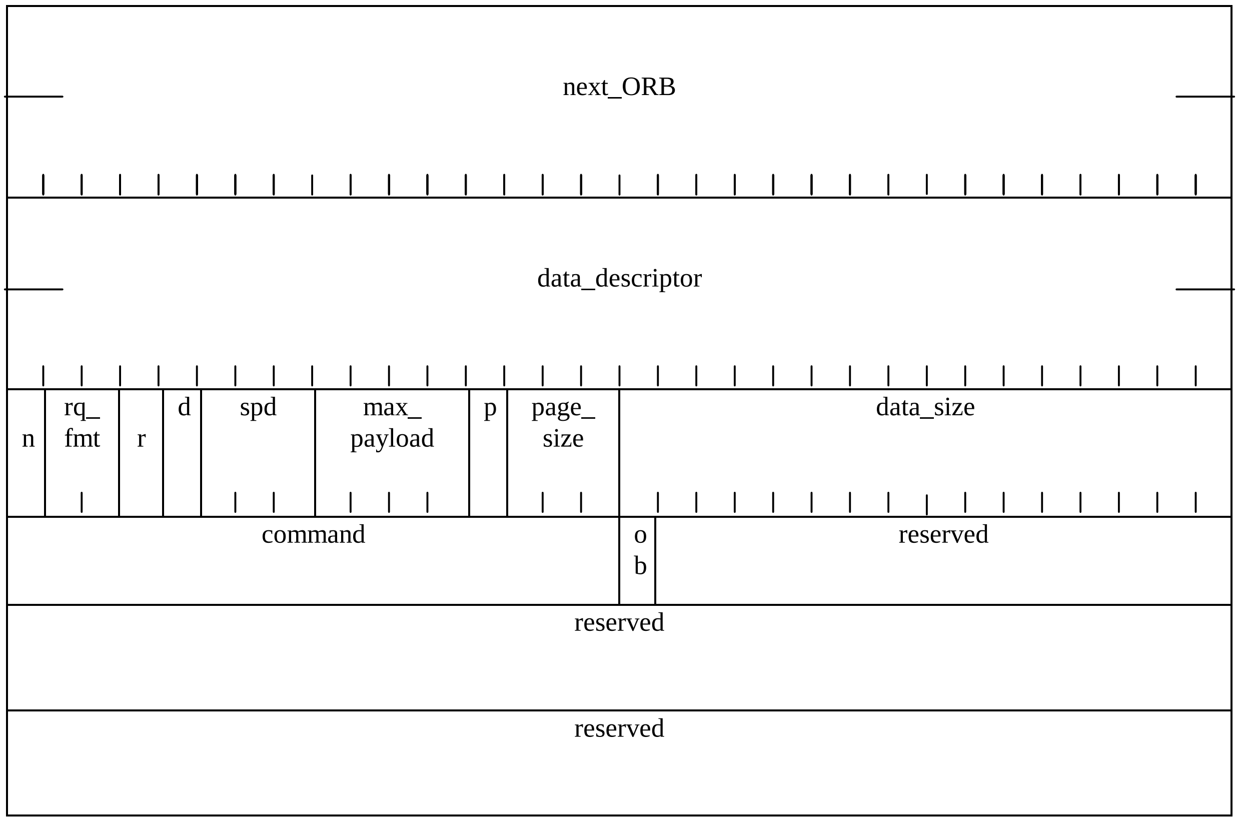
There is a need to provide fair access to devices on a 1394 bus. Each node is accessible from any of the other nodes on the bus and possibly nodes outside of the bus in a bridged environment. It is reasonable to expect that more than one initiator node may attempt to communicate with a target service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

21 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. The *command_block* field shall follow the format below.

Command Block ORB



The Command field (? bits) shall be encoded as follows. All unspecified values are reserved for future use.

Command Code		Description
??		SND data in - transfer data block from target to initiator
??		RCV data out - transfer data block from initiator to target

21.1 Send Data-in Command

The Send Data-in command requests the target to store up to the requested data size (or one packet if in packet mode) into the referenced buffer. The buffer shall be able to contain up to the negotiated target to initiator Max Data Payload Size. The amount of valid data stored shall be returned in the status block for the request. The status block shall be written only after the data payload is valid for the return status. Successful completion of the status block write shall indicate completion of the command.

Data payloads shall be limited to 2^{20} bytes.

The ob field (out-of-band) shall be treated as a reserved field with respect to the Send Data In command.

The status block shall indicate if additional data is available for the initiator as of the time of completion of the command.

Send Data In Response Format

sm ft	d a	o b	reserved	valid_length

smft Code	Description
0	Normal command response format
1-2	Reserved
3	Unsolicited Status response format

da - Data Available Code	Description
0	No target data exists for initiator
1	Target data exists for the initiator

ob - out-of-band tag. Shall be set to one by the target if the data transferred has been identified as out-of-band data, and shall be set to zero otherwise.

valid_length - amount of valid bytes transferred to the data payload buffer

21.2 Receive Data-out Command

The Receive Data-out command requests the target to store the referenced data into the target’s input buffer. The buffer shall be no larger than the negotiated initiator to target Max Data Payload Size. The amount of valid data to store is encoded as per the SBP-2 specification . The status block shall be written only after the data payload is valid for the return status. Successful completion of the status block write shall indicate completion of the command.

Data payloads shall be limited to 2^{20} bytes.

The ob field (out-of-band) shall be set to one if the data is tagged as out-of-band by the transport client, and shall be set to zero if the data is not tagged as out-of-band by the transport client.

The status block shall indicate if data is available for the initiator as of the time of completion of the command.

Receive Data Out Response Format

smft	da	reserved

smft Code	Description
0	Normal command response format
1-2	Reserved
3	Unsolicited Status response format

da - Data Available Code	Description
0	No target data exists for initiator
1	Target data exists for the initiator

22 Stream Command Block ORB

All devices that conform to this communications profile shall recognize the Stream Command Block ORB and process the request with a *resp* equal to zero, REQUEST COMPLETE, and the *sbp_status* equal to one, Request type not supported.

23 Stream Control ORB

All devices that conform to this communications profile shall recognize the Stream Command Block ORB and process the request with a *resp* equal to zero, REQUEST COMPLETE, and the *sbp_status* equal to one, Request type not supported.

24 Management ORB

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

25 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.

In order to meet the requirement that this communications profile is tolerant of transient link interruptions, this requires that access control to the device is maintained across the interruptions. Since these interruptions may last longer than the explicit one second reconnect timeout specified by SBP-2, The following issues must be addressed.

- Extending the Access Control (login/reconnect) Service
- Allowing the target to determine when the initiator is no longer operational
- Guaranteeing data transfers are independent of transport client processing

This requires additional information to be passed with the Login ORB and shall be encoded absolutely in the password field. The response information shall be appended to the end of the Login Response structure and shall be included in the length field of that structure.

25.1 Negotiated Access Resumption Timeout

Initiators are not allowed to have permanent access to a service and targets are not allowed to give permanent access of a service. Initiators and targets negotiate for the maximum commonly supported access resumption timeout. Since the nature of the feature is to support transient link interruptions (in human terms), the specified maximum access resumption timeout is limited (arbitrarily) to 5 minutes. The supported maximum is implementation-dependent.

Currently, the Microsoft policy is to login to every identified service during bus enumeration (during boot-up), and to not logout until the system is shut down. This not only goes against the desired behavior described above, but it also does not extend well to 1394 bus configurations with multiple nodes running a Microsoft O/S, or with other nodes wishing access to the devices on the bus (such as interoperable peer devices). This issue must be resolved.

The target shall retain sufficient information to allow each initiator to regain service access (and any associated context) for the negotiated timeout subsequent to an implicit logout due to a missed reconnect . Access may be resumed with a keyed login request. Once the access has been resumed, the LUN's timer shall be stopped and reset. After the timeout expires, the target shall release access to the service and shall perform an implicit logout for all login ID's that have not been successfully reconnected to their original initiator(s).

When the initiator explicitly issues a logout command for the session, the resources shall be immediately released.

25.2 Negotiated Transport Alive Timeout

It is desirable for a target to be able to detect that the initiator stack is not functioning in such a way to cause a Bus Reset event. This is beneficial to help release the resources accessed by the initiator. This leads to a need to negotiate an initiator response interval to an unsolicited status transfer. This may be used by the target to "ping" the initiator when it needs to recover resources. The target shall ping the initiator by issuing an unsolicited status message indicating the current status. If the initiator does not issue additional unsolicited status credit within the negotiated timeout, then the login shall be invalidated by the target, and access to the device shall be released.

If Unsolicited Status credit does not exist, the target shall wait for one negotiated timeout interval for indication of initiator activity. If unsolicited status credit still does not exist, the initiator will be automatically logged out by the target, access control to any interfaces or state maintained by this service will be released, and the service will become generally available on the 1394 bus.

25.3 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^{20} bytes.. Sizes must be negotiated per direction. The details of this scheme are specified in the Password field of the Login ORB and as an extension to the Login Response ORB in the following sections.

25.4 The Extended Access Control Policy -

The login protocol shall be extended as follows:

For each logical unit, an additional access_control_descriptor shall hold context required to extend the access control policy. This context consists of the access_owner_EUI_64, a access_owner_ID, and the access control negotiated parameters.

The access_owner_EUI_64 is the unique 64-bit identifier of the current owner of access to the application. Upon a power reset, the access_owner_EUI_64 for all access_control_descriptors shall be reset to all ones. Upon expiration of the access_release_timeout plus one second, or upon an explicit successful logout by the initiator, the access_owner_EUI_64 shall be reset to all ones.

Before an initiator may signal any other requests to a target it shall first perform a login. The login request shall follow the process described in the SBP-2 protocol. The target shall additionally validate a login request as outlined below:

If the logical unit's access_owner_EUI_64 field is not set to all ones, then:

If the logical unit's access_owner_EUI_64 field does match the EUI-64 of the initiator, then:

if the access_owner_ID does not match the access_owner_ID supplied in the Login ORB, then the target shall reject the login request;

If the negotiated values do not match those stored in the access_control_descriptor, then the target shall reject the login request;

Upon successful validation of the login request, the target shall

Reset the service associated with the requested logical unit if and only if the initiator's EUI-64 does not match the value stored in the access_owner_EUI_64 field;

Subsequently store the initiator's EUI-64 value in the access_owner_EUI_64;

Assign an access_owner_ID for this session;

compute the minimum access_release_timeout value between the timeout requested by the initator and the maximum timeout supported by the application, store this timeout value in the access_control_descriptor;

Compute the negotiated maximum_data_payload_size for each direction (initiator to target and target to initiator) between the sizes requested by the initiator and those supported by the target, and store these in the access_control_descriptor

Compute the minimum initiator_inactive_timeout value between the timeout requested by the initiator and the maximum timeout supported by the application, and store this timeout value in the access_control_descriptor;

Finally return the access_owner_ID, the access_release_timeout, the maximum_data_payload_size for each direction, and the initiator_active_timeout as specified in the ammended Login response descriptor specified in section ???

25.5 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

Initiator to Target Data Payload limit			ntat	Target to Initiator Data Payload Limit	nart
reserved				access_owner_ID	
login_response					
n	rq_ fmt	x	reserved 0x00	Function 0x00	lun
password_length 0x00				login_response_length 0x05	
Status_FIFO					

4) The Initiator will build the Login ORB with 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.

    During the login process the initiator shall provide the access_owner_ID assigned to the previous login (if resuming access), or shall use the value of zero to request a new access. The target shall attempt to validate the request, and shall reject it if the access cannot be given. The

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		login_ID	
command_block_agent 0xFFFF F001 0008 (example)			
Initiator to Target Data Payload limit	ntat	Target to Initiator Data Payload Limit	nart

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 of the size of the Login Response specified in this standard.

The Login response shall be extended by one quadlet. This quadlet shall be of the same format as described above. The information shall be the minimum values between what the initiator desires and what is the maximum supported by the target for this service.

Initiator to Target Data Payload limit - This is the maximum data payload size supported for confirmed initiator-to-target data transfers. The value is in 128-byte units.

ntat - Negotiated Transport Alive Timeout. This is the timeout used by the target to determine that an initiator is no longer functional. It waits this amount of time for the initiator to respond to an unsolicited status transfer by giving additional unsolicited status credit.

Table XXX - Negotiated Transport Alive Timeout Code

Reconnect Timeout Code Field	Duration (seconds)
0_{16}	100 (msec)
1_{16}	200 (msec)
2_{16}	500 (msec)
3_{16}	1
4_{16}	3
5_{16}	5
6_{16}	10
7_{16}	15

Target to Initiator Data Payload Limit - This is the maximum data payload size supported for confirmed target-to-initiator data transfers. The value is in 128-byte units. All requests for data transfers from the target shall be of this size. The target may return less than the requested information.
nart - Negotiated access resumption timeout. This is the extended time allowed by the target for an initiator to reestablish a login to a service after failing to reconnect in sufficient time.

Table XXX - Negotiated Access Resumption Timeout Code

Reconnect Timeout Code Field	Duration (seconds)
0_{16}	0
1_{16}	5
2_{16}	10
3_{16}	15
4_{16}	30
5_{16}	60 (1 min)
6_{16}	120 (2 min)
7_{16}	300 (5 min)

26 Logout

26.1 Initiator-Explicit Logout

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

26.2 SBP-2 Implicit Logout

Upon detection of an implicit logout by SBP-2, the access_release_watchdog shall start. This watchdog shall only be stopped by it's expiration or a revalidated access. Access only needs to be revalidated after an implicit Logout.

26.3 Initiator Inactivity

Upon expiration of the transport_alive_timeout, the target shall implicitly logout the initiator and immediately release resources. The conditions which cause the target to start the transport_alive_timer are beyond the scope of this document.

27 Unsolicited Status

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 to one.
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.

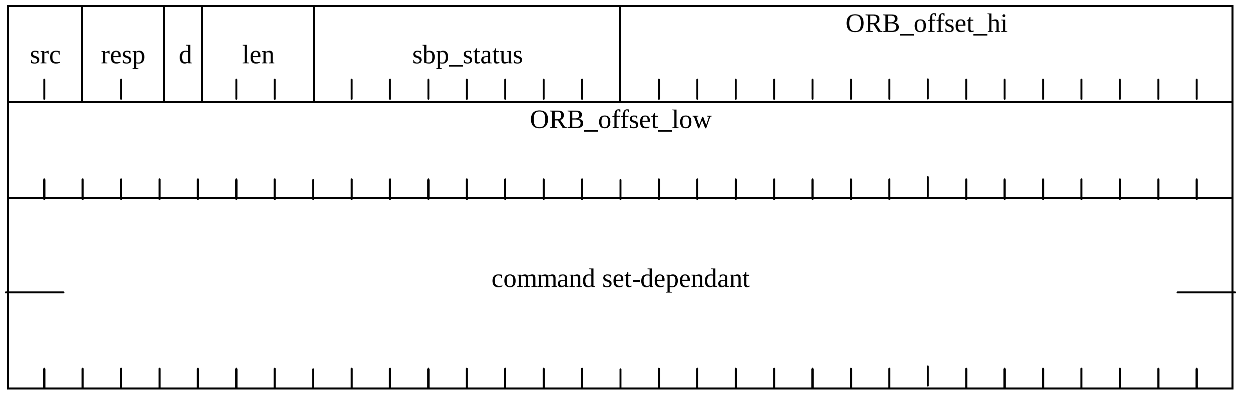
28 Status Block

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.

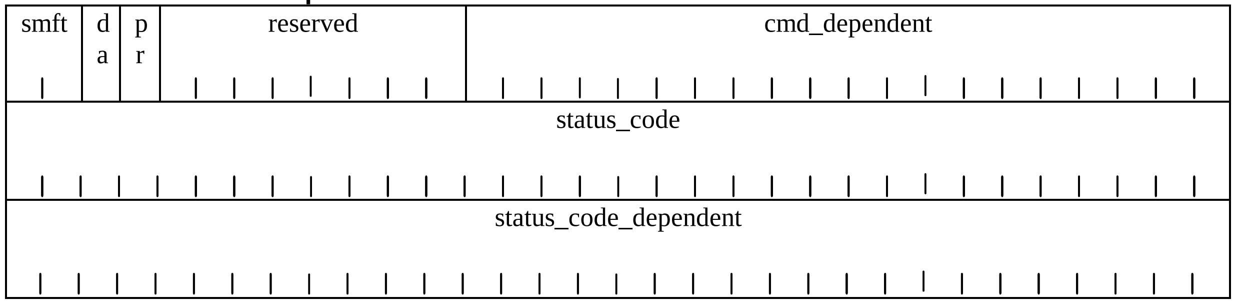
resp field

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

1284.4 Command Set-Dependent Status



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

sfmt value	Description
0_{16}	status block format defined by this standard
1_{16} - 2_{16}	Reserved for future standardization
3_{16}	Status block format vendor-dependent

da - The *data_avaiible* field shall indicate whether or not data is available for the initiator at the time the command was completed. A value of zero shall indicate that data is not available. A value of one shall indicate that data is available.

ia - The *initiator_alive_req*field shall only be valid within an unsolicited status update. It shall be ignored otherwise. A value of zero shall indicate no request from the target. A value of one shall indicate a request by the target for the host to write to the UNSOLICITED_STATUS_ENABLE register within the negotiated *transport_alive_timeout*. The initiator shall expect that the timer began upon successful completion of the unsolicited status write into the Status_FIFO.

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

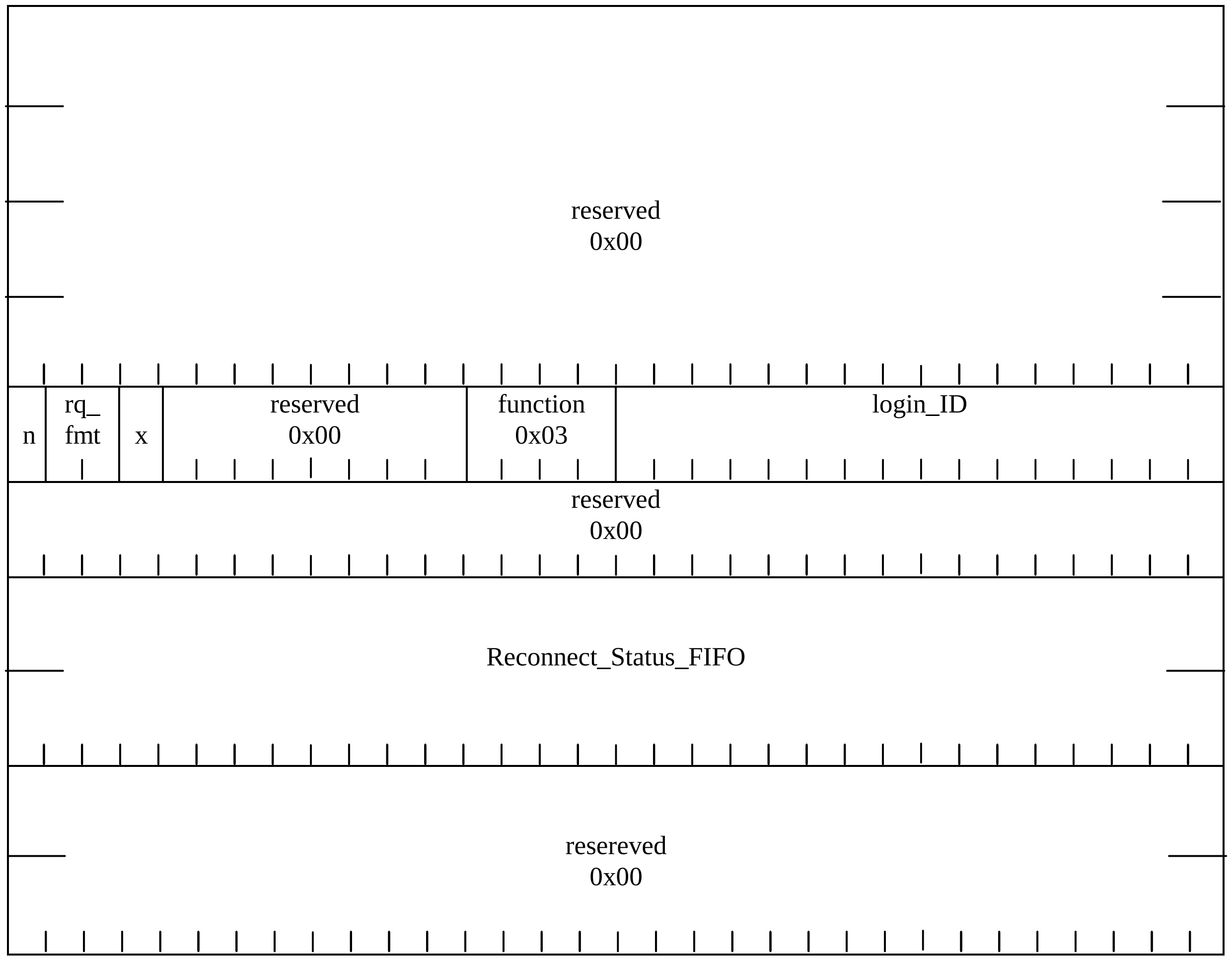
status_code	description	status_code_dependent
0x01	data transfer complete	actual number of bytes transferred
0x02	Target to Initiator transfer request	number of bytes to transfer

29 Reconnection

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

A successful reconnection shall be interpreted as initiator activity, resulting in the initiator_inactive_watchdog being reset and restarted..

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 timeout period.

30 Query Login

31 Isochronous Data Transmission

32 1394 Bus Reset Behavior

Upon a Bus Reset, the initiator_inactive_watchdog shall be reset and restarted to allow a window for the initiator to re-establish the connection and give permission for the target to issue an Unsolicited Status message.

After a Bus Reset:

During idle state - no data transactions pending

Reconnect as described in preceding section.

During Asynchronous data transmission

Reconnect as described in preceding section and continue data phase.

During Isochronous data transmission

Continue with Isochronous data transmission.

33 Error Recovery

Asynchronous Data Transmission Error Recovery

Any packet, which contains a CRC error, shall be re-transmitted when the error_ACK is returned to the sender.

Appendix A - Protocol Proposal Comparison

The 1394 PWG has explored several options for peripheral communications protocols. In general proposals have gravitated towards SBP-2 and IEC 61883 (FCP). As these were examined with respect to the list of 1394 PWG requirements both of these seem to …. to be continued

Appendix B - SBP-2 Extended Access Control Policy Proposal

The issue of a fixed access control limit imposed by SBP-2 reconnect presents a generic problem to devices which do not use isochronous services and want to tolerate transient link interruptions. Therefore, it seems reasonable to request a generic extension to SBP-2 to address this issue. It is recommended that the following request be given to the committee working on the SBP-2 proposal and future revisions.

Use three of the reserved bits in the Login ORB and define them for a reconnect timeout negotiation field. The same number of bits needs to be defined in the Login Response structure for indicating the resulting reconnect timeout code. A value of zero is used to indicate the (default) of 1 second for backwards compatibility. If isochronous services are supported, the maximum value supported shall be zero. Reconnect timeout are configured from the following table. The value returned in the Login Response structure is the minimum value of what is requested by the initiator and the maximum of what is supported by the target.

Table XXX - Reconnect Timeout Code

Reconnect Timeout Code Field	Duration (seconds)
0_{16}	1
1_{16}	5
2_{16}	10
3_{16}	15
4_{16}	30
5_{16}	60 (1 min)
6_{16}	120 (2 min)
7_{16}	300 (5 min)