Design goals

- Build upon SBP
- Encapsulate 12-byte CDB’s, minimum
- Optimize for single-initiator environment
  - Simple queuing model
  - No asynchronous event notification (AEN)
  - Permit multiple-initiator architecture
- Simplify target hardware and firmware
  - No FIFO’s to accept “taps”
  - Small request blocks to maximize use of on-chip memory
  - Uniform fixed-length status information
- Isochronous support optional
Key differences from SBP

■ New command delivery mechanism
  ➤ No “taps”
  ➤ Target paces all command delivery to suit its needs
  ➤ One fetch agent per initiator per logical unit
    – No need for subchains or round-robin scheduling
    – Eliminate some request block fields (e.g., LUN)

■ Compact, 32-byte request block

■ SAM requirements simplified
  ➤ No asynchronous event notification (AEN)
  ➤ No auto contingent allegiance (ACA)
Key enhancements to SBP

■ Isochronous model articulated better
  ➤ Isochronous stream is the unifying concept
    – One or more channels form a stream
  ➤ Two target objects work in tandem to support a stream
    – Device manager takes care of transfers to or from the medium
    – Stream controller synchronizes talking or listening on 1394

■ Connection management documented
  ➤ Plug control registers (PCR’s) to be part of P1394a

■ Recorded isochronous data format (CIP) documented

■ Isochronous data transforms specified
  ➤ Physical ID substitution upon playback
  ➤ Time stamp modification upon recording and playback
Request types

- Four basic types
  - Management, which includes login (32 bytes)
  - Conventional SCSI (32 and 64 bytes)
  - SCSI stream (64 bytes)
  - Stream control (64 bytes)

- All except management requests may be linked together
  - Working set of requests limited by initiator memory, not by target memory
  - New requests may be appended without interrupting target

- Both SCSI stream and stream control requests must be used together for isochronous
## Login request

- **Login parameters**
  - Login type (conventional, listener or talker)
  - Constant status offset
- **Login data**
  - Fetch agent CSR addresses

---

<table>
<thead>
<tr>
<th>most significant</th>
<th>login_parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>login_data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rq_ctrl</td>
<td>reserved</td>
<td>0</td>
</tr>
<tr>
<td>lun</td>
<td>login_parameters_length</td>
<td>login_data_length</td>
</tr>
<tr>
<td>status_buffer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Login parameters
  - Login type (conventional, listener or talker)
  - Constant status offset

- Login data
  - Fetch agent CSR addresses
32-byte conventional SCSI request

- Up to a 12-byte CDB supported
- Status always returned at a fixed offset from the request block address
  ➤ Offset established at login
64-byte conventional SCSI request
Data transfer

- Application buffer described by data_buffer and data_buffer_length
- If page_size is zero, direct addressing
- If page_size is nonzero, indirect addressing via page table
  - Page table address is in data_buffer
  - Count of page table elements is in data_buffer_length
- In either case, spd and max_payload constrain speed and packet size when the target fetches or stores data in the application buffer
Page table

- Page table is an array of 8-byte elements

- In this example, page size is 4,096 bytes
  - The segment_offset field is \( \text{page\_size} + 8 \) bits wide

- Elements are constructed as follows:
  - All elements, transfer length = \( 2^{\text{page\_size} + 8} - \text{segment\_offset} \)
  - First element, concatenate segment_base and segment_offset to get starting transfer address
  - Second through last elements, concatenate segment_base and least significant bits of zero to get starting transfer address
Completion status

- Fixed-length, 8-byte status block
  
- Overload the `rq_status` field to encode both SBP-2 and SCSI status
  - When `rq_status` equals FF\textsubscript{16}, no SCSI status present

- The `sfmt` field encodes SCSI sense data error code
  - Current, deferred or vendor unique (70\textsubscript{16}, 71\textsubscript{16} or 7F\textsubscript{16})

- The information field is the same as in SCSI sense data
  - Logical block address or transfer residual, as appropriate to device
Fetch agent initialization

- Fetch agent usually initialized with a dummy request
  - Set `next_request` in the dummy request to null
  - Store the dummy request address in `CURRENT_REQUEST`
  - Write an op value of RUN to `AGENT_CONTROL`
- Dummy request has NOP that completes immediately
- Fetch agent pauses and waits for doorbell
  - `CURRENT_REQUEST` still points to `next_request`
- Upon resumption, status returned a second time
Dynamic append to request lists

- Update next_request in previous tail request with address of request to be appended
- Write any value to the DOORBELL register

NOTE: The request blocks are aligned to 32-byte boundaries; the pointers show addresses with a hint subfield that describes a 32-byte data structure that may be fetched at S100
Basic task management

- No untagged tasks
  - Serial Bus address of the request block is the tag

- All tasks are SIMPLE
  - No ORDERED, ACA or HEAD OF QUEUE tasks
  - Task attribute is implicit, not part of request block
  - Configuration ROM identifies task management model

- Limited task management functions
  - ABORT TASK
  - ABORT TASK SET
  - TARGET RESET

- Upon a fault, entire task set is cleared
Isochronous task sets

- Leverage SAM terminology to describe isochronous
- Isochronous task sets are inherently ORDERED
- Multiple isochronous task sets per logical unit
  - Organizing concept is a stream
  - Login ID is the stream identifier
- Two target objects mediate stream data transfers
  - SCSI stream requests direct transfer to/from medium
  - Stream control requests direct transfer from/to 1394
- SCSI stream requests do not have buffer addresses
- Stream control requests support time synchronization
SCSI stream request

```
most significant

next_request

stream_offset

reserved

rq_ctrl  reserved

reserved  stream_length

status_buffer

CDB

reserved

least significant
```
Stream control request

```
+---------------------+-----------------------------------+
| most_significant     | next_request                      |
|                     |                                   |
|                     |                                   |
|                     | error_log                         |
| rq_ctrl             |                                   |
|                     |                                   |
| r                   | spd                               |
| max_payload         |                                   |
| reserved            |                                   |
| page_size           |                                   |
| error_log_length    |                                   |
| status_buffer       |                                   |
| stream_ctrl         |                                   |
| stream_event        | reserved                          |
| reserved            | sy                                |
| error_handling      |                                   |
| reserved            |                                   |
| second_count_hi     | reserved                          |
| reserved            |                                   |
| second_count        |                                   |
| cycle_count         | reserved                          |
| reserved            |                                   |
| stream_ctrl-dependent|                                 |
|                     |                                   |
+---------------------+-----------------------------------+
```
Common isochronous packet format

- CIP format indicated by sy value of one

```
+--------------------------------------------------+
|          most significant             |          least significant          |
| data_length |         | channel | DATA | sy |
| 0           | sid     | dbs     | fn   | qpc | s   | r   | dbc |
| 2           | fmt     |         |      |     |     |     |     |
|             |         |         |      |     |     |     |     |
|             |         |         |      |     |     |     |     |
|             |         |         |      |     |     |     |     |
+--------------------------------------------------+
```

- Replace sid with own physical ID on playback
- If s bit is set, source packet header (time stamp) follows in first quadlet of application data
- For some fmt values, time stamp is present in the fmt-dependent field
Time stamp formats

■ Source packet header format

<table>
<thead>
<tr>
<th>reserved</th>
<th>cycle_count</th>
<th>cycle_offset</th>
</tr>
</thead>
</table>

■ Synchronization time (syt) format

<table>
<thead>
<tr>
<th>fmt</th>
<th>fmt-dependent</th>
<th>cycle_count</th>
<th>cycle_offset</th>
</tr>
</thead>
</table>

■ Both present as absolute time stamps on 1394

■ Convert to relative time stamps when recorded

- \( \text{sph\_time}_{\text{stored}} = (\text{sph\_time}_{\text{observed}} \& 0x1FFF000) - (\text{CYCLE\_TIME} \& 0x01FFF000) \)
- \( \text{syt}_{\text{stored}} = (\text{syt}_{\text{observed}} \& 0xF00) - (\text{CYCLE\_TIME} \& 0x0000F000) \)

■ Reconvert to absolute time stamps upon playback

■ Hardware support recommended
Unit_Characteristics entry

- Configuration ROM entry in unit directory

<table>
<thead>
<tr>
<th>most significant</th>
<th>q</th>
<th>reserved</th>
<th>i</th>
<th>x</th>
<th>e</th>
<th>r</th>
<th>m</th>
<th>least significant</th>
</tr>
</thead>
<tbody>
<tr>
<td>39_{16}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>device_type</td>
</tr>
</tbody>
</table>

- The q bit indicates the task management model
  - Basic or full (today’s SAM model)
- The i bit indicates isochronous support
- The x bit indicates support for 64-byte request blocks
- The e bit is equivalent to RMB (removable media)
- The m bit is clear when device has a single logical unit
  - Logical unit number is zero
  - The device_type field is the same as in INQUIRY data
SAM compliance

- SBP-2 intended to be stand-alone document for implementors
  - SAM and other SCSI standards are normative references, but are not necessary to understand SBP-2
- SBP-2 intended to be useful to implementors who build devices that don’t claim SCSI or SAM compliance
- SAM compliance described in a normative annex
  - In the body of the standard, SBP-2 behavior is specified with minimal recourse to SAM terminology
  - SAM formalisms may be easily referenced in one location
Next steps

- Modify SAM queuing for a new subsets
  - Basic task management
- Working group meetings
  - Develop security proposals
  - Isochronous data requirements
  - August 13 hosted by Microsoft in Redmond, WA
- Document review
  - Schedule editor’s meetings
- Aggressive schedule; stabilize by November