CONGRUENT SOFTWARE, INC.
98 Colorado Avenue
Berkeley, CA 94707
(510) 527-3926
(510) 527-3856 FAX

FROM: Peter Johansson

TO: T10 SBP-3 working group

DATE: June 4, 2002

RE: Bridge-aware SBP-3 target operations

An informative description of bridge-aware target operations has been present in the SBP-3 draft since April 2001, as have descriptions of the revised and new ORBs necessary to support such operations, but the normative text for sections 8 and 10 has not been proposed until now.

Absent from this document are detailed descriptions of target management of remote time-out and the errors that may be encountered in transactions with nodes addressed by global node IDs. I am uncertain whether they merit inclusion in the SBP-3 draft or whether these matters are sufficiently generic to expect draft standard IEEE P1394.1 to cover them. I invite working group discussion.

5.1.4.1 Login ORB

Before any requests that require a *login_ID* or address logical unit fetch agent CSRs can be made of a target, the initiator shall first complete a login procedure that uses the ORB format shown below.

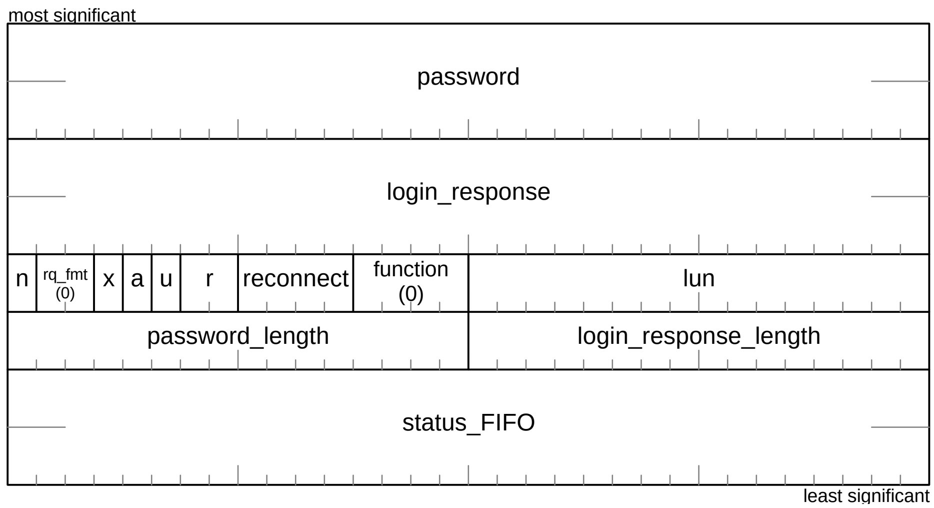

Figure 23 – Login ORB

The *password* and *password_length* fields may contain optional information used to validate the login request. If *password_length* is zero, the *password* field may contain immediate data. When *password_length* is nonzero, the *password* field shall conform to the format for address pointers specified by Figure 11 and shall contain the address of a buffer in the same node as the initiator; consequently the *node_ID* field of this address pointer is reserved. The buffer shall be accessible to a Serial Bus block read request with a data transfer length less than or equal to *password_length*. The format and usage of password data, whether immediate or indirectly addressed, are specified by Annex C.

The *login_response* and *login_response_length* fields specify the address and size of a buffer allocated for the return of the login response. The *login_response* field shall conform to the format for address pointers specified by Figure 11 and shall address the same node as the initiator; consequently the *node_ID* field of this address pointer is reserved. The buffer shall be accessible to a Serial Bus block write request with a data transfer length less than or equal to *login_response_length*. The initiator shall set *login_response_length* to a value of at least 12; the target may ignore this field if it stores no more than 12 bytes of login response data.

The *notify* bit and the *rq_fmt* field are as previously defined for management ORB formats.

The *exclusive* bit (abbreviated as *x* in the figure above) shall specify target behavior with respect to concurrent login to a logical unit. When *exclusive* is zero, the target, subject to its own implementation capabilities, may permit more than one initiator to login to a logical unit. If *exclusive* is one the target shall permit only one login to a logical unit at a time; see 8.2 for a description of target behavior.

The *aware* bit (abbreviated as *a* in the figure above) permits the initiator to request bridge-aware behavior from the target for the duration of this login. When the *aware* bit is zero, target behavior shall be compatible with SBP-2 (see [B1]). Otherwise, the target is requested to behave in a bridge-aware manner compatible with this standard and draft standard IEEE P1394.1.

The *reconnect* field shall specify the desired reconnect time-out as $2^{reconnect}$ seconds. The default reconnect time-out, when *reconnect* is zero, is one second. The target may not be able to support the requested value; see *reconnect_hold* in the login response data below.

The *update* bit (abbreviated as *u* in the figure above), when zero, specifies that the initiator is not logged in to the logical unit identified by *lun*. Otherwise, an existing login is to be updated with the information in the login request.

When the *update* bit is zero, the *lun* field specifies the logical unit number (LUN) to which the request is addressed. Otherwise, the *lun* field shall contain a *login ID* value obtained as the result of a successful login.

The *status_FIFO* field is as previously defined for management ORB formats and shall contain an address allocated for the return of status for the LOGIN request, status for all subsequent requests signaled to the *command_block_agent* allocated for this login and any unsolicited status generated by the logical unit.

If the login fails the contents of the response buffer are unspecified. Otherwise, upon successful completion of a login, the target shall store a minimum of 12 bytes of login response data and may store up to the entire 16 bytes illustrated below so long as the amount of data stored is an integral number of quadlets. Truncated login response data shall be interpreted as if the omitted fields had been stored as zeros.

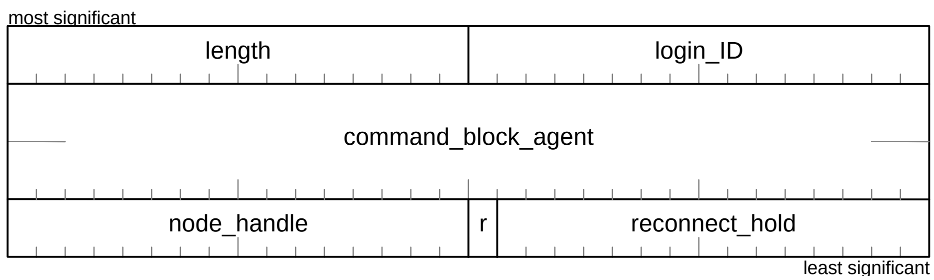

Figure 1 – Login response

The *length* field shall contain the length, in bytes, of the login response data.

The initiator shall use the *login_ID* value returned by the target to identify all subsequent requests directed to the target's management agent that pertain to this login.

The *command_block_agent* field specifies the base address of the agent's CSRs, which are defined in 0. This field shall conform to the format for address pointers specified by Figure 11.

When the *aware* bit in the login request is zero, the contents of the *node_handle* field are unspecified. Otherwise the node handle field shall contain a node handle that the initiator shall use as the most significant 16 bits of any address pointer that references initiator memory in an ORB which is signaled to the target in the context of this login. This requirement applies equally to all address pointers with the exception of those contained in the login request itself.

The *reconnect_hold* field shall specify the time, in seconds less one, that the target will hold resources for a previously logged-in initiator subsequent to a bus reset. The value of *reconnect_hold* shall not be greater than $2^{reconnect}$ -1, where *reconnect* is obtained from the login request. If an initiator fails to complete a successful reconnect request within *reconnect_hold* + 1 seconds after a bus reset, the target will perform a logout and release all resources held by that initiator (see 8.3).

5.3.1 Request status

Upon completion of a request, if the *notify* bit in the ORB is one or if there is exception status to report, the target shall store all or part of the status block shown in Figure 37. For management ORBs (which explicitly provide the *status_FIFO* address as part of the ORB), the target shall store the status block at the address specified. Otherwise (for normal command block ORBs) the target shall store the status block at the *status_FIFO* determined by the fetch agent to which the ORB was signaled. In the case of command block ORBs the initiator provides the *status_FIFO* address as part of the login request while for stream command block and stream control ORBs it is provided in the create stream request.

When *resp* is equal to zero, REQUEST COMPLETE, the possible values for *sbp_status* are specified by the table below. Any value not enumerated is reserved for future standardization.

Value	Description
0	No additional information 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	Reserved for future standardization
8	Resources unavailable
9	Function rejected
10	~~Login ID not recognized~~ Invalid login ID
11	Dummy ORB completed
12	Request aborted
13	Unknown EUI-64
14	~~Node handle not recognized~~ Invalid node handle
FF_{16}	Unspecified error

6.4 Command block and stream control agent registers

Unlike the management agent, which services a single request at a time, the command block and stream control agents manage linked lists from which they fetch requests. For this reason they are referred to as fetch agents. Each fetch agent has a set of control and status registers that lie within the target's units space; the fetch agent CSRs shall be located at or above address FFFF F001 0000_{16} within the node's 48-bit address range.

Although the location of each fetch agent's CSRs is not fixed, the relative relationship of the registers is fixed with respect to each other, as defined by the table below. Implementation of the HEARTBEAT_MONITOR register is optional, but required if the logical unit is bridge-aware ~~*bridge_aware* bit in the target's configuration ROM bus information block is one~~. Implementation of the FAST_START register is optional.

Relative offset	Name	Description
00_{16}	AGENT_STATE	Reports fetch agent state
04_{16}	AGENT_RESET	Resets fetch agent
08_{16}	ORB_POINTER	Address of most recently fetched ORB
10_{16}	DOORBELL	Signals fetch agent to refetch an address pointer
14_{16}	UNSOLICITED_STATUS_ENABLE	Acknowledges the initiator's receipt of unsolicited status
18_{16}	HEARTBEAT_MONITOR	Maintains bridge-aware initiator login during idle periods
$1C_{16} – 3C_{16}$		Reserved for future standardization
vendor-dependent	FAST_START	Signals a reset or suspended fetch agent to start a task; equivalent to a write to the DOORBELL register if the fetch agent is active

The base address of a fetch agent's CSRs is obtained from the appropriate field, *command_block_agent*, *stream_command_block_agent* or *stream_control_agent,* in the response returned by the target as part of a successful login or create stream request.

A target shall ignore or reject Serial Bus request subactions addressed to any of a fetch agent's CSRs unless the *source_ID* matches the node ID of the initiator associated with the login.

6.4.5a HEARTBEAT_MONITOR register

The HEARTBEAT_MONITOR register provides a means by which the initiator signals the target that a reconnect hold time-out period should not be started for the bridge-aware login associated with the register. The definition of this write-only register is given by Figure 44a.

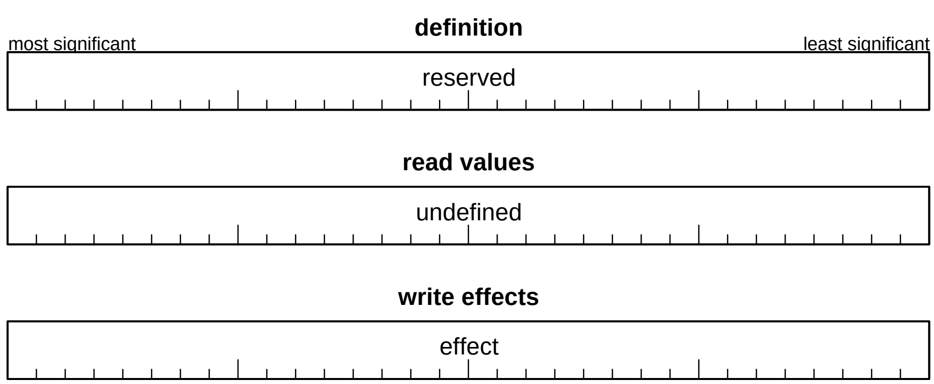

Figure 44a – HEARTBEAT_MONITOR format

Unless the login associated with this register is awaiting reconnection (see 8.5), a quadlet write of any value to this register shall cause the *heartbeat_timeout* variable in the login descriptor associated with the register to be set to the *reconnect_hold* time obtained from the same login descriptor. Otherwise, the target shall reject the request with a type error response.

8 Access

Before an initiator may signal commands to a logical unit or task management requests to a target, access privileges shall first be granted by the target. The criteria for the grant of access may include resource availability or other target requirements. This section specifies the target facilities that support access control and the methods by which an initiator requests access to a logical unit and eventually relinquishes access when it is no longer required.

When an initiator establishes bridge-aware access, it may require additional target resources to manage data transfer operations. This section specifies the methods an initiator uses to obtain or release these resources.

8.1 Access protocols

Targets shall implement a logical unit reservation protocol which may be used to guarantee single initiator access to the logical unit and to preserve that initiator's access rights across a Serial Bus reset. Targets may optionally implement the extensions to the logical unit reservation protocol specified by Annex C, which support both passwords and persistent reservations. Neither of these mechanisms preclude additional, command set-dependent methods that control access to a target.

In order to support the logical unit reservation protocol, a target shall implement resources to manage one or more logins from initiators. These resources are described below and are used in the specification of target actions in response to login requests signaled by an initiator to the target's management agent:

- The target implements a set of one or more *login_descriptors* that are used to hold context for logins. The context of a login stored in a *login_descriptor* consists of the *lun*, the *login_owner_ID*, the *login_owner_EUI_64*, the *status_FIFO* address, an *exclusive* variable, a *bridge_aware* variable, the base addresses of the fetch agent CSRs, the *login_ID* to be used by the initiator to identify the login, the *initiator_node_handle* assigned by the target for a bridge-aware login and the *reconnect_hold* period guaranteed by the target—these last three are returned to the initiator in the *login_response* data.
- The *login_owner_ID* is the 16-bit node ID, either local or global, of the current owner of a login. The target shall use the *login_owner_ID* to qualify the *source_ID* of all write requests addressed to the *login_descriptor* fetch agent CSRs. Upon a power reset, the *login_owner_ID* for all *login_descriptors* shall be set to all ones. ~~Otherwise~~ At all other times, the value of *bridge_aware* determines which events cause *login_owner_ID* to be set to all ones. If *bridge_aware* is false, a Serial Bus reset causes *login_owner_ID* to be initialized; if *bridge_aware* is true, a net update causes *login_owner_ID* to be initialized.
- The *login_owner_EUI_64* is the unique 64-bit identifier of the current owner of a login. Upon a power reset, the *login_owner_EUI_64* for all *login_descriptors* shall be set to all ones. After a Serial Bus reset, for those *login_*descriptors whose *bridge_aware* variable is false, the *login_owner_EUI_64* persists for *reconnect_hold* + 1 seconds and shall then be set to all ones unless the login has been reestablished. After a net update, for those *login_*descriptors whose *bridge_aware* variable is true, the *login_owner_EUI_64* persists for *reconnect_hold* + 1 seconds (measured from the end of quarantine) and shall then be set to all ones unless the login has been reestablished.

A *login_descriptor* is free if both its *login_owner_ID* and *login_owner_EUI_64* are all ones. The resources of this *login_descriptor* may be allocated to any initiator that successfully completes a login request. If a *login_descriptor*'s *login_owner_ID* is all ones but its *login_owner_EUI_64* holds a valid EUI-64, the *login_descriptor* is reserved—the initiator identified by *login_owner_EUI_64* may reestablish the login. Active *login_descriptors* are those whose *login_owner_ID* and *login_owner_EUI_64* are both valid; the initiator that owns the login may signal requests to the fetch agent(s) associated with the *login_descriptor*.

8.2 Access requests

The clauses that follow describe the use of the login and create stream ORBs defined in 5.1.4.1 and 5.1.4.3.

8.2.1 Login

Before an initiator may signal any requests to a target that require a *login_ID* or address fetch agent CSRs, it shall first perform a login. The login request, whose format is specified in 5.1.4.1, shall be signaled to the target's MANAGEMENT_AGENT register by means of an 8-byte block write transaction that specifies the Serial Bus address of the login request. The address of the management agent shall be obtained from configuration ROM.

The speed at which the block write request to the MANAGEMENT_AGENT register is received shall determine the speed used by the target for all subsequent requests to read the initiator's configuration ROM, fetch ORBs from initiator memory or store status at the initiator's *status_FIFO*. Command block ORBs separately specify the speed for requests addressed to the data buffer or page table.

The login ORB shall specify the *lun* of the logical unit for which the initiator desires access.

The target shall perform the following steps, in the order specified, ~~(in any order)~~ to validate a login request:

a) All targets that conform to this standard distinguish between local and global node IDs whether or not they implement bridge-aware capabilities. If the *source_ID* from the write request used to signal the login ORB to the target's MANAGEMENT_AGENT register contains a global node ID but the target does not implement bridge-aware capabilities, the target shall respond with a type error;

NOTE – ~~All targets that conform to this standard distinguish between local and global node IDs whether or not they implement bridge-aware capabilities.~~ Because bridges interdict read requests addressed to global node IDs if they are originated by nodes that are not bridge-aware, a target that is not bridge-aware is unable to fetch the login ORB and determine the address of the *status_FIFO*. Thus, the return of a ~~address~~ type error response is the only method available to notify the initiator that the ~~attempted~~ login failed.

b) Else, If *source_ID* contains a global node ID and the target implements bridge-aware capabilities, the target shall ~~verify that~~ examine the *aware* bit ~~is set~~ in the login ORB and, if zero, shall reject the login with an *sbp_status* of function rejected. When the login specifies a global node ID and the *aware* bit is one, the target shall use a TIMEOUT request, as defined by draft standard IEEE P1394.1, to obtain the EUI-64 of the initiator and its remote timeout information;

c) Otherwise *source_ID* is local and the target shall read the initiator's unique ID, EUI-64, from the bus information block by means of two quadlet read transactions. The *source_ID* from the write transaction used to signal the login ORB to the target's MANAGEMENT_AGENT register shall be used as the *destination_ID* in the quadlet read transactions;

d) In cases where *source_ID* is local, the *aware* bit is set in the login ORB and the target does not implement bridge-aware capabilities, the target shall reject the login request with an *sbp_status* of function rejected;

e) If the *update* bit in the login ORB is zero, the target shall determine whether or not the initiator already owns a login by comparing the EUI-64 just obtained against the *login_owner_EUI_64* for all *login_descriptors*. If the initiator is currently logged-in to the same logical unit, the login request shall be rejected with an *sbp_status* of access denied. Otherwise, when the *update* bit is one, the target shall verify that the initiator owns the login identified by *login_ID* and, if not, shall reject the login request with an *sbp_status* of invalid login ID;

f) If the *exclusive* bit is set in the login ORB and there are any active *login_descriptors* for the logical unit (other than one whose *login_owner_EUI_64* matches the EUI-64 of the initiator requesting the login), the target shall reject the login request with an *sbp_status* of access denied;

g) If an active *login_descriptor* with the *exclusive* attribute exists for the *lun* specified in the login ORB (other than one whose *login_owner_EUI_64* matches the EUI-64 of the initiator requesting the login), the target shall reject the login request with an *sbp_status* of access denied;

h) If the *update* bit in the login ORB is zero, the target shall determine if a free *login_descriptor* is available and, if none are available, reject the login request with an *sbp_status* of resources unavailable. ~~Otherwise, the target shall determine whether or not the initiator already owns a login by comparing the EUI-64 obtained by either a TIMEOUT request or configuration ROM read against *login_owner_EUI_64* for all *login_descriptors*. If the initiator is not logged-in to the logical unit identified by *lun*, the target shall reject the login request with an *sbp_status* of login ID not recognized.~~

If the *update* bit is zero, once the above conditions have been met and a *login_descriptor* allocated, the initiator's *source_ID* is stored in *login_owner_ID*, the initiator's EUI-64 is stored in *login_owner_EUI_64*, the *lun* and *status_FIFO* fields from the login ORB are stored in the *login_descriptor*, the *bridge_aware* and *exclusive* variables in the *login_descriptor* are set to the values of the *aware* and *exclusive* bits, respectively, from the login ORB and the address of the fetch agent and the *reconnect_hold* value chosen by the target are stored in the *login_descriptor*. If the *bridge_aware* variable is true, the target allocates a node handle to the initiator (the process is essentially the same as described by 8.3.1) and stores it in *initiator_node_handle*. Lastly the target assigns a unique *login_ID* to this login and stores it in the *login_descriptor*.

When the *update* bit is one, the login request permits the initiator to change parameters associated with the login. If the login request meets all of the validation requirements described above, the target shall ~~perform a~~ logout ~~for~~ the initiator (see 8.6) without returning completion status and then shall process the login in accordance with the requirements of this clause, 8.2.1. The target shall perform these two steps such that no other initiator is afforded an opportunity to login between the time that target resources have been released ~~by the implicit logout~~ and the time the update login request completes.

If the target is able to satisfy the login request, it shall return a login response as specified in 5.1.4.1. A critical component of a login response returned to the initiator is the base address of the logical unit fetch agent that the initiator shall use to signal any subsequent requests to the target for the indicated *login_ID*.

NOTE – When the *update* bit in the login ORB is one, the *login_ID* value assigned by the target may differ from that previously associated with the login.

8.2.2 Create stream

An isochronous stream may be created for an initiator only after completion of the login process just described. The initiator shall supply a *login_ID* previously obtained as the result of a successful login as well as other information in the create stream request that characterizes the isochronous operations to be performed.

The information consists of four items:

- whether the target is to function as a talker or a listener;
- the isochronous data format;
- the maximum number of channels that may be simultaneously enabled; and
- the aggregate maximum isochronous payload for all channels to be transferred between Serial Bus and the device medium in a single isochronous period.

The aggregate maximum isochronous payload is the worst-case amount of data the target may have to transfer to or from Serial Bus and from or to the medium in an isochronous period. Implementation-dependent constraints may limit the performance of the target, which requires this information in order to determine if the login may be accepted. Upon playback (when the target is a talker), the aggregate maximum isochronous payload shall reflect the total of all channels recorded on the medium—not just the aggregation of payload(s) for the channels to be transmitted on Serial Bus. This is essential since the target reads all of the data from the medium even though the channel mask may select a small subset for playback.

These parameters—listener *vs.* talker, isochronous data format, maximum channels and aggregate maximum isochronous payload—may be used by the target to determine if sufficient resources are available to create the stream and, if so, the manner in which they are to be configured.

The target shall perform the following to validate a create stream request:

– The target shall validate the *login_ID* supplied in the create stream ORB by comparing the *destination_ID* in the read request(s) used to fetch the ORB with the *source_ID* retained when *login_ID* was assigned to the initiator. If the node IDs do not match, the *login_ID* is invalid.

  If the *login_ID* is valid, the target shall determine if a free *stream_descriptor* is available and, if none are available, reject the create stream request with an *sbp_status* of resources unavailable.

Once the above conditions have been met and a *stream_descriptor* allocated, the *stream_descriptor* is associated with the appropriate *login_descriptor* and the addresses of the fetch agent(s) are stored in the *stream_descriptor*. Lastly the target assigns a unique *stream_ID* to this stream and stores it in the *stream_descriptor*.

NOTE – The *stream_descriptor* may also hold other information from the create stream request, such as listener *vs.* talker, isochronous data format, number of channels or aggregate maximum payload as dictated by the target implementation.

In addition to the addresses of the stream command block and stream control fetch agents, the target shall also specify in the *create_stream_response* data the minimum transfer length that the initiator should specify in the *stream_length* field of any stream command block request signaled to the target.

8.3 Node handles

When a bridge-aware login is established, the target returns a node handle in the response data; the initiator uses the node handle in data descriptors in ORBs or page tables that refer to initiator system memory. If necessary to address node(s) other than the initiator itself in data descriptors, the initiator shall first obtain node handle(s) from the target by means of a node handle request. When an initiator no longer requires one or more node handles, it should release them. All node handles for a login are automatically released upon logout.

Bridge-aware targets shall support, at a minimum, the allocation of a node handle that occurs upon successful completion of a login whose *aware* bit is one. Support for the node handle request is optional; without it, bridge-aware target operations are possible so long as all data buffers are located in the initiator.

The function of an individual node handle ORB is encoded by its *allocate* bit. When *allocate* is one, the target is requested to allocate a node handle; otherwise it is requested to release one or ~~more~~ all node handles (except the initiator's own node handle).

Before processing any node handle request, the target shall verify that the *login_descriptor* identified by *login_ID* in the node handle request is active; if not, the target shall reject the node handle request with an *sbp_status* of login ID not recognized.

Node handles shall be unique within the context of a login. An initiator shall not use a node handle obtained for one login in the context of another login, but if necessary shall obtain an additional node handle for the other login even though the node handle references the same node.

8.3.1 Node handle allocation

The target shall perform the following steps to process a node handle allocation request:

- If the least significant six bits of *global_node_ID* are all ones, the target shall reject the node handle request with an *sbp_status* of unknown EUI-64;

- In cases where a node handle exists within the login for the node identified by *eui_64*, the target shall update the *global_node_ID* in the *node_handle_descriptor* with the new value provided in the node handle request, mark the node handle descriptor to indicate that the correlation between *eui*_64 and *global_node_ID* is unverified and then skip the remaining steps described below;

- Otherwise, the target shall determine if the resources to create a node handle are available and, if not, reject the node handle request with an *sbp_status* of resources unavailable; else
- Once these verification steps have succeeded, the target shall store the *eui*_64 and *global_node_ID* from the node handle request in the *node_handle_descriptor* for the allocated node handle. The node handle descriptor shall also indicate that the correlation between *eui*_64 and *global_node_ID* is unverified.

Once the target has associated the global or local node ID and the EUI-64 in the node handle request with a node handle, the target shall store the assigned node handle in the *node_handle_response* buffer provided by the initiator.

Until the node handle is released, the target shall maintain information in a *node_handle_descriptor* that includes, at a minimum, the *node_handle*, the *login_ID* with which the node handle is associated, the *eui_64* of the node, the current global or local *node_ID* for the node and, for global node IDs, the remote time-out for the node. The remote time-out value need not be available when the node handle request completes; the target may defer determination of remote time-out (see NOTE –).

8.3.2 Node handle release

When a target fetches a node handle request whose *allocate* bit is zero, it shall release one or more node handles associated with the login identified by *login_ID*. If *node_handle* equals $FFFF_{16}$, all node handles associated with the login (except the initiator's own node handle) shall be released. If *node_handle* identifies the node handle returned by a login response, the target shall reject the node handle request with an *sbp_status* of invalid node handle. Otherwise, the node handle identified by *node_handle* shall be released.

If *node_handle* does not match any node handle associated with the login identified by *login_ID*, the target shall reject the node handle request with an *sbp_status* of ~~node handle not recognized~~ invalid node handle.

8.3.3 Node handle update after bus reset

Upon a Serial Bus reset, all of a target's active *node_handle_descriptors* whose *node_ID* field contains a local node ID shall be updated with the local node ID currently valid for the associated EUI-64. The target shall obtain this information from its own analysis of self-ID packets observed subsequent to bus reset and the previous bus topology map. The bus topology map correlates EUI-64 with physical ID for nodes on the local bus.

If a local node for which a node handle is allocated is disconnected, the *node_ID* field in its node handle descriptor shall be set to $FFFF_{16}$; this causes an ~~address~~ error if the node handle is present in a command block ORB subsequently signaled to the target.

NOTE – The initiator, which is also aware of the disconnection of the local node, should update the node handle information with a node handle request if the node is subsequently reconnected to the local bus.

8.3.4 Node handle validation after net update

Net update begins when a target's ~~NET_UPDATE.*orphan*~~ QUARANTINE.*orphan* bit changes from ~~one to~~ zero to one, at which time all of its active *node_handle_descriptors* whose *node_ID* field contains a global node ID shall be invalidated. Before the target may execute an ORB that references an invalid node handle it shall reestablish a correlation between the EUI-64 associated with the node handle and a current local or global node ID and obtain the remote time-out value for the bus to which the node is connected.

Node handles assigned to initiators as a consequence of a bridge-aware login are revalidated as part of the reconnection process (see 8.3.2).

Node handles allocated by node handle requests shall be revalidated as follows:

- The target may use one of two methods to revalidate the correlation of an EUI-64 with a global node ID. If the target has already determined remote time-out for the bus ID specified by the global node ID, it may use either a TIMEOUT request addressed to the global node ID or it may use two quadlet read requests to obtain the node's EUI-64 from configuration ROM. If the remote time-out is not known for the bus ID, the target shall use a TIMEOUT request to determine both the EUI-64 and remote time-out;

- If there is no successful response to either the TIMEOUT request or attempted read of configuration ROM or if the EUI-64 obtained does not match the EUI-64 expected, the target shall set the *node_ID* field in the node handle descriptor to $FFFF_{16}$; this causes an ~~address~~ error if the node handle is present in a command block ORB subsequently signaled to the logical unit fetch agent; else

- Otherwise, if the EUI-64 obtained matches that associated with the node handle prior to net update, the target shall mark store the remote time-out value in the *node_handle_descriptor* and mark the node handle valid.

Although a target may revalidate all its node handles when its ~~NET_UPDATE.*orphan*~~ QUARANTINE.*orphan* bit changes from one to zero, ~~or~~ it should elect a "lazy" scheme and defer revalidation until an invalid node handle is encountered in a command block ORB signaled to a target logical unit. If an ORB signaled to a logical unit fetch agent contains a node handle whose global node ID in the *node_handle_descriptor* is equal to $FFFF_{16}$, the task set shall be aborted and an *sbp_status* of unknown EUI-64 shall be reported for the task that caused the abort.

8.4 Heartbeat

A "heartbeat" is a periodic signal from an initiator to a target logical unit; it's purpose is to maintain an active, bridge-aware login for the initiator. Absent such a signal, the target, after a time-out period, commences actions that eventually logout the ~~unresponsive~~ initiator and free target resources.

Whenever a task set becomes empty (this occurs immediately after a successful login or reconnect as well as when a task set's work completes or is aborted) and the owner of the login associated with the task set is identified by a global node ID, the target logical unit starts a timer, *heartbeat_timeout*. The timer is initially set to the *reconnect_hold* time reported to the initiator for the login; it counts down to zero. So long as the task set remains empty, the timer requires periodic refresh by a signal from the initiator. So long as the task set is empty, a request subaction whose *source_ID* matches the node ID of the initiator associated with the login and which is addressed to ~~any of the fetch agent CSRs~~ the

HEARTBEAT_MONITOR register shall cause the target logical unit to reinitialize the value of *heartbeat_timeout* to *reconnect_hold* seconds. ~~The HEARTBEAT_MONITOR register provides a destination address for a write request that may be used when a different destination address, *e.g.*, that of the DOORBELL register, would be inappropriate.~~

When an ORB is signaled to a logical unit fetch agent whose associated task set is empty, the *heartbeat_timeout* timer shall be stopped. Once the task set again becomes empty, the timer shall be initialized to *reconnect_hold* seconds and the heartbeat time-out process shall resume.

If the *heartbeat_timeout* timer decrements to zero while the logical unit task set is empty, the target shall commence a reconnect hold period for the login as described in 8.5. Once a target commences a reconnect hold period, it shall ~~ignore or~~ reject, with a response of type error, Serial Bus request subactions addressed to any of the fetch agent CSRs associated with the login.

8.5 Reconnection

Upon a Serial Bus reset, the target shall abort all task sets for all command block agents created as the result of login request(s) whose *aware* bit was zero. The target shall also determine whether or not net update is active; all net update conditions are signaled by bus reset but not all bus resets signal net update. If net update is not active, task sets created as the result of login request(s) whose *aware* bit was one shall not be aborted. Otherwise, when net update is active, the target shall abort all task sets for all command block agents.

There is a special case that applies to logins that were created as the result of login requests whose *aware* bit was one and for which the initiator is connected to the target's local bus. This condition may exist whether or not bridge portals are connected to the local bus. If a local initiator is the owner of a bridge-aware login and is disconnected from the bus, all its task sets shall be aborted.

If the primary task set for a login is aborted for any of the causes cited above, the task sets for any associated isochronous streams are affected as follows. Both stream command block and stream control requests fetched prior to the bus reset shall continue to be executed by the target but the return of status shall be deferred until a successful reconnection occurs. Stream command block and stream control requests shall not be fetched until the login associated with the stream is successfully reconnected.

For each login whose task set was empty at the expiration of a heartbeat timer or whose task set was aborted by bus reset, disconnection of a local initiator or net update ~~or expiration of a heartbeat timer~~, the target shall retain, for at least *reconnect_hold* + 1 seconds subsequent to the trigger event—either a bus reset, disconnection of a local initiator, net update or expiration of a heartbeat timer—sufficient information to permit an initiator to reconnect its login (and, implicitly, any associated streams). After this time, but within *reconnect_hold* + 2 seconds, the target shall perform an implicit logout for each expired login ID or stream ID that has not been successfully reconnected to its original initiator. The *reconnect_hold* parameter is communicated from the target to the initiator as part of the login response data. If the trigger event is a bus reset or disconnection of a local initiator, the time-out commences when the target observes the first subaction gap subsequent to a bus reset. If a bus reset occurs before the time-out expires, the timer is zeroed then restarted upon detection of a subaction gap. If the trigger event is net update, the time-out commences when the target's ~~NET_UPDATE.*orphan*~~ QUARANTINE.*orphan* bit changes from one to zero. If ~~NET_UPDATE.*orphan*~~ QUARANTINE.*orphan* changes from zero to one before the time-out expires, the timer is zeroed then restarted upon the next transition of ~~NET_UPDATE.*orphan*~~ QUARANTINE.*orphan* from one to zero. Otherwise, if the trigger event is the lack of a heartbeat, the time-out commences upon the heartbeat timer's expiration. In ~~this~~ the last two cases, if a net update commences before the *reconnect_hold* period elapses, the timer is zeroed, restarted upon the next transition of ~~NET_UPDATE.*orphan*~~ QUARANTINE.*orphan* from one to zero and thereafter the reconnect hold time-out is managed as described for a net update trigger event.

NOTE – The rationale for a reconnect hold period of at least one second subsequent to bus reset is to permit initiators on the same bus as the target to reallocate isochronous channels and bandwidth and to reestablish isochronous connections.

After a task set is aborted by bus reset or net update, an initiator shall not signal requests for an otherwise valid login until it first performs a reconnect. The reconnect request, whose format is specified in 5.1.4.3, shall be signaled to the target's MANAGEMENT_AGENT register by means of an 8-byte block write transaction that specifies the Serial Bus address of the reconnect ORB. The address of the management agent is that previously obtained by the initiator from the target's configuration ROM.

The speed at which the block write request to the MANAGEMENT_AGENT register is received shall determine the speed used by the target for subsequent requests to read the initiator's configuration ROM, fetch ORBs from initiator memory or store status at the initiator's *status_FIFO*. This replaces the speed most recently obtained from the prior login or reconnect request.

The target shall perform the following to validate a reconnect request:

– If the *source_ID* from the write transaction used to signal the login ORB to the target's MANAGEMENT_AGENT register contains a global node ID, the target shall use a TIMEOUT request, as defined by draft standard IEEE P1394.1, to verify the EUI-64 of the reconnecting initiator and to obtain its remote timeout information;

– Otherwise, the target shall read the initiator's unique ID, EUI-64, from the bus information block by means of two quadlet read transactions. The *source_ID* from the write transaction used to signal the reconnect ORB to the target's MANAGEMENT_AGENT register shall be used as the *destination_ID* in the quadlet read transactions;

  The target shall determine whether or not the *login_ID* is valid by comparing the just obtained EUI-64 against the *login_owner_EUI_64* for the *login_descriptor* identified by *login_ID*;

If the *login_ID* is valid, the target shall update *login_owner_ID* in the referenced *login_descriptor* (and in all stream descriptors associated with the same initiator) with the initiator's *source_ID*.

Fetch agents for stream command block and stream control requests for the reconnected initiator may resume; status for completed ORBs that had not been stored in the initiator's *status_FIFO* (because the initiator's *source_ID* had been invalidated by the bus reset) may also be stored.

Upon successful completion of a reconnect request, the fetch agent associated with *login_ID* shall be in the reset state; the state of the fetch agent(s), if any, for the streams dependent upon *login_ID* is not affected by the reconnect request. No *login_response* data is stored for a reconnect request; the completion status is indicated by the status block stored at the *status_FIFO* address.

8.6 Logout

When an initiator no longer requires access to a target's resources, it shall signal a logout request to the management agent. The login or stream resources to be released shall be identified by *login_ID* in the logout ORB. A target shall reject a logout request if *login_ID* does not match that of any active *login_descriptor* or if the *source_ID* of the write request used to signal the logout ORB to the MANAGEMENT_AGENT register is not equal to the *source_ID* of the matching *login_descriptor*. A logout whose *login_ID* was obtained as the result of a login request implicitly causes the release of all node handles associated with *login_ID* and the logout of all streams associated with *login_ID*. Any tasks or stream control ORBs active at the time of the logout request shall be aborted in the same fashion as if the task set had been aborted. Upon successful completion of a logout request, all resources allocated to the initiator are free once again and may be used by the target to satisfy subsequent login or create stream requests.

10.5 Task management event matrix

Common events that affect the state of target fetch agents and their associated task set(s) are summarized below. Refer to the governing clauses in sections 8 and 9 as well as this section for detailed information.

Event	AGENT_STATE.*st* Normal	AGENT_STATE.*st* Stream	Task set(s) Normal	Task set(s) Stream
Power reset	RESET		Clear all task sets	
Command reset (write to RESET_START)	RESET		Clear all task sets	
Bus reset	RESET (logins that are not bridge-aware)	—	Clear all task sets that are not bridge-aware	—
Net update	RESET (logins that are bridge-aware)	—	Clear all bridge-aware task sets	—
Reconnect hold expires	—		Logout the initiator that has failed to successfully reconnect	
Login	—		—	
Create stream	—		—	
Reconnect	—		—	
Logout	RESET		Abort initiator's task set	
Fetch agent reset (write to AGENT_RESET)	RESET		Abort initiator's task set	
Faulted command (CHECK CONDITION)	DEAD		Abort faulted initiator's task set	
ABORT TASK	—		—	
ABORT TASK SET	DEAD		Abort specified task set	
CLEAR TASK SET	DEAD		Clear all task sets	
LOGICAL UNIT RESET	DEAD		Abort all the logical unit's task sets	
TARGET RESET	DEAD		Clear all task sets	
TERMINATE TASK	—		—	

When an event affects more than one task set, all of the associated fetch agents transition to the state indicated by the table. With respect to events supported by the target's management agent, *e.g.*, logout, there is an assumption of successful completion. In the case of a function rejected response or other indication of failure, the preceding table does not apply.

Bus resets affect target fetch agents and task sets according to the kind of request, login or create stream, by which the initiator first acquired access privileges. A login request allocates normal command block resources while a create stream request allocates stream command block and stream control resources.

Immediately upon detection of a bus reset, all normal command block fetch agents for logins without the *bridge_aware* attribute transition to the reset state and their associated task sets are cleared without the return of completion status. The operations of normal command block fetch agents for logins with the *bridge_aware* attribute are paused until the node IDs for any node handles that refer to nodes on the local bus are updated to reflect changes in physical ID caused by bus reset; once this is complete, fetch agent operations resume without clearing the task set. Associated stream command block and stream control fetch agents do not fetch any additional ORBs subsequent to a bus reset but otherwise preserve their state. The task sets associated with these stream agents continue execution, but status for completed commands is held by the target and not stored to the initiator's *status_FIFO*.

Immediately upon detection of a net update, all normal command block fetch agents for logins with the *bridge_aware* attribute transition to the reset state and their associated task sets are cleared without the return of completion status. Associated stream command block and stream control fetch agents do not fetch any additional ORBs subsequent to a net update but otherwise preserve their state. The task sets associated with these stream agents continue execution, but status for completed commands is held by the target and not stored to the initiator's *status_FIFO*.

For *reconnect_hold* + 1 seconds subsequent to a bus reset, net update or missed heartbeat, targets save state information for initiators that were logged-in at the time of the event. For bus reset, the timer commences when the target observes the first subaction gap subsequent to a bus reset; if a bus reset occurs before the timer expires, the timer is reset. For net update, the time-out commences when the target's ~~NET_UPDATE.*orphan*~~ QUARANTINE.*orphan* bit changes from one to zero; if the *orphan* bit changes from zero to one before the time-out expires, the timer is zeroed and restarted when the *orphan* bit is once again zeroed. Otherwise, for missed heartbeat, the time-out commences upon the heartbeat timer's expiration; if net update commences before the *reconnect_hold* the time-out expires, the timer is zeroed, restarted upon the next transition of ~~NET_UPDATE.*orphan*~~ QUARANTINE.*orphan* from one to zero and thereafter the time-out is managed as described for net update. If an initiator successfully completes a reconnect request during this period, the actions described in 8.3 occur. For normal command block requests, the task set is empty and, once the fetch agent is initialized, the initiator may signal new ORBs to the target. For both stream command block and stream control agents, fetching operations resume from the same point as before the bus reset. Any completion status held by the target during this one second period may also be stored to the initiator's *status_FIFO* after the successful reconnection.

Once *reconnect_hold* + 1 seconds have elapsed after a bus reset, net update or missed heartbeat, the target shall automatically perform a logout operation for all login IDs and stream IDs that have not been reconnected with their initiator. This returns all the affected fetch agents to the reset state and aborts any associated stream task sets.

C.2 Login

The description of the login protocol below reproduces that specified by section 0 and adds validation of cumulative login attempts and the *password* field from the login request. The target shall implement an internal counter, *login_attempts*, which shall be zeroed upon a power reset or upon a successful login or logout request. The target shall perform the following, in the order specified, ~~(in any order)~~ to validate a login request:

a) If the *source_ID* from the write request used to signal the login ORB to the target's MANAGEMENT_AGENT register contains a global node ID but the target does not implement bridge-aware capabilities, the target shall respond with a type error;

NOTE – All targets that conform to this standard distinguish between local and global node IDs whether or not they implement bridge-aware capabilities. Because bridges interdict request subactions addressed to global node IDs if they are originated by nodes that are not bridge-aware, the return of a type error response is the only method available to notify the a bridge-aware initiator that the attempted login failed.

b) Else, If *source_ID* contains a global node ID and the target implements bridge-aware capabilities, the target shall verify that examine the *aware* bit is set in the login ORB and, if zero, shall reject the login with an *sbp_status* of function rejected. When the login specifies a global node ID and the *aware* bit is one, the target shall use a TIMEOUT request, as defined by draft standard IEEE P1394.1, to obtain the EUI-64 of the initiator and its remote timeout information;

c) The target shall reject the login request if *login_attempts* is equal to three;

d) Otherwise *source_ID* is local and the target shall read the initiator's unique ID, EUI-64, from the bus information block by means of two quadlet read transactions. The *source_ID* from the write transaction used to signal the login ORB to the target's MANAGEMENT_AGENT register shall be used as the *destination_ID* in the quadlet read transactions;

e) In cases where *source_ID* is local, the *aware* bit is set in the login ORB and the target does not implement bridge-aware capabilities, the target shall reject the login request with an *sbp_status* of function rejected but *login_attempts* shall not be incremented;

f) If the *update* bit in the login ORB is zero, the target shall determine whether or not the initiator already owns a login by comparing the EUI-64 just obtained against the *login_owner_EUI_64* for all *login_descriptors*. If the initiator is currently logged-in to the same logical unit, the login request shall be rejected with an *sbp_status* of access denied but *login_attempts* shall not be incremented. Otherwise, when the *update* bit is one, the target shall verify that the initiator owns the login identified by *login_ID* and, if not, shall reject the login request with an *sbp_status* of invalid login ID but *login_attempts* shall not be incremented;

g) The target shall validate the password provided by the login request. If *password_length* is zero, the password is eight bytes of immediate data present in the *password* field. Otherwise *password_length* specifies the size of the password addressed by *password*. If *password_length* is greater than 28 the target shall increment *login_attempts* and reject the login request with an *sbp_status* of access denied. When *password_length* is valid, the password provided is extended to 28 bytes by the addition of least significant bytes of zeros; the result is compared with the target's passwords. If the password provided matches neither the target's current nor its master password, the *login_attempts* count shall be incremented and the login request shall be rejected with an *sbp_status* of access denied;

h) If the *exclusive* bit is set in the login ORB, the target shall reject the login request (with an *sbp_status* of access denied) if there are any active *login_descriptors* for the logical unit (other than one whose *login_owner_EUI_64* matches the EUI-64 of the initiator requesting the login) but shall not increment *login_attempts*;

i) If an active *login_descriptor* with the exclusive attribute exists for the lun specified in the login ORB (other than one whose *login_owner_EUI_64* matches the EUI-64 of the initiator requesting the login), the target shall reject the login request (with an *sbp_status* of access denied) but shall not increment *login_attempts*; else

j) If the *update* bit in the login ORB is zero, the target shall determine if a free *login_descriptor* is available and, if none are available, reject the login request with an *sbp_status* of resources unavailable. Otherwise, the target shall determine whether or not the initiator already owns a login by comparing the EUI-64 obtained by either a TIMEOUT request or configuration ROM read against *login_owner_EUI_64* for all *login_descriptors*. If the initiator is not logged-in to the logical unit identified by *lun*, the target shall reject the login request with an *sbp_status* of invalid login ID.

If the *update* bit is zero, once the above conditions have been met and a *login_descriptor* allocated, the initiator's *source_ID* is stored in *login_owner_ID*, the initiator's EUI-64 is stored in *login_owner_EUI_64*, the *lun* and *status_FIFO* fields from the login ORB are stored in the *login_descriptor*, the *bridge_aware* and *exclusive* variables in the *login_descriptor* are set to the values of the *aware* and *exclusive* bits, respectively, from the login ORB and the address of the fetch agent and the *reconnect_hold* value chosen by the target are stored in the *login_descriptor*. If the *bridge_aware* variable is true, the target allocates a node handle to the initiator (the process is essentially the same as described by 8.3.1). Lastly the target assigns a unique *login_ID* to this login and stores it in the *login_descriptor*.

When the *update* bit is one, the login request permits the initiator to change parameters associated with the login. If the login request meets all of the validation requirements described above, the target shall logout the initiator (see 8.6) without returning completion status and then shall process the login in accordance with the requirements of this clause, 8.2.1. The target shall perform these two steps such that no other initiator is afforded an opportunity to login between the time that target resources have been released and the time the update login request completes.

If the target is able to satisfy the login request, it shall zero *login_attempts* and return a login response as specified in 5.1.4.1. A critical component of a login response returned to the initiator is the base address of the logical unit fetch agent that the initiator shall use to signal any subsequent requests to the target for the indicated *login_ID*. When the *update* bit in the login ORB is one, the target may return a *login_ID* value different from that previously associated with the login.