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: February 7, 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.

5.3.1 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
11	Dummy ORB completed
12	Request aborted
13	Unknown EUI-64
14	Node handle not recognized
FF_{16}	Unspecified error

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 between the target and nodes other than the initiator itself. 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 **Error! Reference source not found.**, 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 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. ~~Upon either a Serial Bus reset or a power reset, the *login_owner_ID* for all *login_descriptors* shall be reset to all ones.~~ The target shall use the *login_owner_ID* to qualify the *source_ID* 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, 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 generation change 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 ~~it~~ the login has been reestablished. After a net generation change, for those *login*_descriptors whose *bridge*_aware variable is true, 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.

A *login_descriptor* is considered 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 any order) to validate a login 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 obtain remote timeout information for the initiator identified by *source_ID*;[1]
- 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;

TO BE DETERMINED – The strategy of two separate quadlet reads of the parts of the EUI-64 is a legacy from when we thought hardware would be unable to implement block reads of the EUI-64. Is there some way we can move on?

  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;
- If 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;
- If the *exclusive* bit is set in the login ORB and there are any active *login_descriptors* for the logical unit, the target shall reject the login request with an *sbp_status* of access denied;
- If an active *login_descriptor* with the *exclusive* attribute exists for the *lun* specified in the login ORB, the target shall reject the login request with an *sbp_status* of access denied; else
- 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.

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*

[1] Targets compliant with this standard are not required to implement bridge-aware capabilities, but they should be able to distinguish between local and global node IDs. A target without bridge-awareness should respond with an address error to an attempted write to its MANAGEMENT_AGENT register by a remote node.

fields from the login ORB are stored in the *login_descriptor*, the *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 *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*.

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 ~~target~~ logical unit fetch agent that the initiator shall use to signal any subsequent requests to the target for the indicated *login_ID*.

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) 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.

The function of an individual node handle ORB is encoded by its *node_handle* field. When *node_handle* is zero, the target is requested to allocate a node handle; otherwise it is requested to release one or more node handles.

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

8.3.1 Node handle allocation

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

- The target shall determine if a node handle is free[2] and, if none are available, reject the node handle request with an *sbp_status* of resources unavailable; else
- If the node identified by *eui_64* is connected to the target's local bus, the target shall associate that node's current local node ID with the node handle and skip the remaining steps below. The target shall maintain an accurate bus topology map which includes the correlation between physical ID and EUI-64, as described in draft standard IEEE P1394.1;
- Otherwise, for remote nodes, the target shall establish a correlation between the *eui_64* provided by the initiator and a currently valid global node ID. If the *global_node_ID* in the node handle request is other than all ones, a hint is provided. The target should use a TIMEOUT request, as defined by draft standard IEEE P1394.1, to determine whether or not the global node ID and EUI-64 correlate;
- If *global_node_ID* is all ones or if the EUI-64 obtained by a TIMEOUT request differs from *eui_64* in the node handle request, the target shall use a DEP EUI-64 discovery request, as defined by draft standard IEEE P1394.1, to determine the global node ID of the remote node identified by *eui_64*;

  The target shall time outstanding EUI-64 discovery requests and if no response is received within an implementation-dependent time limit, reject the node handle request with an *sbp_status* of unknown EUI-64. The time-out limit for EUI-64 discovery requests shall not be less than specified by *mgt_ORB_timeout* in the configuration ROM Unit_Characteristics entry (see 7.7.9);

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

[2] Node handles are opaque to the initiator; a target may manage these resources in any manner so long as a node handle is unique within the context of each login.

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. Dependent upon the method used to establish a correlation between EUI-64 and a global node ID, the remote time-out value may not be available when the node handle request; the target may defer determination of remote time-out (see TBD).

8.3.2 Node handle release

When a target fetches a node handle request whose *node_handle* field is nonzero, 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 shall be released. 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.

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 a bus topology map it maintains. 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 descriptor shall be set to $FFFF_{16}$; this causes and address error if the node handle is present in a command block ORB subsequently signaled to the target.

8.3.4 Node handle validation after net update

Net update begins when a target's NET_UPDATE.*orphan* bit changes from one to zero, 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 contains an invalid node handle it shall reestablish a correlation between the EUI-64 associated with the node handle and a current global node ID.

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:

- If the target has retained a global node ID that correlated with the node handle EUI-64 prior to net update, it should use a TIMEOUT request to revalidate the global node ID. A successful response to the TIMEOUT request addressed to a global node ID contains the EUI-64 of the node and the remote time-out for the node.
- If the target does not have a previously valid global node ID or if the TIMEOUT request was not successful, the target shall attempt to correlate the node handle EUI-64 with a global node ID by means of a DEP EUI-64 discovery request.

  The target shall time outstanding EUI-64 discovery requests and if no response is received within an implementation-dependent time limit, set the *node_ID* field in the *node_handle_descriptor* to $FFFF_{16}$;

Once the target has established a correlation between a global node ID and the EUI-64 for the node handle, it shall store the global node ID in the *node_handle_descriptor* and mark the node handle valid.

A target may revalidate all its node handles when its NET_UPDATE.*orphan* bit changes from one to zero or it may 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.

8.4 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. Task sets created as the result of login request(s) whose *aware* bit was one ~~associated with isochronous streams~~ shall not be aborted.

In a complementary fashion, upon net update, the target shall abort all task sets for all command block agents created as the result of login request(s) whose *aware* bit was one. Task sets created as the result of login request(s) whose *aware* bit was zero shall not be aborted.

If the login associated with an isochronous stream is aborted for either of the causes cited above, the task sets for any such 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 aborted, the target shall retain, for at least *reconnect_hold* + 1 seconds subsequent to the trigger event—either a bus reset or a net update—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 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, 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. Otherwise, if the trigger event is net update, the time-out commences when the target's NET_UPDATE.*orphan* bit changes from one to zero. If NET_UPDATE.*orphan* changes from zero to one before the time-out expires, the timer is zeroed then restarted upon detection of a subaction gap.

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. ~~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.~~

After a ~~Serial Bus reset~~ task set is aborted by bus reset or net update, an initiator should 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 all 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 obtain remote timeout information for the initiator identified by *source_ID*;

- 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*. If the *aware* variable in the *login_descriptor* is true, the logical unit shall mark all node handles assigned to the initiator (except the initiator's own node handle) so that their associated global node IDs are revalidated before use in data transfer operations.

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.

8.5 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 **Error! Reference source not found.** as well as this section for detailed information.

Event	AGENT_STATE.*st*		Task set(s)	
	Normal	Stream	Normal	Stream
Power reset	RESET		Clear all task sets	
Command reset (write to RESET_START)	RESET		Clear all task sets	
Bus reset (immediate)	RESET	—	Clear all task sets that are not bridge-aware	—
Bus reset (after *reconnect_hold* + 1 seconds)	—		Logout any initiator that has failed to successfully reconnect	
Net update (immediate)	RESET	—	Clear all bridge-aware task sets	—
Net update (after *reconnect_hold* + 1 seconds)	—		Logout any 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 initiator's 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 *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 *aware*

attribute are paused until the node IDs for any node handles that refer to nodes on the local bus are updated to reflect possible changes in physical ID caused by bus reset; once this is complete, fetch agent operations resume without clearing the task set. 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*.

For *reconnect_hold* + 1 seconds subsequent to a bus reset or net update, targets save state information for initiators that were logged-in at the time of the event ~~bus reset~~. 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. Otherwise, for net update, the time-out commences when the target's NET_UPDATE.*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 then restarted when the *orphan* bit is once again zeroed. 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 or net update, 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.