9.5 Error recovery

A target aborts task set when target detects any of following conditions;

- Bus reset
- TARGET_RESET management function
- AGENT_RESET register write
- ABORT_TASK_SET management function
- LOGICAL_UNIT_RESET management function
- Unrecoverable transaction error

Upon these conditions, the initiator shall take an appropriate action to maintain a login and resynchronize connections as described below.

9.5.1 Bus reset

Upon a Serial Bus reset, the target aborts all task sets for all command block agents created as result of login request(s). For each login, the target retains, for at least *reonnect_hold*+1 seconds subsequent to a bus reset, sufficient information to permit an initiator to reconnect its login and awaits reconnection from initiator(s) active prior to the reset. Upon successful completion of a reconnect request, the fetch agent associated with *login_ID* is in the reset state.

Once an initiator successfully completes a RECONNECT with the target, initiator resumes data transfer with target on a connection by connection basis by re-synchronizing a connection(s) (See 9.5.6).

9.5.2 Target reset

Upon a target reset, the target aborts all task sets for all command block agents within a unit, and store the unit attention status to logged in initiator(s) other than the initiator which performed target reset.

Each initiator resumes data transfer with target on a connection by connection basis by re-synchronizing a connection(s) (See 9.5.6).

9.5.3 Agent reset

Upon a agent reset, the target aborts all task sets for a fetch agent. The initiator resumes data transfer with target on a connection by connection basis by re-synchronizing a connection(s) (See 9.5.6).

9.5.4 Logical unit reset (optional)

Upon a logical unit reset, the target aborts all task sets for a logical unit, and store the unit attention status to logged in initiators other than the initiator which performed the logical unit reset.

Each initiator resumes data transfer with target on a connection by connection basis by re-synchronizing a connection(s) (See 9.5.6).

9.5.5 Unrecoverable transaction error

Upon an unrecoverable transaction error, the target aborts all task sets for a command block agent and the fetch agent transits to dead state. When target does not receive *ack_complete* or *resp_complete* followed by *ack_pending* for a status block write request subaction, target is not permitted to take a recovery action and the fetch agent transits to dead state. The initiator is responsible to detect this occurrence.

Once an initiator detects that the fetch agent has transit to dead state, the initiator resets the fetch agent and resumes data transfer with target on a connection by connection basis by re-synchronizing a connection(s) (See 9.5.6).

9.5.6 Resynchronization of connections

Data transfer between a client application and a service may have caused device operations to commence even if not all the data had been transferred before the resumption. For this reason, it is essential for each connection to be resynchronized

The target must be able to recognize resumption of an ORB in progress at the time of the resynchronization. The *signature* field in a transport ORB provides a method by which identical ORBs may be recognized if they are resubmitted after conditions described in 9.5.

Data transfer may be safely resumed if initiator and target can identify, for each queue, the ORBs active at the time of the bus reset. For a particular queue, for the initiator, the active ORBs are the outstanding ORBs when the conditions described in 9.5 occurred. An initiator shall perform the following steps for each queue that forms the connection:

a) If there were no uncompleted ORBs in the task set whose *queue* field identifies one of the queue(s) that form the connection, no action is necessary the initiator may resume data transfer for the connection.
b) Otherwise, for all uncompleted ORBs for each of the connection's queues, the initiator shall signal equivalent ORBs to the target fetch agent. Certain parts of the ORB shall remain unchanged: the *direction* bit and the *queue* and *signature* fields shall have the same values both before and after the resumption. If the *direction* bit is zero, the *special* and *end_of_message* bits shall have the same values both before and after the resumption. The *data_descriptor*, and *data_size* fields and the *page_table_present* may have different values, but they shall describe a buffer of the same size and whose contents are identical to the buffer described by the corresponding uncompleted ORB before the resumption. The *spd* and *max_payload* bits may differ as a result of a different topology between the initiator and target after the bus reset.

NOTE - The initiator shall keep the contents of a buffer including the contents modified by the target.

NOTE - A straightforward implementation for the initiator transport is to signal equivalent ORBs (subject to the requirements in the preceding paragraph) in exactly the same order (for each queue) that they were signaled prior to the resumption. This strategy is known to work but other implementations are possible.

The target is responsible to identify whether the signaled ORB(s) is equivalent to the ORB(s) which is in progress before the resumption or not, and to avoid to process the same part of the initiator’s request, which has already processed before the resumption, again. The target uses the context information (a history log) described in 9.4 to perform this role.

a) If the history log(s) that maintains the same queue value as the value specified by the ORB does not exist, no action is necessary and the target may resume data transfer for the queue.

b) Otherwise, if a status block write request which indicates request completion and is not responded by the initiator does not exist for an ORB of which *queue* value is equal to the signaled one, the target shall resume data transfer from the point maintained in the history log.

c) Otherwise, if the history log(s), which maintains the same *queue* value as the value specified by the ORB, exists and a status block write request, which indicates request completion and is not responded by the initiator, exists for an ORB of which *queue* value is equal to the signaled one, target shall compare the *signature* value specified by the ORB with the value maintained in the oldest history log among the history log(s) which maintains the same *queue* value.

d) If the *signature* value specified by the ORB is equal to the value maintained in the oldest history log, the target shall resume data transfer from the point maintained in the history log.

e) Otherwise, if the *signature* value specified by the ORB is not equal to the value maintained in the oldest history log, the target shall discard the oldest history log and its succeeding history log(s) up to the history log which maintains that a request completion status block has stored.

f) If no more history log that maintains the same *queue* value exist, no more action is necessary and the target may resume data transfer for the queue.

g) Otherwise, if such history log(s) is exists, the target shall resume data transfer from the point maintained in the history log.