APPENDIX A: STATE TRANSITIONS

The tables in this appendix describe all state transitions associated with the Transport Interface. First, however, the states and events will be described.

Transport Interface States

Figure A-1 defines the states used to describe the Transport Interface state transitions.

State	Description	Service Type
T_UNINIT	uninitialized – initial and final state of interface	T_COTS, T_COTS_ORD, T_CLTS
T_UNBND	initialized but not bound	T_COTS, T_COTS_ORD, T_CLTS
T_IDLE	no connection established	T_COTS, T_COTS_ORD, T_CLTS
T_OUTCON	outgoing connection pending for client	T_COTS, T_COTS_ORD
T_INCON	incoming connection pending for server	T_COTS, T_COTS_ORD
T_DATAXFER	data transfer	T_COTS, T_COTS_ORD
T_OUTREL	outgoing orderly release (waiting for orderly release indication)	T_COTS_ORD
T_INREL	incoming orderly release (waiting to send orderly release request)	T_COTS_ORD

Figure A-1. Transport Interface States

Outgoing Events

The outgoing events described in Figure A-2 correspond to the return of the specified transport routines, where these routines send a request or response to the transport provider.

In the figure, some events (such as *acceptN*) are distinguished by the context in which they occur. The context is based on the values of the following variables:

ocnt count of outstanding connect indications

fd file descriptor of the current transport endpoint

resfd file descriptor of the transport endpoint where a connection will be accepted

Event	Description	Service Type
opened	successful return of **t_open**	T_COTS, T_COTS_ORD, T_CLTS
bind	successful return of **t_bind**	T_COTS, T_COTS_ORD, T_CLTS
optmgmt	successful return of **t_optmgmt**	T_COTS, T_COTS_ORD, T_CLTS
unbind	successful return of **t_unbind**	T_COTS, T_COTS_ORD, T_CLTS
closed	successful return of **t_close**	T_COTS, T_COTS_ORD, T_CLTS
connect1	successful return of **t_connect** in synchronous mode	T_COTS, T_COTS_ORD
connect2	TNODATA error on **t_connect** in asynchronous mode, or TLOOK error due to a disconnect indication arriving on the transport endpoint	T_COTS, T_COTS_ORD
accept1	successful return of **t_accept** with *ocnt* == 1, *fd* == *resfd*	T_COTS, T_COTS_ORD
accept2	successful return of **t_accept** with *ocnt* == 1, *fd* != *resfd*	T_COTS, T_COTS_ORD
accept3	successful return of **t_accept** with *ocnt* > 1	T_COTS, T_COTS_ORD
snd	successful return of **t_snd**	T_COTS, T_COTS_ORD
snddis1	successful return of **t_snddis** with *ocnt* <= 1	T_COTS, T_COTS_ORD
snddis2	successful return of **t_snddis** with *ocnt* > 1	T_COTS, T_COTS_ORD
sndrel	successful return of **t_sndrel**	T_COTS_ORD
sndudata	successful return of **t_sndudata**	T_CLTS

Figure A-2. Transport Interface Outgoing Events

Incoming Events

The incoming events correspond to the successful return of the specified routines, where these routines retrieve data or event information from the transport provider. The only incoming event not associated directly with the return of a routine is *pass_conn*, which occurs when a user transfers a connection to another transport endpoint. This event occurs on the endpoint that is being passed the connection, despite the fact that no Transport Interface routine is issued on that endpoint. *pass_conn* is included in the state tables to describe the behavior when a user accepts a connection on another transport endpoint.

In Figure A-3, the *rcvdis* events are distinguished by the context in which they occur. The context is based on the value of *ocnt*, which is the count of outstanding connect indications on the transport endpoint.

Incoming Event	Description	Service Type
listen	successful return of **t_listen**	T_COTS, T_COTS_ORD
rcvconnect	successful return of **t_rcvconnect**	T_COTS, T_COTS_ORD
rcv	successful return of **t_rcv**	T_COTS, T_COTS_ORD
rcvdis1	successful return of **t_rcvdis** with **ocnt** <= 0	T_COTS, T_COTS_ORD
rcvdis2	successful return of **t_rcvdis** with *ocnt* == 1	T_COTS, T_COTS_ORD
rcvdis3	successful return of **t_rcvdis** with *ocnt* > 1	T_COTS, T_COTS_ORD
rcvrel	successful return of **t_rcvrel**	T_COTS_ORD
rcvudata	successful return of **t_rcvudata**	T_CLTS
rcvuderr	successful return of **t_rcvuderr**	T_CLTS
pass_conn	receive a passed connection	T_COTS, T_COTS_ORD

Figure A-3. Transport Interface Incoming Events

Transport User Actions

In the state tables that follow, some state transitions are accompanied by a list of actions the transport user must take. These actions are represented by the notation [n], where n is the number of the specific action as described below.

[1] Set the count of outstanding connect indications to zero.

[2] Increment the count of outstanding connect indications.

[3] Decrement the count of outstanding connect indications.

[4] Pass a connection to another transport endpoint as indicated in *t_accept*.

State Tables

The following tables describe the Transport Interface state transitions. Given a current state and an event, the transition to the next state is shown, as well as any actions that must be taken by the transport user (indicated by [n]). The state is that of the transport provider as seen by the transport user.

The contents of each box represent the next state, given the current state (column) and the current incoming or outgoing event (row). An empty box represents a state/event combination that is invalid. Along with the next state, each box may include an action list (as specified in the previous section). The transport user must take the specific actions in the order specified in the state table.

The following should be understood when studying the state tables:

- The **t_close** routine is referenced in the state tables (see *closed* event in Figure A-2), but may be called from any state to close a transport endpoint. If **t_close** is called when a transport address is bound to an endpoint, the address will be unbound. Also, if **t_close** is called when the transport connection is still active, the connection will be aborted.
- If a transport user issues a routine out of sequence, the transport provider will recognize this and the routine will fail, setting **t_errno** to TOUTSTATE. The state will not change.
- If any other transport error occurs, the state will not change unless explicitly stated on the manual page for that routine. The exception to this is a TLOOK or TNODATA error on **t_connect**, as described in Figure A-2. The state tables assume correct use of the Transport Interface.
- The support routines **t_getinfo**, **t_getstate**, **t_alloc**, **t_free**, **t_sync**, **t_look**, and **t_error** are excluded from the state tables because they do not affect the state.

A separate table is shown for common local management steps, data transfer in connectionless-mode, and connection-establishment/connection-release/data-transfer in connection-mode.

state / event	T_UNINIT	T_UNBND	T_IDLE
opened	T_UNBND		
bind		T_IDLE [1]	
optmgmt			T_IDLE
unbind			T_UNBND
closed		T_UNINIT	

Figure A-4. Common Local Management State Table

state / event	T_IDLE
sndudata	T_IDLE
rcvudata	T_IDLE
rcvuderr	T_IDLE

Figure A-5. Connectionless-Mode State Table

APPENDIX A: STATE TRANSITIONS

state / event	T_IDLE	T_OUTCON	T_INCON	T_DATAXFER	T_OUTREL	T_INREL
connect1	T_DATAXFER					
connect2	T_OUTCON					
rcvconnect		T_DATAXFER				
listen	T_INCON [2]		T_INCON [2]			
accept1			T_DATAXFER[3]			
accept2			T_IDLE [3][4]			
accept3			T_INCON [3][4]			
snd				T_DATAXFER		T_INREL
rcv				T_DATAXFER	T_OUTREL	
snddis1		T_IDLE	T_IDLE [3]	T_IDLE	T_IDLE	T_IDLE
snddis2			T_INCON [3]			
rcvdis1		T_IDLE		T_IDLE	T_IDLE	T_IDLE
rcvdis2			T_IDLE [3]			
rcvdis3			T_INCON [3]			
sndrel				T_OUTREL		T_IDLE
rcvrel				T_INREL	T_IDLE	
pass_conn	T_DATAXFER					

Figure A-6. Connection-Mode State Table