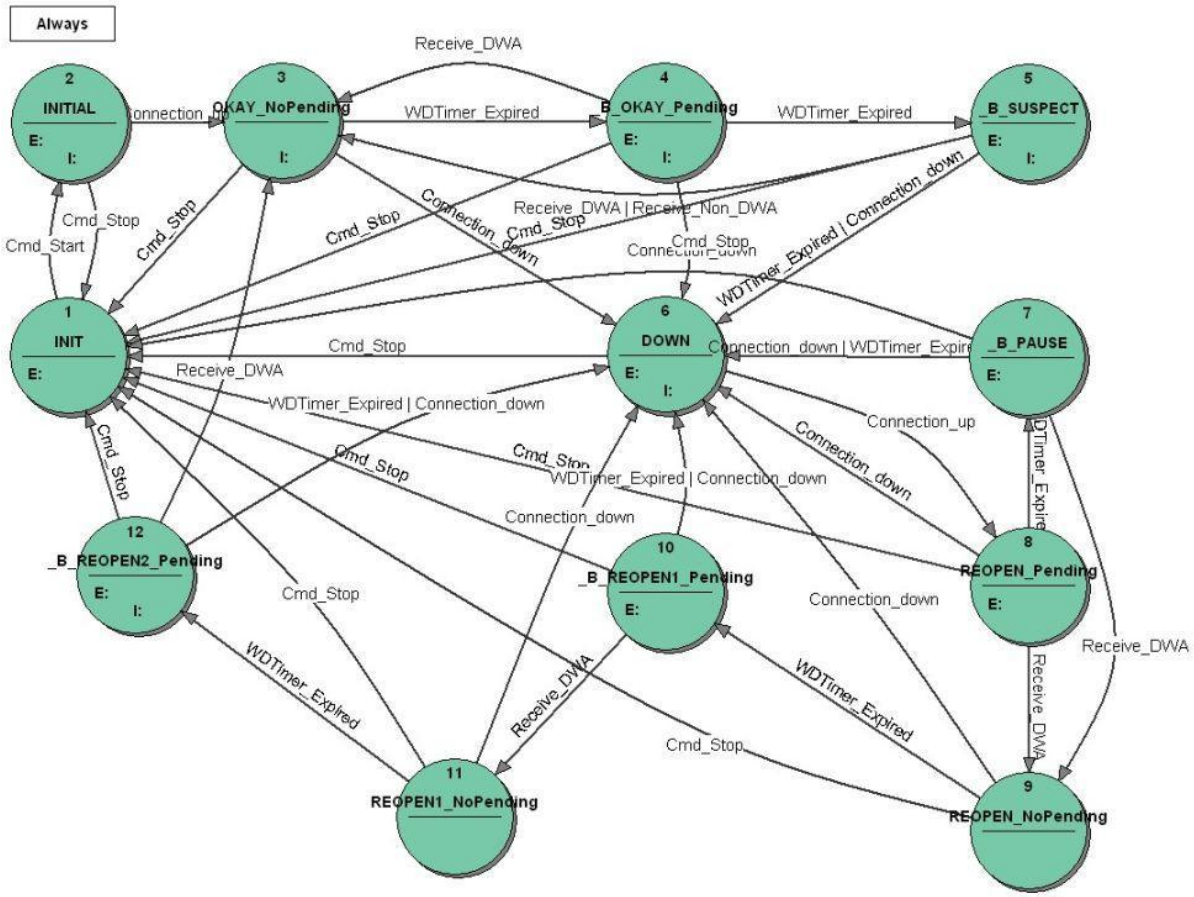
VFSM: Failover

VFSM type: Failover(user vfsm)

The Failover state machine is part of the DIAMETER protocal (RFC 3588) which requires the failover mechanism. The failover is based on a single timer (watchdog) and its algorithm is specified in the RFC 3539 document.The task of the state machine is to detect transport failures and is based on a cyclic exchange of Device-WatchDog-Request (DWR) and Device-Watchdog-Answer (DWA).If the connection is down the state machine continues periodically to reopen the connection.If the connection is succesfully opened a connection validation is required before the connection is returned to service; this is done via exchange of three watchdog messages.



Prefix: FAI

IOid name: MyCmd (type: CMD-IN)
-->Virtual Input: Cmd_Start (1)
-->Virtual Input: Cmd_Stop (2)

IOid name: Event (type: CMD-IN)
-->Virtual Input: Connection_down (52)
-->Virtual Input: Connection_up (51)
-->Virtual Input: Receive_DWA (24)
-->Virtual Input: Receive_Non_DWA (20)
-->Virtual Output: Event_Clear (0)

IOid name: ConnectUnitCmd (type: CMD-OUT)
-->Virtual Output: AttemptOpen (42)
-->Virtual Output: CloseConnection (43)
-->Virtual Output: Failback (44)
-->Virtual Output: Failover (45)

-->Virtual Output: SendWatchdog (41)
-->Virtual Output: UnitCmd_Clear (0)

IOid name: WDTimer (type: TI)
-->Virtual Input: WDTimer_Expired (OVER)
-->Virtual Output: WDTimer_Stop (Stop)

IOid name: Ack (type: XDA)
-->Virtual Input: UnitCmd_Ack (1)
-->Virtual Output: UnitCmdAck_Clear (0)

IOid name: TWINIT (type: PAR)

IOid name: WDTime (type: PAR)

IOid name: SetWDTimer (type: OFUN)
-->Virtual Output: SetWatchdog (46)

Always: input actions valid in each state:
All received events are cleared immediately.CmdAck acknowledges the unit output operation: it clears the Cmd and the Ack signals.The action SetWatchdog triggers setting a new value of the watchdog time and starting the watchdog.

	Connection_up **or** Connection_down **or** Receive_DWA **or** Receive_Non_DWA	Event_Clear
	UnitCmd_Ack	UnitCmd_Clear UnitCmdAck_Clear

State transition table, state ' INIT ':
The initial state, when the Failover state machine is first created, or when it has been deliberately stopped by a command.

INIT	***EntryAction:***	WDTimer_Stop
	ExitAction:	
INITIAL	Cmd_Start	

State transition table, state ' INITIAL ':
The original initial state according to the RFC 3539.The watchdog in this state triggers attempts to establish a connection.On entering the state the watchdog time is calculated and set in the IO-Handler, thereafter the timer is restarted.Everytime the watchdog elapses it triggers a trial to open a connection (AttemptOpen) and the watchdog is restarted.On receiving the signal that connection is established the watchdog is set and restarted again and the state machine goes to the state OKAY_NoPending.AttemptOpen has to trigger the event Connection_up.Remark. Normally, the state machine will pass this state going immendiately to the state OKAY_NoPending: it comes here on receiving a command Start from the Peer which sends it entering the state I_Open or R_Open. In that moment the connection is established. Thus, the Connection_up event will come immediately as an answer to the AttemptOpen.

INITIAL	***EntryAction:***	AttemptOpen SetWatchdog
	ExitAction:	
	WDTimer_Expired	AttemptOpen SetWatchdog
OKAY_NoPending	Connection_up	
INIT	Cmd_Stop	

State transition table, state ' OKAY_NoPending ':
The connection is up.The watchdog supervises whether the connection is alive (i.e. receives messages).Any message received resets the Watchdog, so the time spent in this state can be very long.If the watchdog expires with no pending DWA the state machine goes to the state OKAY_Pending, where a new DWR will be issued. This only occurs when there is very low traffic on the open connection: for instance when it is a standby connection.If connection is down or on receiving the command Stop the state machine goes to the state DOWN. The connection down causes the Failover (send all pending messages to the alternate peer).

OKAY_NoPending	***EntryAction:***	
	ExitAction:	
	Connection_down	Failover
	Receive_DWA **or** Receive_Non_DWA	SetWatchdog
DOWN	Connection_down	
INIT	Cmd_Stop	
_B_OKAY_Pending	WDTimer_Expired	

State transition table, state ' _B_OKAY_Pending ':
On entering the state the watchdog is restarted and the DWR sent. This would be the first DWR sent on a newly-established connection.The state machine supervises in that state the receipt of the watchdog answer (DWA): this state means that there is a pending DWR request.On receiving the DWA answer the state machine returns to the state OKAY_NoPending.If the watchdog expires (something is wrong with the connection) the state machine goes to the state SUSPECT.

_B_OKAY_Pending	***EntryAction:***	SendWatchdog SetWatchdog
	ExitAction:	
	Receive_DWA **or** Receive_Non_DWA	SetWatchdog
OKAY_NoPending	Receive_DWA	
DOWN	Connection_down	
INIT	Cmd_Stop	
_B_SUSPECT	WDTimer_Expired	

State transition table, state ' _B_SUSPECT ':
Failover detect connection problem.The failover process should be started by the application state machine(s).On entering the state the watchdog is restarted and if it expires the state machine goes to the state DOWN.On receiving any message the connection is considered to be ok and the state machine returns to the state OKAY and the failback process should be initiated by the application state machine(s). Failover means to send all pending messages to the alternate peer.

_B_SUSPECT	***EntryAction:***	Failover SetWatchdog
	ExitAction:	
	Receive_DWA **or** Receive_Non_DWA	SetWatchdog Failback
OKAY_NoPending	Receive_DWA **or** Receive_Non_DWA	
DOWN	WDTimer_Expired **or** Connection_down	

INIT	Cmd_Stop	

State transition table, state ' DOWN ':

On entering the state the connection is closed and the watchdog restarted.The watchdog in this state triggers attempts to establish a connection.Any time the watchdog elapses it is attempts to reconnect and the watchdog is restarted, unless a command Stop had been received, which would send us directly to INIT.If the transport layer signals a working connection the state machine goes to the state REOPEN_Pending.

DOWN	***EntryAction:***	CloseConnection SetWatchdog
	ExitAction:	
	WDTimer_Expired	AttemptOpen SetWatchdog
REOPEN_Pending	Connection_up	
INIT	Cmd_Stop	

State transition table, state ' _B_PAUSE ':

The REOPEN process seemed to be interrupted.To give it another chance we delay the transition to the state DOWN.If the DWA answer comes before the timer expires we continue REOPENing process.

_B_PAUSE	***EntryAction:***	SetWatchdog
	ExitAction:	
DOWN	Connection_down **or** WDTimer_Expired	
REOPEN_NoPending	Receive_DWA	
INIT	Cmd_Stop	

State transition table, state ' REOPEN_Pending ':

An attempt to reopen a closed connection seems to be successful.To be sure that it truly works we start to count DWA: on receiving DWA the state machine goes to the state REOPEN1_Pending via REOPEN_NoPending.If the watchdog expires the state machine returns to the state DOWN but not immediately: the state machine delays it by going first to the state PAUSE.If the connection is down or must be stopped the state machine returns immediately to the state DOWN.

REOPEN_Pending	***EntryAction:***	SendWatchdog SetWatchdog
	ExitAction:	
DOWN	Connection_down	
REOPEN_NoPending	Receive_DWA	
INIT	Cmd_Stop	
_B_PAUSE	WDTimer_Expired	

State transition table, state ' REOPEN_NoPending ':

The first DWA has been received.The state delays sending the next watchdog which should be sent not too frequently, i.e. the watchdog timer dictates when to send another watchdog and not the DWA answer.

REOPEN_NoPending	***EntryAction:***	
	ExitAction:	
DOWN	Connection_down	
INIT	Cmd_Stop	
_B_REOPEN1_Pending	WDTimer_Expired	

State transition table, state ' _B_REOPEN1_Pending ':
One DWA has been already received and we can sent the next watchdog request.

_B_REOPEN1_Pending	***EntryAction:***	SendWatchdog SetWatchdog
	ExitAction:	
DOWN	WDTimer_Expired **or** Connection_down	
REOPEN1_NoPending	Receive_DWA	
INIT	Cmd_Stop	

State transition table, state ' REOPEN1_NoPending ':
The second DWA has been received.The state delays sending the next watchdog which should be sent not too frequently, i.e. the watchdog timer dictates when to send another watchdog and not the DWA answer.

REOPEN1_NoPending	***EntryAction:***	
	ExitAction:	
DOWN	Connection_down	
INIT	Cmd_Stop	
_B_REOPEN2_Pending	WDTimer_Expired	

State transition table, state ' _B_REOPEN2_Pending ':
Two DWAs have been received and we can sent the next watchdog request.If the third DWA comes the connection is definitely ok, the failback process should be initiated by the application state machine(s) and the state machine goes to the state OKAY_NoPending (corresponding to state OKAY of RFC 3539).

_B_REOPEN2_Pending	***EntryAction:***	SetWatchdog SendWatchdog
	ExitAction:	
	Receive_DWA	Failback
DOWN	WDTimer_Expired **or** Connection_down	
OKAY_NoPending	Receive_DWA	
INIT	Cmd_Stop	

Total number of states: 13