

PAA State Machine

CONDITION	ACTION
Rx: PDI_STATEFUL_DISC_PBACK(1) (OFFLINE → WAIT_EAP_MSG_IN_DISC)	
(Rx:PDI PAC_FOUND) && USE_COOKIE==Unset && PIGGYBACK==Set	EAP_Restart();
Rx: PDI_STATEFUL_DISC(1) (OFFLINE → STATEFUL_DISC)	
(Rx:PDI PAC_FOUND) && USE_COOKIE==Unset && PIGGYBACK==Unset	if (SEPARATE==Set) PSR.S_flag=1; if (CARRY_NAP_INFO==Set) PSR.insert_avp ("NAP-Information"); if (CARRY_ISP_INFO==Set) PSR.insert_avp ("ISP-Information"); if (CARRY_PPAC==Set) PSR.insert_avp ("Post-PANA-Address- Configuration"); Tx:PSR(); RtxTimerStart();
Rx: PDI_STATELESS_DISC(1) (OFFLINE → OFFLINE)	
(Rx:PDI PAC_FOUND) && USE_COOKIE==Set	if (SEPARATE==Set) PSR.S_flag=1; PSR.insert_avp ("Cookie"); if (CARRY_NAP_INFO==Set) PSR.insert_avp ("NAP-Information"); if (CARRY_ISP_INFO==Set) PSR.insert_avp ("ISP-Information"); if (CARRY_PPAC==Set) PSR.insert_avp ("Post-PANA-Address- Configuration"); Tx:PSR();
Rx: PSA_NO_MOBILITY(1) (OFFLINE → WAIT_EAP_MSG)	
Rx:PSA && USE_COOKIE==Set && (!PSA.exist_avp ("Session-Id") !PSA.exit_avp ("Nonce") MOBILITY==Unset (MOBILITY==Set && !retrieve_pana_sa (PSA.SESSION_ID)))	if (SEPARATE==Set && PSA.S_flag==0) SEPARATE=Unset; EAP_Restart();
Rx: PSA_MOBILITY(1) (OFFLINE → WAIT_SUCC_PBA)	

<pre>Rx:PSA && USE_COOKIE==Set && PSA.exist_avp ("Session-Id") && PSA.exist_avp ("Nonce") && MOBILITY==Set && retrieve_pana_sa (PSA.SESSION_ID)</pre>	<pre>PBR.insert_avp("MAC"); PBR.insert_avp("Nonce"); PBR.insert_avp("Key-Id"); if (CARRY_EP_DEVICE_ID ==Set) PBR.insert_avp ("EP-Device-Id"); if (PROTECTION_CAP==Set) PBR.insert_avp ("Protection-Cap."); PBR.insert_avp("MAC"); Tx:PBR(); RtxTimerStart();</pre>
EAP_REQUEST_DISC(1) (WAIT EAP MSG IN DISC → STATEFUL DISC)	
<pre>EAP_REQUEST</pre>	<pre>PSR.insert_avp ("EAP-Payload"); if (CARRY_NAP_INFO==Set) PSR.insert_avp ("NAP-Information"); if (CARRY_ISP_INFO==Set) PSR.insert_avp ("ISP-Information"); if (CARRY_PPAC==Set) PSR.insert_avp ("Post-PANA-Address- Configuration"); Tx:PSR(); RtxTimerStart();</pre>
Rx:PSA(1) (STATEFUL DISC → WAIT PAN)	
<pre>Rx:PSA</pre>	<pre>if (SEPARATE==Set && PSA.S_flag==0) SEPARATE=Unset; if (SEPARATE==Set) { PAR.S_flag=1; NAP_AUTH=Set Unset; if (NAP_AUTH==Set) PAR.N_flag=1; } Tx:PAR();</pre>
EAP_TIMEOUT_DISC(1) (STATEFUL DISC → CLOSED)	
<pre>EAP_TIMEOUT</pre>	<pre>Tx:PER(); Disconnect();</pre>
EAP_REQUEST_EAPMSG(1) (WAIT EAP MSG → WAIT PAN)	
<pre>EAP_REQUEST</pre>	<pre>if (key_available()) PAR.insert_avp("MAC"); if (SEPARATE==Set) { PAR.S_flag=1; if (NAP_AUTH==Set) PAR.N_flag=1; } Tx:PAR();</pre>
EAP_TIMEOUT_EAPMSG(1) (WAIT EAP MSG → CLOSED)	
<pre>EAP_TIMEOUT</pre>	<pre>Tx:PER(); Disconnect();</pre>

EAP_FAILURE_TIMEOUT_NOAUTH(8) (WAIT EAP MSG → WAIT FAIL PBA)	
EAP_FAILURE && 1ST_EAP==Unset && SEPARATE==Unset	1ST_EAP=Failure PBR.insert_avp ("EAP-Payload"); if (key_available()) PBR.insert_avp("MAC"); Tx:PBR(); RtxTimerStart();
EAP_SUCCESS && 1ST_EAP==Unset && SEPARATE==Unset && !Authorize()	1ST_EAP=Success PBR.insert_avp ("EAP-Payload"); if (new_key_available()) PBR.insert_avp ("Key-Id"); if (key_available()) PBR.insert_avp("MAC"); Tx:PBR(); RtxTimerStart();
EAP_TIMEOUT && 1ST_EAP==Unset && SEPARATE==Unset	1ST_EAP=Failure if (key_available()) PBR.insert_avp("MAC"); Tx:PBR(); RtxTimerStart();
EAP_FAILURE && 1ST_EAP==Failure	PBR.insert_avp ("EAP-Payload"); if (key_available()) PBR.insert_avp("MAC"); if (SEPARATE) PBR.S_flag=1; if (NAP_AUTH) PBR.N_flag=1; Tx:PBR(); RtxTimerStart();
EAP_SUCCESS && 1ST_EAP==Success && !Authorize()	PBR.insert_avp ("EAP-Payload"); if (new_key_available()) PBR.insert_avp ("Key-Id"); if (key_available()) PBR.insert_avp("MAC"); if (SEPARATE) PBR.S_flag=1; if (NAP_AUTH) PBR.N_flag=1; Tx:PBR(); RtxTimerStart();

<pre>EAP_FAILURE && 1ST_EAP==Success && !Authorize()</pre>	<pre>PBR.insert_avp ("EAP-Payload"); if (key_available()) PBR.insert_avp("MAC"); if (SEPARATE) PBR.S_flag=1; if (NAP_AUTH) PBR.N_flag=1; Tx:PBR(); RtxTimerStart();</pre>
<pre>EAP_TIMEOUT && 1ST_EAP==Failure</pre>	<pre>if (key_available()) PBR.insert_avp("MAC"); if (SEPARATE) PBR.S_flag=1; if (NAP_AUTH) PBR.N_flag=1; Tx:PBR(); RtxTimerStart();</pre>
<pre>EAP_TIMEOUT && 1ST_EAP==Success && !Authorize()</pre>	<pre>if (key_available()) PBR.insert_avp("MAC"); if (SEPARATE) PBR.S_flag=1; if (NAP_AUTH) PBR.N_flag=1; Tx:PBR(); RtxTimerStart();</pre>
EAP_SUCC_AUTH (4) (WAIT_EAP_MSG → WAIT_SUCC_PBA)	
<pre>EAP_SUCCESS && 1ST_EAP==Unset && SEPARATE==Unset && Authorize()</pre>	<pre>1ST_EAP=Success PBR.insert_avp ("EAP-Payload"); if (CARRY_EP_DEVICE_ID ==Set) PBR.insert_avp ("EP-Device-Id"); if (CARRY_LIFETIME==Set) PBR.insert_avp ("Session-Lifetime"); if (PROTECTION_CAP==Set) PBR.insert_avp ("Protection-Cap."); if (new_key_available()) PBR.insert_avp ("Key-Id"); if (key_available()) PBR.insert_avp("MAC"); Tx:PBR(); RtxTimerStart();</pre>
<pre>EAP_SUCCESS && 1ST_EAP==Success && Authorize()</pre>	<pre>PBR.insert_avp ("EAP-Payload"); if (CARRY_EP_DEVICE_ID==Set) PBR.insert_avp ("EP-Device-Id"); if (PROTECTION_CAP==Set) PBR.insert_avp ("Protection-Cap.");</pre>

	<pre> if (new_key_available()) PBR.insert_avp("Key-Id"); if (key_available()) PBR.insert_avp("MAC"); if (SEPARATE) PBR.S_flag=1; if (NAP_AUTH) PBR.N_flag=1; Tx:PBR(); RtxTimerStart(); </pre>
<pre> EAP_FAILURE && 1ST_EAP==Success && Authorize() </pre>	<pre> PBR.insert_avp ("EAP-Payload"); if (key_available()) PBR.insert_avp("MAC"); if (SEPARATE) PBR.S_flag=1; if (NAP_AUTH) PBR.N_flag=1; Tx:PBR(); RtxTimerStart(); </pre>
<pre> EAP_TIMEOUT && 1ST_EAP==Success && Authorize() </pre>	<pre> if (key_available()) PBR.insert_avp("MAC"); if (SEPARATE) PBR.S_flag=1; if (NAP_AUTH) PBR.N_flag=1; Tx:PBR(); RtxTimerStart(); </pre>
EAP_SUCC_TIMEOUT_FAIL_SEPARATE(3) (WAIT_EAP_MSG → WAIT_PFEA)	
<pre> EAP_FAILURE && 1ST_EAP==Unset && SEPARATE==Set && ABORT_ON_1ST_EAP_FAILURE ==Unset </pre>	<pre> 1ST_EAP=Failure PBR.insert_avp ("EAP-Payload"); if (key_available()) PFER.insert_avp("MAC"); PFER.S_flag=1; if (NAP_AUTH) PFER.N_flag=1; Tx:PFER(); RtxTimerStart(); </pre>
<pre> EAP_SUCCESS && 1ST_EAP==Unset && SEPARATE==Set </pre>	<pre> 1ST_EAP=Success PFER.insert_avp ("EAP-Payload"); if (new_key_available()) PFER.insert_avp ("Key-Id"); if (key_available()) PFER.insert_avp("MAC"); PFER.S_flag=1; if (NAP_AUTH) PFER.N_flag=1; Tx:PFER(); RtxTimerStart(); </pre>
<pre> EAP_TIMEOUT && 1ST_EAP==Unset && SEPARATE==Set && </pre>	<pre> 1ST_EAP=Failure if (key_available()) PFER.insert_avp("MAC"); </pre>

ABORT_ON_1ST_EAP_FAILURE ==Unset	PFER.S_flag=1; if (NAP_AUTH) PFER.N_flag=1; Tx:PFER(); RtxTimerStart();
EAP_FAILURE_TIMEOUT_ABORT_ON(2) (WAIT EAP MSG → WAIT FAIL PFEA)	
EAP_FAILURE && 1ST_EAP==Unset && SEPARATE==Set && ABORT_ON_1ST_EAP_FAILURE ==Set	1ST_EAP=Failure PFER.insert_avp ("EAP-Payload"); if (key_available()) PFER.insert_avp("MAC"); PFER.S_flag=0; Tx:PFER(); RtxTimerStart();
EAP_TIMEOUT && 1ST_EAP==Unset && SEPARATE==Set && ABORT_ON_1ST_EAP_FAILURE ==Set	1ST_EAP=Failure if (key_available()) PFER.insert_avp("MAC"); SEPARATE=Unset; PFER.S_flag=0; Tx:PFER(); RtxTimerStart();
Rx:PFEA_SFLAG1(1) (WAIT PFEA → WAIT EAP MSG)	
Rx:PFEA && PFEA.S_flag==1	if (key_available()) PAR.insert_avp("MAC"); if (NAP_AUTH==Set) { NAP_AUTH=Unset; PAR.N_flag=0; } else { NAP_AUTH=Set; PAR.N_flag=1; }; EAP_Restart();
Rx:PFEA_SFLAG0(1) (WAIT PFEA → CLOSED)	
Rx:PFEA && PFEA.S_flag==0	RtxTimerStop(); Disconnect();
Rx:PFEA(1) (WAIT FAIL PFEA → CLOSED)	
Rx:PFEA	RtxTimerStop(); Disconnect();
Rx:PBA_SUCC(1) (WAIT SUCC PBA → OPEN)	
Rx:PBA	SessionTimerStart(); Authorize();
Rx:PBA_FAIL(1) (WAIT FAIL PBA → CLOSED)	

Rx:PBA	RtxTimerStop(); Disconnect();
Rx:PBA_FAIL(1) (OPEN → WAIT EAP MSG)	
EAP_REAUTH (Rx:PDI && PDI.exist_avp ("Session-Id"))	if (key_available()) PAR.insert_avp("MAC"); EAP_Restart(); 1ST_EAP=Unset; NAP_AUTH=Set Unset;
Rx:PBA_FAIL(1) (OPEN → WAIT PRAA)	
FAST_REAUTH	Tx:PRAR(); RtxTimerStart();
Rx:PBA_FAIL(1) (OPEN → SESS TERM)	
TERMINATE	if (key_available()) PTR.insert_avp("MAC"); Tx:PTR(); RtxTimerStart();
Rx:PBA_FAIL(1) (OPEN → CLOSED)	
Rx:PTR	if (key_available()) PTA.insert_avp("MAC"); Tx:PTA(); Disconnect();
Rx:PAUR_AUTH(1) (OPEN → OPEN)	
Rx:PAUR && Authorize()	Tx:PAUA();
Rx:PRAA(1) (WAIT PRAA → OPEN)	
Rx:PRAA	RtxTimerStop();
Rx:PRAA(1) (WAIT PAN → WAIT EAP MSG)	
Rx:PAN	TxEAP();
EAP_TIMEOUT_PAN(1) (WAIT PAN → CLOSED)	
EAP_TIMEOUT	Tx:PER(); RtxTimerStop(); Disconnect();
EAP_TIMEOUT_PAN(1) (SESS TERM → CLOSED)	
Rx:PTA	RtxTimerStop(); Disconnect();