# wpa_supplicant Reference Manual

0.5.x

Generated by Doxygen 1.4.2

Sun Dec 31 13:48:50 2006

# Contents


**1 Developers' documentation for wpa_supplicant** **1**

**2 wpa_supplicant Data Structure Index** **3**

2.1 wpa_supplicant Data Structures . . . . . 3

**3 wpa_supplicant File Index** **5**

3.1 wpa_supplicant File List . . . . . 5

**4 wpa_supplicant Page Index** **11**

4.1 wpa_supplicant Related Pages . . . . . 11

**5 wpa_supplicant Data Structure Documentation** **13**

5.1 eap_config Struct Reference . . . . . 13

5.2 eap_method Struct Reference . . . . . 15

5.3 eap_method_ret Struct Reference . . . . . 21

5.4 eap_sm Struct Reference . . . . . 22

5.5 eapol_callbacks Struct Reference . . . . . 24

5.6 eapol_config Struct Reference . . . . . 28

5.7 eapol_ctx Struct Reference . . . . . 30

5.8 eapol_sm Struct Reference . . . . . 34

5.9 rsn_pmksa_cache_entry Struct Reference . . . . . 37

5.10 tls_connection_params Struct Reference . . . . . 38

5.11 wpa_config Struct Reference . . . . . 40

5.12 wpa_config_blob Struct Reference . . . . . 45

5.13 wpa_ctrl Struct Reference . . . . . 46

5.14 wpa_ctrl_dst Struct Reference . . . . . 47

5.15 wpa_driver_associate_params Struct Reference . . . . . 48

5.16 wpa_driver_capa Struct Reference . . . . . 50

5.17 wpa_driver_ops Struct Reference . . . . . 51

5.18 wpa_event_data Union Reference . . . . . 64

5.19 wpa_event_data::assoc_info Struct Reference . . . 66
5.20 wpa_event_data::interface_status Struct Reference . . . 68
5.21 wpa_event_data::michael_mic_failure Struct Reference . . . 69
5.22 wpa_event_data::pmkid_candidate Struct Reference . . . 70
5.23 wpa_event_data::stkstart Struct Reference . . . 71
5.24 wpa_global Struct Reference . . . 72
5.25 wpa_interface Struct Reference . . . 73
5.26 wpa_params Struct Reference . . . 75
5.27 wpa_ptk Struct Reference . . . 77
5.28 wpa_scan_result Struct Reference . . . 78
5.29 wpa_sm Struct Reference . . . 79
5.30 wpa_ssid Struct Reference . . . 81
5.31 wpa_supplicant Struct Reference . . . 97

**6 wpa_supplicant File Documentation** **99**

6.1 aes.c File Reference . . . 99
6.2 aes.h File Reference . . . 105
6.3 aes_wrap.c File Reference . . . 106
6.4 aes_wrap.h File Reference . . . 113
6.5 asn1.c File Reference . . . 120
6.6 asn1.h File Reference . . . 122
6.7 asn1_test.c File Reference . . . 124
6.8 base64.c File Reference . . . 126
6.9 base64.h File Reference . . . 128
6.10 bignum.c File Reference . . . 130
6.11 bignum.h File Reference . . . 137
6.12 build_config.h File Reference . . . 143
6.13 common.c File Reference . . . 144
6.14 common.h File Reference . . . 151
6.15 config.c File Reference . . . 162
6.16 config.h File Reference . . . 171
6.17 config_file.c File Reference . . . 181
6.18 config_none.c File Reference . . . 184
6.19 config_ssid.h File Reference . . . 187
6.20 config_types.h File Reference . . . 190
6.21 config_winreg.c File Reference . . . 191
6.22 crypto.c File Reference . . . 194

6.23 crypto.h File Reference . . . 197
6.24 crypto_cryptoapi.c File Reference . . . 210
6.25 crypto_gnutls.c File Reference . . . 213
6.26 crypto_internal.c File Reference . . . 215
6.27 crypto_libtomcrypt.c File Reference . . . 216
6.28 crypto_none.c File Reference . . . 219
6.29 ctrl_iface.c File Reference . . . 221
6.30 ctrl_iface.h File Reference . . . 225
6.31 ctrl_iface_dbus.c File Reference . . . 231
6.32 ctrl_iface_dbus.h File Reference . . . 239
6.33 ctrl_iface_dbus_handlers.c File Reference . . . 240
6.34 ctrl_iface_dbus_handlers.h File Reference . . . 252
6.35 ctrl_iface_named_pipe.c File Reference . . . 253
6.36 ctrl_iface_udp.c File Reference . . . 257
6.37 ctrl_iface_unix.c File Reference . . . 261
6.38 dbus_dict_helpers.c File Reference . . . 265
6.39 dbus_dict_helpers.h File Reference . . . 274
6.40 defs.h File Reference . . . 283
6.41 des.c File Reference . . . 286
6.42 driver.h File Reference . . . 288
6.43 driver_atmel.c File Reference . . . 290
6.44 driver_broadcom.c File Reference . . . 292
6.45 driver_bsd.c File Reference . . . 295
6.46 driver_hostap.c File Reference . . . 298
6.47 driver_hostap.h File Reference . . . 300
6.48 driver_ipw.c File Reference . . . 302
6.49 driver_madwifi.c File Reference . . . 304
6.50 driver_ndis.c File Reference . . . 306
6.51 driver_ndis.h File Reference . . . 310
6.52 driver_ndis_.c File Reference . . . 311
6.53 driver_ndiswrapper.c File Reference . . . 313
6.54 driver_prism54.c File Reference . . . 315
6.55 driver_test.c File Reference . . . 317
6.56 driver_wext.c File Reference . . . 319
6.57 driver_wext.h File Reference . . . 327
6.58 driver_wired.c File Reference . . . 334

6.59 drivers.c File Reference . . . 336
6.60 eap.c File Reference . . . 338
6.61 eap.h File Reference . . . 354
6.62 eap_aka.c File Reference . . . 366
6.63 eap_defs.h File Reference . . . 367
6.64 eap_fast.c File Reference . . . 369
6.65 eap_gpsk.c File Reference . . . 371
6.66 eap_gpsk_common.c File Reference . . . 373
6.67 eap_gpsk_common.h File Reference . . . 377
6.68 eap_gtc.c File Reference . . . 381
6.69 eap_i.h File Reference . . . 383
6.70 eap_leap.c File Reference . . . 391
6.71 eap_md5.c File Reference . . . 393
6.72 eap_methods.c File Reference . . . 395
6.73 eap_methods.h File Reference . . . 401
6.74 eap_mschapv2.c File Reference . . . 407
6.75 eap_otp.c File Reference . . . 409
6.76 eap_pax.c File Reference . . . 411
6.77 eap_pax_common.c File Reference . . . 412
6.78 eap_pax_common.h File Reference . . . 416
6.79 eap_peap.c File Reference . . . 420
6.80 eap_psk.c File Reference . . . 421
6.81 eap_psk_common.c File Reference . . . 422
6.82 eap_psk_common.h File Reference . . . 424
6.83 eap_sake.c File Reference . . . 425
6.84 eap_sake_common.c File Reference . . . 427
6.85 eap_sake_common.h File Reference . . . 430
6.86 eap_sim.c File Reference . . . 433
6.87 eap_sim_common.c File Reference . . . 435
6.88 eap_sim_common.h File Reference . . . 437
6.89 eap_tls.c File Reference . . . 440
6.90 eap_tls_common.c File Reference . . . 441
6.91 eap_tls_common.h File Reference . . . 448
6.92 eap_tlv.c File Reference . . . 455
6.93 eap_tlv.h File Reference . . . 458
6.94 eap_ttls.c File Reference . . . 461

6.95 eap_ttls.h File Reference . . . . . 463
6.96 eap_vendor_test.c File Reference . . . . . 465
6.97 eapol_sm.c File Reference . . . . . 467
6.98 eapol_sm.h File Reference . . . . . 481
6.99 eapol_test.c File Reference . . . . . 494
6.100eloop.c File Reference . . . . . 497
6.101eloop.h File Reference . . . . . 504
6.102eloop_none.c File Reference . . . . . 516
6.103eloop_win.c File Reference . . . . . 520
6.104events.c File Reference . . . . . 528
6.105includes.h File Reference . . . . . 531
6.106l2_packet.h File Reference . . . . . 535
6.107l2_packet_freebsd.c File Reference . . . . . 541
6.108l2_packet_linux.c File Reference . . . . . 545
6.109l2_packet_ndis.c File Reference . . . . . 549
6.110l2_packet_none.c File Reference . . . . . 554
6.111l2_packet_pcap.c File Reference . . . . . 558
6.112l2_packet_winpcap.c File Reference . . . . . 563
6.113libtommath.c File Reference . . . . . 568
6.114main.c File Reference . . . . . 570
6.115main_none.c File Reference . . . . . 572
6.116main_winmain.c File Reference . . . . . 574
6.117main_winsvc.c File Reference . . . . . 576
6.118md4.c File Reference . . . . . 578
6.119md5.c File Reference . . . . . 580
6.120md5.h File Reference . . . . . 583
6.121mlme.c File Reference . . . . . 585
6.122mlme.h File Reference . . . . . 589
6.123ms_funcs.c File Reference . . . . . 590
6.124ms_funcs.h File Reference . . . . . 598
6.125ndis_events.c File Reference . . . . . 605
6.126os.h File Reference . . . . . 607
6.127os_internal.c File Reference . . . . . 614
6.128os_none.c File Reference . . . . . 620
6.129os_unix.c File Reference . . . . . 626
6.130os_win32.c File Reference . . . . . 632

6.131pcsc_funcs.c File Reference . . . . . 638
6.132pcsc_funcs.h File Reference . . . . . 644
6.133pmksa_cache.c File Reference . . . . . 646
6.134pmksa_cache.h File Reference . . . . . 652
6.135preauth.c File Reference . . . . . 657
6.136preauth.h File Reference . . . . . 663
6.137preauth_test.c File Reference . . . . . 669
6.138priv_netlink.h File Reference . . . . . 672
6.139radius.c File Reference . . . . . 674
6.140radius.h File Reference . . . . . 677
6.141radius_client.c File Reference . . . . . 681
6.142radius_client.h File Reference . . . . . 683
6.143rc4.c File Reference . . . . . 685
6.144rc4.h File Reference . . . . . 688
6.145rsa.c File Reference . . . . . 690
6.146rsa.h File Reference . . . . . 694
6.147sha1.c File Reference . . . . . 698
6.148sha1.h File Reference . . . . . 703
6.149sha256.c File Reference . . . . . 708
6.150sha256.h File Reference . . . . . 711
6.151state_machine.h File Reference . . . . . 714
6.152tests/test_aes.c File Reference . . . . . 718
6.153tests/test_eap_sim_common.c File Reference . . . . . 720
6.154tests/test_md4.c File Reference . . . . . 721
6.155tests/test_md5.c File Reference . . . . . 723
6.156tests/test_ms_funcs.c File Reference . . . . . 725
6.157tests/test_sha1.c File Reference . . . . . 726
6.158tests/test_sha256.c File Reference . . . . . 728
6.159tests/test_x509v3.c File Reference . . . . . 730
6.160tls.h File Reference . . . . . 732
6.161tls_gnutls.c File Reference . . . . . 746
6.162tls_internal.c File Reference . . . . . 762
6.163tls_none.c File Reference . . . . . 780
6.164tls_openssl.c File Reference . . . . . 782
6.165tls_schannel.c File Reference . . . . . 796
6.166tlsv1_client.c File Reference . . . . . 811

6.167tlsv1_client.h File Reference . . . 822
6.168tlsv1_common.c File Reference . . . 832
6.169tlsv1_common.h File Reference . . . 838
6.170win_if_list.c File Reference . . . 846
6.171wpa.c File Reference . . . 848
6.172wpa.h File Reference . . . 861
6.173wpa_cli.c File Reference . . . 874
6.174wpa_common.h File Reference . . . 876
6.175wpa_ctrl.c File Reference . . . 878
6.176wpa_ctrl.h File Reference . . . 880
6.177wpa_gui-qt4/eventhistory.h File Reference . . . 886
6.178wpa_gui-qt4/networkconfig.h File Reference . . . 887
6.179wpa_gui-qt4/scanresults.h File Reference . . . 888
6.180wpa_gui-qt4/userdatarequest.h File Reference . . . 889
6.181wpa_gui-qt4/wpagui.h File Reference . . . 890
6.182wpa_gui-qt4/wpamsg.h File Reference . . . 891
6.183wpa_i.h File Reference . . . 892
6.184wpa_passphrase.c File Reference . . . 893
6.185wpa_supplicant.c File Reference . . . 895
6.186wpa_supplicant.h File Reference . . . 915
6.187wpa_supplicant_i.h File Reference . . . 919
6.188x509v3.c File Reference . . . 936
6.189x509v3.h File Reference . . . 938

**7 wpa_supplicant Example Documentation** **941**

7.1 com . . . 941

**8 wpa_supplicant Page Documentation** **943**

8.1 Structure of the source code . . . 943
8.2 Control interface . . . 947
8.3 Driver wrapper implementation (driver.h, drivers.c) . . . 954
8.4 EAP peer implementation . . . 957
8.5 Porting to different target boards and operating systems . . . 958
8.6 Testing and development tools . . . 961

# Chapter 1

# Developers' documentation for wpa_supplicant

wpa_supplicant is a WPA Supplicant for Linux, BSD and Windows with support for WPA and WPA2 (IEEE 802.11i / RSN). Supplicant is the IEEE 802.1X/WPA component that is used in the client stations. It implements key negotiation with a WPA Authenticator and it can optionally control roaming and IEEE 802.11 authentication/association of the wlan driver.

The goal of this documentation and comments in the source code is to give enough information for other developers to understand how wpa_supplicant has been implemented, how it can be modified, how new drivers can be supported, and how wpa_supplicant can be ported to other operating systems. If any information is missing, feel free to contact Jouni Malinen <jkmaline@cc.hut.fi> for more information. Contributions as patch files are also very welcome at the same address. Please note that wpa_supplicant is licensed under dual license, GPLv2 or BSD at user's choice. All contributions to wpa_supplicant are expected to use compatible licensing terms.

The source code and read-only access to wpa_supplicant CVS repository is available from the project home page at http://hostap.epitest.fi/wpa_supplicant/. This developers' documentation is also available as a PDF file from http://hostap.epitest.fi/wpa_supplicant/wpa_supplicant-devel.pdf .

The design goal for wpa_supplicant was to use hardware, driver, and OS independent, portable C code for all WPA functionality. The source code is divided into separate C files as shown on the code structure page. All hardware/driver specific functionality is in separate files that implement a well-defined driver API. Information about porting to different target boards and operating systems is available on the porting page.

EAPOL (IEEE 802.1X) state machines are implemented as a separate module that interacts with EAP peer implementation. In addition to programs aimed at normal production use, wpa_supplicant source tree includes number of testing and development tools that make it easier to test the programs without having to setup a full test setup with wireless cards. These tools can also be used to implement automatic test suites.

wpa_supplicant implements a control interface that can be used by external programs to control the operations of the wpa_supplicant daemon and to get status information and event notifications. There is a small C library that provides helper functions to facilitate the use of the control interface. This library can also be used with C++.

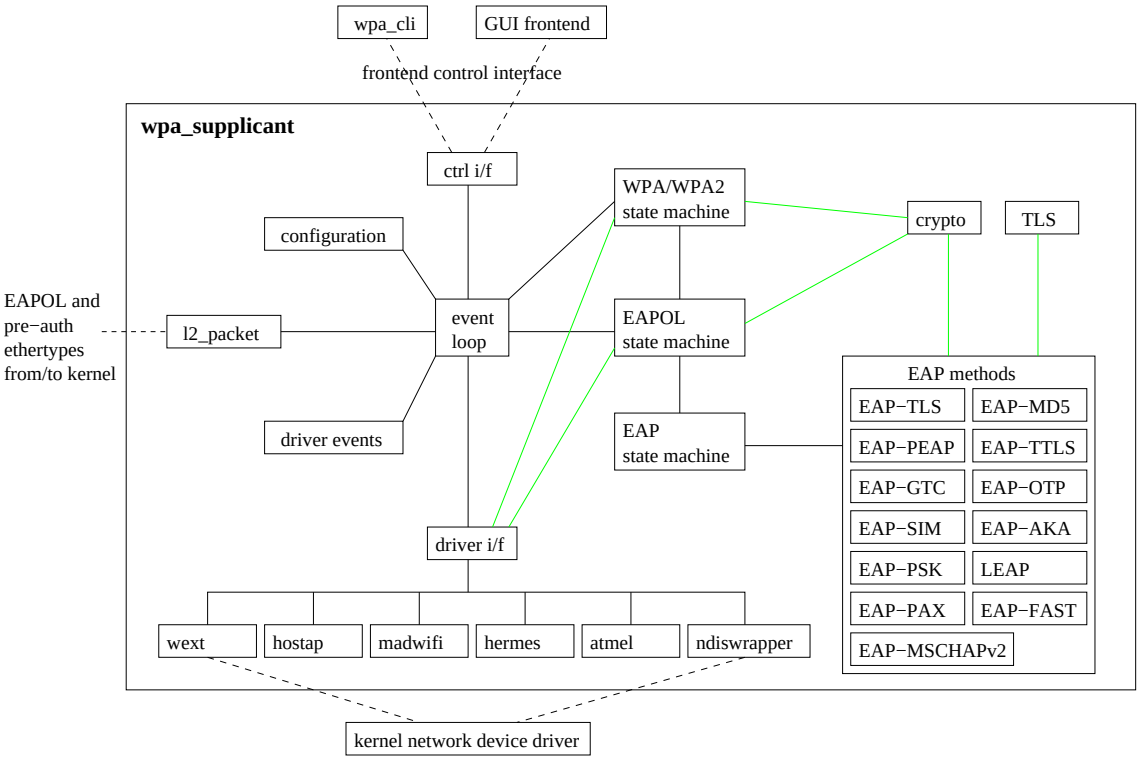



Figure 1.1: wpa_supplicant modules

# Chapter 2

# wpa_supplicant Data Structure Index

## 2.1 wpa_supplicant Data Structures

Here are the data structures with brief descriptions:


eap_config (Configuration for EAP state machine ) . . . 13
eap_method (EAP method interface ) . . . 15
eap_method_ret (EAP return values from struct eap_method::process() ) . . . 21
eap_sm (EAP state machine data ) . . . 22
eapol_callbacks (Callback functions from EAP to lower layer ) . . . 24
eapol_config (Per network configuration for EAPOL state machines ) . . . 28
eapol_ctx (Global (for all networks) EAPOL state machine context ) . . . 30
eapol_sm (Internal data for EAPOL state machines ) . . . 34
rsn_pmksa_cache_entry (PMKSA cache entry ) . . . 37
tls_connection_params (Parameters for TLS connection ) . . . 38
wpa_config (Wpa_supplicant configuration data ) . . . 40
wpa_config_blob (Named configuration blob ) . . . 45
wpa_ctrl (Internal structure for control interface library ) . . . 46
wpa_ctrl_dst (Internal data structure of control interface clients ) . . . 47
wpa_driver_associate_params (Association parameters ) . . . 48
wpa_driver_capa (Driver capability information ) . . . 50
wpa_driver_ops (Driver interface API definition ) . . . 51
wpa_event_data . . . 64
wpa_event_data::assoc_info (Data for EVENT_ASSOC and EVENT_ASSOCINFO events ) . . . 66
wpa_event_data::interface_status (Data for EVENT_INTERFACE_STATUS ) . . . 68
wpa_event_data::michael_mic_failure (Data for EVENT_MICHAEL_MIC_FAILURE ) . . . 69
wpa_event_data::pmkid_candidate (Data for EVENT_PMKID_CANDIDATE ) . . . 70
wpa_event_data::stkstart (Data for EVENT_STKSTART ) . . . 71
wpa_global (Internal, global data for all wpa_supplicant interfaces ) . . . 72
wpa_interface (Parameters for wpa_supplicant_add_iface() ) . . . 73
wpa_params (Parameters for wpa_supplicant_init() ) . . . 75
wpa_ptk (WPA Pairwise Transient Key ) . . . 77
wpa_scan_result (Scan results ) . . . 78
wpa_sm (Internal WPA state machine data ) . . . 79
wpa_ssid (Network configuration data ) . . . 81
wpa_supplicant (Internal data for wpa_supplicant interface ) . . . 97

# Chapter 3

# wpa_supplicant File Index

## 3.1 wpa_supplicant File List

Here is a list of all documented files with brief descriptions:


aes.c (AES (Rijndael) cipher ) . . . 99
aes.h (AES functions ) . . . 105
aes_wrap.c (AES-based functions ) . . . 106
aes_wrap.h (AES-based functions ) . . . 113
asn1.c (ASN.1 DER parsing ) . . . 120
asn1.h (ASN.1 DER parsing ) . . . 122
asn1_test.c (Testing tool for ASN.1/X.509v3 routines ) . . . 124
base64.c (Base64 encoding/decoding (RFC1341) ) . . . 126
base64.h (Base64 encoding/decoding (RFC1341) ) . . . 128
bignum.c (Big number math ) . . . 130
bignum.h (Big number math ) . . . 137
build_config.h (Wpa_supplicant/hostapd - Build time configuration defines ) . . . 143
common.c (Wpa_supplicant/hostapd / common helper functions, etc ) . . . 144
common.h (Wpa_supplicant/hostapd / common helper functions, etc ) . . . 151
config.c (WPA Supplicant / Configuration parser and common functions ) . . . 162
config.h (WPA Supplicant / Configuration file structures ) . . . 171
config_file.c (WPA Supplicant / Configuration backend: text file ) . . . 181
config_none.c (WPA Supplicant / Configuration backend: empty starting point ) . . . 184
config_ssid.h (WPA Supplicant / Network configuration structures ) . . . 187
config_types.h (Hostapd / Shared configuration file defines ) . . . 190
config_winreg.c (WPA Supplicant / Configuration backend: Windows registry ) . . . 191
crypto.c (WPA Supplicant / wrapper functions for libcrypto ) . . . 194
crypto.h (WPA Supplicant / wrapper functions for crypto libraries ) . . . 197
crypto_cryptoapi.c (WPA Supplicant / Crypto wrapper for Microsoft CryptoAPI ) . . . 210
crypto_gnutls.c (WPA Supplicant / wrapper functions for libgcrypt ) . . . 213
crypto_internal.c (WPA Supplicant / Crypto wrapper for internal crypto implementation ) . . . 215
crypto_libtomcrypt.c (WPA Supplicant / Crypto wrapper for LibTomCrypt (for internal TLSv1) ) 216
crypto_none.c (WPA Supplicant / Empty template functions for crypto wrapper ) . . . 219
ctrl_iface.c (WPA Supplicant / Control interface (shared code for all backends) ) . . . 221
ctrl_iface.h (WPA Supplicant / UNIX domain socket -based control interface ) . . . 225
ctrl_iface_dbus.c (WPA Supplicant / dbus-based control interface ) . . . 231
ctrl_iface_dbus.h (WPA Supplicant / dbus-based control interface ) . . . 239
ctrl_iface_dbus_handlers.c (WPA Supplicant / dbus-based control interface ) . . . 240

ctrl_iface_dbus_handlers.h (WPA Supplicant / dbus-based control interface ) . . . . . 252
ctrl_iface_named_pipe.c (WPA Supplicant / Windows Named Pipe -based control interface ) . . 253
ctrl_iface_udp.c (WPA Supplicant / UDP socket -based control interface ) . . . . . 257
ctrl_iface_unix.c (WPA Supplicant / UNIX domain socket -based control interface ) . . . . . 261
dbus_dict_helpers.c (WPA Supplicant / dbus-based control interface ) . . . . . 265
dbus_dict_helpers.h (WPA Supplicant / dbus-based control interface ) . . . . . 274
defs.h (WPA Supplicant - Common definitions ) . . . . . 283
des.c (DES and 3DES-EDE ciphers ) . . . . . 286
driver.h (WPA Supplicant - driver interface definition ) . . . . . 288
driver_atmel.c (WPA Supplicant - Driver interaction with Atmel Wireless LAN drivers ) . . . . 290
driver_broadcom.c (WPA Supplicant - driver interaction with Broadcom wl.o driver ) . . . . . 292
driver_bsd.c (WPA Supplicant - driver interaction with BSD net80211 layer ) . . . . . 295
driver_hostap.c (WPA Supplicant - driver interaction with Linux Host AP driver ) . . . . . 298
driver_hostap.h (WPA Supplicant - driver interaction with Linux Host AP driver ) . . . . . 300
driver_ipw.c (WPA Supplicant - driver interaction with Linux ipw2100/2200 drivers ) . . . . . 302
driver_madwifi.c (WPA Supplicant - driver interaction with MADWIFI 802.11 driver ) . . . . . 304
driver_ndis.c (WPA Supplicant - Windows/NDIS driver interface ) . . . . . 306
driver_ndis.h (WPA Supplicant - Windows/NDIS driver interface ) . . . . . 310
driver_ndis_.c (WPA Supplicant - Windows/NDIS driver interface - event processing ) . . . . . 311
driver_ndiswrapper.c (WPA Supplicant - driver interaction with Linux ndiswrapper ) . . . . . 313
driver_prism54.c (WPA Supplicant - driver interaction with Linux Prism54.org driver ) . . . . . 315
driver_test.c (WPA Supplicant - testing driver interface ) . . . . . 317
driver_wext.c (WPA Supplicant - driver interaction with generic Linux Wireless Extensions ) . . 319
driver_wext.h (WPA Supplicant - driver_wext exported functions ) . . . . . 327
driver_wired.c (WPA Supplicant - wired Ethernet driver interface ) . . . . . 334
drivers.c (WPA Supplicant / driver interface list ) . . . . . 336
eap.c (EAP peer state machines (RFC 4137) ) . . . . . 338
eap.h (EAP peer state machine functions (RFC 4137) ) . . . . . 354
eap_aka.c (EAP peer method: EAP-AKA (RFC 4187) ) . . . . . 366
eap_defs.h (EAP server/peer: Shared EAP definitions ) . . . . . 367
eap_fast.c (EAP peer method: EAP-FAST (draft-cam-winget-eap-fast-03.txt) ) . . . . . 369
eap_gpsk.c (EAP peer method: EAP-GPSK (draft-ietf-emu-eap-gpsk-01.txt) ) . . . . . 371
eap_gpsk_common.c (EAP server/peer: EAP-GPSK shared routines ) . . . . . 373
eap_gpsk_common.h (EAP server/peer: EAP-GPSK shared routines ) . . . . . 377
eap_gtc.c (EAP peer method: EAP-GTC (RFC 3748) ) . . . . . 381
eap_i.h (EAP peer state machines internal structures (RFC 4137) ) . . . . . 383
eap_leap.c (EAP peer method: LEAP ) . . . . . 391
eap_md5.c (EAP peer method: EAP-MD5 (RFC 3748 and RFC 1994) ) . . . . . 393
eap_methods.c (EAP peer: Method registration ) . . . . . 395
eap_methods.h (EAP peer: Method registration ) . . . . . 401
eap_mschapv2.c (EAP peer method: EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.txt) ) . . . . . 407
eap_otp.c (EAP peer method: EAP-OTP (RFC 3748) ) . . . . . 409
eap_pax.c (EAP peer method: EAP-PAX (draft-clancy-eap-pax-11.txt) ) . . . . . 411
eap_pax_common.c (EAP server/peer: EAP-PAX shared routines ) . . . . . 412
eap_pax_common.h (EAP server/peer: EAP-PAX shared routines ) . . . . . 416
eap_peap.c (EAP peer method: EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-07.txt) ) . . . . 420
eap_psk.c (EAP peer method: EAP-PSK (draft-bersani-eap-psk-11.txt) ) . . . . . 421
eap_psk_common.c (EAP server/peer: EAP-PSK shared routines ) . . . . . 422
eap_psk_common.h (EAP server/peer: EAP-PSK shared routines ) . . . . . 424
eap_sake.c (EAP peer method: EAP-SAKE (RFC 4763) ) . . . . . 425
eap_sake_common.c (EAP server/peer: EAP-SAKE shared routines ) . . . . . 427
eap_sake_common.h (EAP server/peer: EAP-SAKE shared routines ) . . . . . 430
eap_sim.c (EAP peer method: EAP-SIM (RFC 4186) ) . . . . . 433

eap_sim_common.c (EAP peer: EAP-SIM/AKA shared routines ) . . . 435
eap_sim_common.h (EAP peer: EAP-SIM/AKA shared routines ) . . . 437
eap_tls.c (EAP peer method: EAP-TLS (RFC 2716) ) . . . 440
eap_tls_common.c (EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions ) . . . 441
eap_tls_common.h (EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions ) . . . 448
eap_tlv.c (EAP peer method: EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt) ) . . . 455
eap_tlv.h (EAP peer method: EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt) ) . . . 458
eap_ttls.c (EAP peer method: EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt) ) . . . 461
eap_ttls.h (EAP server/peer: EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt) ) . . . 463
eap_vendor_test.c (EAP peer method: Test method for vendor specific (expanded) EAP type ) . . . 465
eapol_sm.c (WPA Supplicant / EAPOL state machines ) . . . 467
eapol_sm.h (WPA Supplicant / EAPOL state machines ) . . . 481
eapol_test.c (WPA Supplicant - test code ) . . . 494
eloop.c (Event loop based on select() loop ) . . . 497
eloop.h (Event loop ) . . . 504
eloop_none.c (Event loop - empty template (basic structure, but no OS specific operations) ) . . . 516
eloop_win.c (Event loop based on Windows events and WaitForMultipleObjects ) . . . 520
events.c (WPA Supplicant - Driver event processing ) . . . 528
**hostapd.h** . . . **??**
includes.h (Wpa_supplicant/hostapd - Default include files ) . . . 531
l2_packet.h (WPA Supplicant - Layer2 packet interface definition ) . . . 535
l2_packet_freebsd.c (WPA Supplicant - Layer2 packet handling with FreeBSD ) . . . 541
l2_packet_linux.c (WPA Supplicant - Layer2 packet handling with Linux packet sockets ) . . . 545
l2_packet_ndis.c (WPA Supplicant - Layer2 packet handling with Microsoft NDISUIO ) . . . 549
l2_packet_none.c (WPA Supplicant - Layer2 packet handling example with dummy functions ) . . . 554
l2_packet_pcap.c (WPA Supplicant - Layer2 packet handling with libpcap/libdnet and WinPcap ) . . . 558
l2_packet_winpcap.c (WPA Supplicant - Layer2 packet handling with WinPcap RX thread ) . . . 563
libtommath.c (Minimal code for RSA support from LibTomMath 0.3.9 http://math.libtomcrypt.com/ http://math.libtomcrypt.com/files/ltm-0.39.tar.bz2 This library was released in public domain by Tom St Denis ) . . . 568
main.c (WPA Supplicant / main() function for UNIX like OSes and MinGW ) . . . 570
main_none.c (WPA Supplicant / Example program entrypoint ) . . . 572
main_winmain.c (WPA Supplicant / WinMain() function for Windows-based applications ) . . . 574
main_winsvc.c (WPA Supplicant / main() function for Win32 service ) . . . 576
md4.c (MD4 hash implementation ) . . . 578
md5.c (MD5 hash implementation and interface functions ) . . . 580
md5.h (MD5 hash implementation and interface functions ) . . . 583
mlme.c (WPA Supplicant - Client mode MLME ) . . . 585
mlme.h (WPA Supplicant - Client mode MLME ) . . . 589
ms_funcs.c (WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759 ) . . . 590
ms_funcs.h (WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759 ) . . . 598
ndis_events.c (Ndis_events - Receive NdisMIndicateStatus() events using WMI ) . . . 605
os.h (Wpa_supplicant/hostapd / OS specific functions ) . . . 607
os_internal.c (Wpa_supplicant/hostapd / Internal implementation of OS specific functions ) . . . 614
os_none.c (Wpa_supplicant/hostapd / Empty OS specific functions ) . . . 620
os_unix.c (Wpa_supplicant/hostapd / OS specific functions for UNIX/POSIX systems ) . . . 626
os_win32.c (Wpa_supplicant/hostapd / OS specific functions for Win32 systems ) . . . 632
pcsc_funcs.c (WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM ) . . . 638
pcsc_funcs.h (WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM ) . . . 644
pmksa_cache.c (WPA Supplicant - RSN PMKSA cache ) . . . 646
pmksa_cache.h (Wpa_supplicant - WPA2/RSN PMKSA cache functions ) . . . 652
preauth.c (WPA Supplicant - RSN pre-authentication ) . . . 657
preauth.h (Wpa_supplicant - WPA2/RSN pre-authentication functions ) . . . 663

preauth_test.c (WPA Supplicant - test code for pre-authentication ) . . . 669
priv_netlink.h (Wpa_supplicant - Private copy of Linux netlink/rtnetlink definitions ) . . . 672
radius.c (Hostapd / RADIUS message processing ) . . . 674
radius.h (Hostapd / RADIUS message processing ) . . . 677
radius_client.c (Hostapd / RADIUS client ) . . . 681
radius_client.h (Hostapd / RADIUS client ) . . . 683
rc4.c (RC4 stream cipher ) . . . 685
rc4.h (RC4 stream cipher ) . . . 688
rsa.c (RSA ) . . . 690
rsa.h (RSA ) . . . 694
sha1.c (SHA1 hash implementation and interface functions ) . . . 698
sha1.h (SHA1 hash implementation and interface functions ) . . . 703
sha256.c (SHA-256 hash implementation and interface functions ) . . . 708
sha256.h (SHA256 hash implementation and interface functions ) . . . 711
state_machine.h (Wpa_supplicant/hostapd - State machine definitions ) . . . 714
tls.h (WPA Supplicant / SSL/TLS interface definition ) . . . 732
tls_gnutls.c (WPA Supplicant / SSL/TLS interface functions for openssl ) . . . 746
tls_internal.c (WPA Supplicant / TLS interface functions and an internal TLS implementation ) . . . 762
tls_none.c (WPA Supplicant / SSL/TLS interface functions for no TLS case ) . . . 780
tls_openssl.c (WPA Supplicant / SSL/TLS interface functions for openssl ) . . . 782
tls_schannel.c (WPA Supplicant / SSL/TLS interface functions for Microsoft Schannel ) . . . 796
tlsv1_client.c (Wpa_supplicant: TLSv1 client (RFC 2246) ) . . . 811
tlsv1_client.h (Wpa_supplicant: TLSv1 client (RFC 2246) ) . . . 822
tlsv1_common.c (Wpa_supplicant/hostapd: TLSv1 common routines ) . . . 832
tlsv1_common.h (Wpa_supplicant/hostapd: TLSv1 common definitions ) . . . 838
**version.h** . . . **??**
win_if_list.c (Win_if_list - Display network interfaces with description (for Windows) ) . . . 846
**wireless_copy.h** . . . **??**
wpa.c (WPA Supplicant - WPA state machine and EAPOL-Key processing ) . . . 848
wpa.h (Wpa_supplicant - WPA definitions ) . . . 861
wpa_cli.c (WPA Supplicant - command line interface for wpa_supplicant daemon ) . . . 874
wpa_common.h (WPA definitions shared between hostapd and wpa_supplicant ) . . . 876
wpa_ctrl.c (Wpa_supplicant/hostapd control interface library ) . . . 878
wpa_ctrl.h (Wpa_supplicant/hostapd control interface library ) . . . 880
wpa_i.h (Wpa_supplicant - Internal WPA state machine definitions ) . . . 892
wpa_passphrase.c (WPA Supplicant - ASCII passphrase to WPA PSK tool ) . . . 893
wpa_supplicant.c (WPA Supplicant ) . . . 895
wpa_supplicant.h (Wpa_supplicant - Exported functions for wpa_supplicant modules ) . . . 915
wpa_supplicant_i.h (Wpa_supplicant - Internal definitions ) . . . 919
x509v3.c (X.509v3 certificate parsing and processing (RFC 3280 profile) ) . . . 936
x509v3.h (X.509v3 certificate parsing and processing ) . . . 938
doc/**code_structure.doxygen** . . . **??**
doc/**ctrl_iface.doxygen** . . . **??**
doc/**driver_wrapper.doxygen** . . . **??**
doc/**eap.doxygen** . . . **??**
doc/**mainpage.doxygen** . . . **??**
doc/**porting.doxygen** . . . **??**
doc/**testing_tools.doxygen** . . . **??**
tests/test_aes.c (Test program for AES ) . . . 718
tests/test_eap_sim_common.c (Test program for EAP-SIM PRF ) . . . 720
tests/test_md4.c (Test program for MD4 (test vectors from RFC 1320) ) . . . 721
tests/test_md5.c (Test program for MD5 (test vectors from RFC 1321) ) . . . 723
tests/test_ms_funcs.c (Test program for ms_funcs ) . . . 725
tests/test_sha1.c (Test program for SHA1 and MD5 ) . . . 726

tests/test_sha256.c (Test program for SHA256 ) . . . 728
tests/test_x509v3.c (Testing tool for X.509v3 routines ) . . . 730
wpa_gui-qt4/eventhistory.h (Wpa_gui - EventHistory class ) . . . 886
wpa_gui-qt4/networkconfig.h (Wpa_gui - NetworkConfig class ) . . . 887
wpa_gui-qt4/scanresults.h (Wpa_gui - ScanResults class ) . . . 888
wpa_gui-qt4/userdatarequest.h (Wpa_gui - UserDataRequest class ) . . . 889
wpa_gui-qt4/wpagui.h (Wpa_gui - WpaGui class ) . . . 890
wpa_gui-qt4/wpamsg.h (Wpa_gui - WpaMsg class for storing event messages ) . . . 891
wpa_gui/**eventhistory.ui.h** . . . **??**
wpa_gui/**networkconfig.ui.h** . . . **??**
wpa_gui/**scanresults.ui.h** . . . **??**
wpa_gui/**userdatarequest.ui.h** . . . **??**
wpa_gui/**wpagui.ui.h** . . . **??**
wpa_gui/**wpamsg.h** . . . **??**

# Chapter 4

# wpa_supplicant Page Index

## 4.1 wpa_supplicant Related Pages

Here is a list of all related documentation pages:


Structure of the source code . . . . . 943
Control interface . . . . . 947
Driver wrapper implementation (driver.h, drivers.c) . . . . . 954
EAP peer implementation . . . . . 957
Porting to different target boards and operating systems . . . . . 958
Testing and development tools . . . . . 961

# Chapter 5

# wpa_supplicant Data Structure Documentation

## 5.1 eap_config Struct Reference

Configuration for EAP state machine.

`#include <eap.h>`

### Data Fields

- const char * opensc_engine_path
  *OpenSC engine for OpenSSL engine support.*
- const char * pkcs11_engine_path
  *PKCS#11 engine for OpenSSL engine support.*
- const char * pkcs11_module_path
  *OpenSC PKCS#11 module for OpenSSL engine.*

### 5.1.1 Detailed Description

Configuration for EAP state machine.

Definition at line 241 of file eap.h.

### 5.1.2 Field Documentation

#### 5.1.2.1 const char* eap_config::opensc_engine_path

OpenSC engine for OpenSSL engine support.

Usually, path to engine_opensc.so.

Definition at line 248 of file eap.h.

#### 5.1.2.2 const char* eap_config::pkcs11_engine_path

PKCS#11 engine for OpenSSL engine support.

Usually, path to engine_pkcs11.so.

Definition at line 255 of file eap.h.

#### 5.1.2.3 const char* eap_config::pkcs11_module_path

OpenSC PKCS#11 module for OpenSSL engine.

Usually, path to opensc-pkcs11.so.

Definition at line 262 of file eap.h.

The documentation for this struct was generated from the following file:

- eap.h

## 5.2 eap_method Struct Reference

EAP method interface.

`#include <eap_i.h>`

Collaboration diagram for eap_method:


### Data Fields

- int vendor

  *EAP Vendor-ID (EAP_VENDOR_∗) (0 = IETF).*

- EapType method

  *EAP type number (EAP_TYPE_∗).*

- const char ∗ name

  *Name of the method (e.g., "TLS").*

- void ∗(∗ init )(struct eap_sm ∗sm)

  *Initialize an EAP method.*

- void(∗ deinit )(struct eap_sm ∗sm, void ∗priv)

  *Deinitialize an EAP method.*

- u8 ∗(∗ process )(struct eap_sm ∗sm, void ∗priv, struct eap_method_ret ∗ret, const u8 ∗reqData, size_t reqDataLen, size_t ∗respDataLen)

  *Process an EAP request.*

- Boolean(∗ isKeyAvailable )(struct eap_sm ∗sm, void ∗priv)

  *Find out whether EAP method has keying material.*

- u8 ∗(∗ getKey )(struct eap_sm ∗sm, void ∗priv, size_t ∗len)

  *Get EAP method specific keying material (eapKeyData).*

- int(∗ get_status )(struct eap_sm ∗sm, void ∗priv, char ∗buf, size_t buflen, int verbose)

  *Get EAP method status.*

- Boolean(∗ has_reauth_data )(struct eap_sm ∗sm, void ∗priv)

  *Whether method is ready for fast reauthentication.*

- void(∗ deinit_for_reauth )(struct eap_sm ∗sm, void ∗priv)

  *Release data that is not needed for fast re-auth.*

- void ∗(∗ init_for_reauth )(struct eap_sm ∗sm, void ∗priv)

  *Prepare for start of fast re-authentication.*

- const u8 *(* get_identity )(struct eap_sm *sm, void *priv, size_t *len)

  *Get method specific identity for re-authentication.*

- void(* free )(struct eap_method *method)

  *Free EAP method data.*

- int version

  *Version of the EAP peer method interface.*

- eap_method * next

  *Pointer to the next EAP method.*

- u8 *(* get_emsk )(struct eap_sm *sm, void *priv, size_t *len)

  *Get EAP method specific keying extended material (EMSK).*

### 5.2.1 Detailed Description

EAP method interface.

This structure defines the EAP method interface. Each method will need to register its own EAP type, EAP name, and set of function pointers for method specific operations. This interface is based on section 4.4 of RFC 4137.

Definition at line 74 of file eap_i.h.

### 5.2.2 Field Documentation

#### 5.2.2.1 void(* eap_method::deinit)(struct eap_sm *sm, void *priv)

Deinitialize an EAP method.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***priv*** Pointer to private EAP method data from eap_method::init()

Deinitialize the EAP method and free any allocated private data.

#### 5.2.2.2 void(* eap_method::deinit_for_reauth)(struct eap_sm *sm, void *priv)

Release data that is not needed for fast re-auth.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***priv*** Pointer to private EAP method data from eap_method::init()

This function is an optional handler that only EAP methods supporting fast re-authentication need to implement. This is called when authentication has been completed and EAP state machine is requesting that enough state information is maintained for fast re-authentication

#### 5.2.2.3 void(* eap_method::free)(struct eap_method *method)

Free EAP method data.

**Parameters:**
***method*** Pointer to the method data registered with eap_peer_method_register().

This function will be called when the EAP method is being unregistered. If the EAP method allocated resources during registration (e.g., allocated struct eap_method), they should be freed in this function. No other method functions will be called after this call. If this function is not defined (i.e., function pointer is NULL), a default handler is used to release the method data with free(method). This is suitable for most cases.

#### 5.2.2.4 u8*(* eap_method::get_emsk)(struct eap_sm *sm, void *priv, size_t *len)

Get EAP method specific keying extended material (EMSK).

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()
***priv*** Pointer to private EAP method data from eap_method::init()
***len*** Pointer to a variable to store EMSK length

**Returns:**
EMSK or NULL if not available

This function can be used to get the extended keying material from the EAP method. The key may already be stored in the method-specific private data or this function may derive the key.

#### 5.2.2.5 const u8*(* eap_method::get_identity)(struct eap_sm *sm, void *priv, size_t *len)

Get method specific identity for re-authentication.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()
***priv*** Pointer to private EAP method data from eap_method::init()
***len*** Length of the returned identity

**Returns:**
Pointer to the method specific identity or NULL if default identity is to be used

This function is an optional handler that only EAP methods that use method specific identity need to implement.

#### 5.2.2.6 int(* eap_method::get_status)(struct eap_sm *sm, void *priv, char *buf, size_t buflen, int verbose)

Get EAP method status.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***priv*** Pointer to private EAP method data from eap_method::init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

**Returns:**
Number of bytes written to buf

Query EAP method for status information. This function fills in a text area with current status information from the EAP method. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

#### 5.2.2.7 u8∗(∗ eap_method::getKey)(struct eap_sm ∗sm, void ∗priv, size_t ∗len)

Get EAP method specific keying material (eapKeyData).

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***priv*** Pointer to private EAP method data from eap_method::init()

***len*** Pointer to variable to store key length (eapKeyDataLen)

**Returns:**
Keying material (eapKeyData) or NULL if not available

This function can be used to get the keying material from the EAP method. The key may already be stored in the method-specific private data or this function may derive the key.

#### 5.2.2.8 Boolean(∗ eap_method::has_reauth_data)(struct eap_sm ∗sm, void ∗priv)

Whether method is ready for fast reauthentication.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***priv*** Pointer to private EAP method data from eap_method::init()

**Returns:**
TRUE or FALSE based on whether fast reauthentication is possible

This function is an optional handler that only EAP methods supporting fast re-authentication need to implement.

#### 5.2.2.9 void∗(∗ eap_method::init)(struct eap_sm ∗sm)

Initialize an EAP method.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

**Returns:**
Pointer to allocated private data, or NULL on failure

This function is used to initialize the EAP method explicitly instead of using METHOD_INIT state as specific in RFC 4137. The method is expected to initialize it method-specific state and return a pointer that will be used as the priv argument to other calls.

#### 5.2.2.10 void*(* eap_method::init_for_reauth)(struct eap_sm *sm, void *priv)

Prepare for start of fast re-authentication.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()
***priv*** Pointer to private EAP method data from eap_method::init()

This function is an optional handler that only EAP methods supporting fast re-authentication need to implement. This is called when EAP authentication is started and EAP state machine is requesting fast re-authentication to be used.

#### 5.2.2.11 Boolean(* eap_method::isKeyAvailable)(struct eap_sm *sm, void *priv)

Find out whether EAP method has keying material.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()
***priv*** Pointer to private EAP method data from eap_method::init()

**Returns:**
TRUE if key material (eapKeyData) is available

#### 5.2.2.12 struct eap_method* eap_method::next

Pointer to the next EAP method.

This variable is used internally in the EAP method registration code to create a linked list of registered EAP methods.

Definition at line 269 of file eap_i.h.

#### 5.2.2.13 u8*(* eap_method::process)(struct eap_sm *sm, void *priv, struct eap_method_ret *ret, const u8 *reqData, size_t reqDataLen, size_t *respDataLen)

Process an EAP request.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()
***priv*** Pointer to private EAP method data from eap_method::init()
***ret*** Return values from EAP request validation and processing
***reqData*** EAP request to be processed (eapReqData)

***reqDataLen*** Length of the EAP request

***respDataLen*** Length of the returned EAP response

**Returns:**
Pointer to allocated EAP response packet (eapRespData)

This function is a combination of m.check(), m.process(), and m.buildResp() procedures defined in section 4.4 of RFC 4137 In other words, this function validates the incoming request, processes it, and build a response packet. m.check() and m.process() return values are returned through struct eap_method_ret *ret variable. Caller is responsible for freeing the returned EAP response packet.

#### 5.2.2.14 int eap_method::version

Version of the EAP peer method interface.

The EAP peer method implementation should set this variable to EAP_PEER_METHOD_INTERFACE_-VERSION. This is used to verify that the EAP method is using supported API version when using dynamically loadable EAP methods.

Definition at line 260 of file eap_i.h.

The documentation for this struct was generated from the following file:

- eap_i.h

## 5.3 eap_method_ret Struct Reference

EAP return values from struct eap_method::process().

`#include <eap_i.h>`

### Data Fields

- Boolean ignore

  *Whether method decided to drop the current packed (OUT).*

- EapMethodState methodState

  *Method-specific state (IN/OUT).*

- EapDecision decision

  *Authentication decision (OUT).*

- Boolean allowNotifications

  *Whether method allows notifications (OUT).*

### 5.3.1 Detailed Description

EAP return values from struct eap_method::process().

These structure contains OUT variables for the interface between peer state machine and methods (RFC 4137, Sect. 4.2). eapRespData will be returned as the return value of struct eap_method::process() so it is not included in this structure.

Definition at line 40 of file eap_i.h.

The documentation for this struct was generated from the following file:

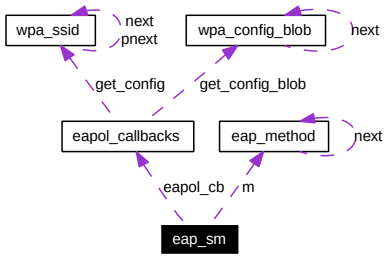
- eap_i.h

## 5.4 eap_sm Struct Reference

EAP state machine data.

`#include <eap_i.h>`

Collaboration diagram for eap_sm:




### Public Types

- enum {
  **EAP_INITIALIZE**, **EAP_DISABLED**, **EAP_IDLE**, **EAP_RECEIVED**,
  **EAP_GET_METHOD**, **EAP_METHOD**, **EAP_SEND_RESPONSE**, **EAP_DISCARD**,
  **EAP_IDENTITY**, **EAP_NOTIFICATION**, **EAP_RETRANSMIT**, **EAP_SUCCESS**,
  **EAP_FAILURE** }

### Data Fields

- enum eap_sm:: { ... } **EAP_state**
- EapType **selectedMethod**
- EapMethodState **methodState**
- int **lastId**
- u8 * **lastRespData**
- size_t **lastRespDataLen**
- EapDecision **decision**
- Boolean **rxReq**
- Boolean **rxSuccess**
- Boolean **rxFailure**
- int **reqId**
- EapType **reqMethod**
- int **reqVendor**
- u32 **reqVendorMethod**
- Boolean **ignore**
- int **ClientTimeout**
- Boolean **allowNotifications**
- u8 * **eapRespData**
- size_t **eapRespDataLen**
- Boolean **eapKeyAvailable**
- u8 * **eapKeyData**
- size_t **eapKeyDataLen**

- const struct eap_method * **m**
- Boolean **changed**
- void * **eapol_ctx**
- eapol_callbacks * **eapol_cb**
- void * **eap_method_priv**
- int **init_phase2**
- int **fast_reauth**
- Boolean **rxResp**
- Boolean **leap_done**
- Boolean **peap_done**
- u8 **req_md5** [16]
- u8 **last_md5** [16]
- void * **msg_ctx**
- void * **scard_ctx**
- void * **ssl_ctx**
- unsigned int **workaround**
- u8 * **peer_challenge**
- u8 * **auth_challenge**
- int **mschapv2_full_key**
- int **num_rounds**
- int **force_disabled**

### 5.4.1 Detailed Description

EAP state machine data.

Definition at line 303 of file eap_i.h.

The documentation for this struct was generated from the following file:

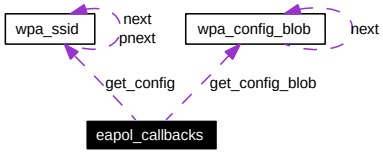
- eap_i.h

## 5.5 eapol_callbacks Struct Reference

Callback functions from EAP to lower layer.

`#include <eap.h>`

Collaboration diagram for eapol_callbacks:




### Data Fields

- wpa_ssid *(* get_config )(void *ctx)
  *Get pointer to the current network configuration.*
- Boolean(* get_bool )(void *ctx, enum eapol_bool_var variable)
  *Get a boolean EAPOL state variable.*
- void(* set_bool )(void *ctx, enum eapol_bool_var variable, Boolean value)
  *Set a boolean EAPOL state variable.*
- unsigned int(* get_int )(void *ctx, enum eapol_int_var variable)
  *Get an integer EAPOL state variable.*
- void(* set_int )(void *ctx, enum eapol_int_var variable, unsigned int value)
  *Set an integer EAPOL state variable.*
- u8 *(* get_eapReqData )(void *ctx, size_t *len)
  *Get EAP-Request data.*
- void(* set_config_blob )(void *ctx, struct wpa_config_blob *blob)
  *Set named configuration blob.*
- const struct wpa_config_blob *(* get_config_blob )(void *ctx, const char *name)
  *Get a named configuration blob.*
- void(* notify_pending )(void *ctx)
  *Notify that a pending request can be retried.*

### 5.5.1 Detailed Description

Callback functions from EAP to lower layer.

This structure defines the callback functions that EAP state machine requires from the lower layer (usually EAPOL state machine) for updating state variables and requesting information. eapol_ctx from eap_sm_init() call will be used as the ctx parameter for these callback functions.

Definition at line 147 of file eap.h.

## 5.5.2 Field Documentation

### 5.5.2.1 Boolean(* eapol_callbacks::get_bool)(void *ctx, enum eapol_bool_var variable)

Get a boolean EAPOL state variable.

**Parameters:**
***variable*** EAPOL boolean variable to get

**Returns:**
Value of the EAPOL variable

### 5.5.2.2 struct wpa_ssid*(* eapol_callbacks::get_config)(void *ctx)

Get pointer to the current network configuration.

**Parameters:**
***ctx*** eapol_ctx from eap_sm_init() call

### 5.5.2.3 const struct wpa_config_blob*(* eapol_callbacks::get_config_blob)(void *ctx, const char *name)

Get a named configuration blob.

**Parameters:**
***ctx*** eapol_ctx from eap_sm_init() call

***name*** Name of the blob

**Returns:**
Pointer to blob data or NULL if not found

### 5.5.2.4 u8*(* eapol_callbacks::get_eapReqData)(void *ctx, size_t *len)

Get EAP-Request data.

**Parameters:**
***ctx*** eapol_ctx from eap_sm_init() call

***len*** Pointer to variable that will be set to eapReqDataLen

**Returns:**
Reference to eapReqData (EAP state machine will not free this) or NULL if eapReqData not available.

#### 5.5.2.5 unsigned int(∗ eapol_callbacks::get_int)(void ∗ctx, enum eapol_int_var variable)

Get an integer EAPOL state variable.

**Parameters:**

***ctx*** eapol_ctx from eap_sm_init() call

***variable*** EAPOL integer variable to get

**Returns:**

Value of the EAPOL variable

#### 5.5.2.6 void(∗ eapol_callbacks::notify_pending)(void ∗ctx)

Notify that a pending request can be retried.

**Parameters:**

***ctx*** eapol_ctx from eap_sm_init() call

An EAP method can perform a pending operation (e.g., to get a response from an external process). Once the response is available, this callback function can be used to request EAPOL state machine to retry delivering the previously received (and still unanswered) EAP request to EAP state machine.

#### 5.5.2.7 void(∗ eapol_callbacks::set_bool)(void ∗ctx, enum eapol_bool_var variable, Boolean value)

Set a boolean EAPOL state variable.

**Parameters:**

***ctx*** eapol_ctx from eap_sm_init() call

***variable*** EAPOL boolean variable to set

***value*** Value for the EAPOL variable

#### 5.5.2.8 void(∗ eapol_callbacks::set_config_blob)(void ∗ctx, struct wpa_config_blob ∗blob)

Set named configuration blob.

**Parameters:**

***ctx*** eapol_ctx from eap_sm_init() call

***blob*** New value for the blob

Adds a new configuration blob or replaces the current value of an existing blob.

#### 5.5.2.9 void(∗ eapol_callbacks::set_int)(void ∗ctx, enum eapol_int_var variable, unsigned int value)

Set an integer EAPOL state variable.

**Parameters:**

***ctx*** eapol_ctx from eap_sm_init() call

***variable*** EAPOL integer variable to set

***value*** Value for the EAPOL variable

The documentation for this struct was generated from the following file:

- eap.h

## 5.6 eapol_config Struct Reference

Per network configuration for EAPOL state machines.

`#include <eapol_sm.h>`

### Data Fields

- int accept_802_1x_keys

  *Accept IEEE 802.1X (non-WPA) EAPOL-Key frames.*

- int required_keys

  *Which EAPOL-Key packets are required.*

- int fast_reauth

  *Whether fast EAP reauthentication is enabled.*

- unsigned int workaround

  *Whether EAP workarounds are enabled.*

- int eap_disabled

  *Whether EAP is disabled.*

### 5.6.1 Detailed Description

Per network configuration for EAPOL state machines.

Definition at line 28 of file eapol_sm.h.

### 5.6.2 Field Documentation

#### 5.6.2.1 int eapol_config::accept_802_1x_keys

Accept IEEE 802.1X (non-WPA) EAPOL-Key frames.

This variable should be set to 1 when using EAPOL state machines with non-WPA security policy to generate dynamic WEP keys. When using WPA, this should be set to 0 so that WPA state machine can process the EAPOL-Key frames.

Definition at line 38 of file eapol_sm.h.

#### 5.6.2.2 int eapol_config::required_keys

Which EAPOL-Key packets are required.

This variable determines which EAPOL-Key packets are required before marking connection authenticated. This is a bit field of EAPOL_REQUIRE_KEY_UNICAST and EAPOL_REQUIRE_KEY_BROADCAST flags.

Definition at line 50 of file eapol_sm.h.

The documentation for this struct was generated from the following file:

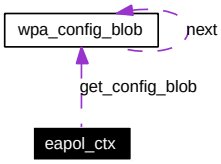
- eapol_sm.h

## 5.7 eapol_ctx Struct Reference

Global (for all networks) EAPOL state machine context.

`#include <eapol_sm.h>`

Collaboration diagram for eapol_ctx:




### Data Fields

- void * ctx
  *Pointer to arbitrary upper level context.*
- int preauth
  *IEEE 802.11i/RSN pre-authentication.*
- void(* cb )(struct eapol_sm *eapol, int success, void *ctx)
  *Function to be called when EAPOL negotiation has been completed.*
- void * cb_ctx
  *Callback context for cb().*
- void * msg_ctx
  *Callback context for wpa_msg() calls.*
- void * scard_ctx
  *Callback context for PC/SC scard_*() function calls.*
- void * eapol_send_ctx
  *Callback context for eapol_send() calls.*
- void(* eapol_done_cb )(void *ctx)
  *Function to be called at successful completion.*
- int(* eapol_send )(void *ctx, int type, const u8 *buf, size_t len)
  *Send EAPOL packets.*
- int(* set_wep_key )(void *ctx, int unicast, int keyidx, const u8 *key, size_t keylen)
  *Configure WEP keys.*
- void(* set_config_blob )(void *ctx, struct wpa_config_blob *blob)
  *Set or add a named configuration blob.*
- const struct wpa_config_blob *(* get_config_blob )(void *ctx, const char *name)

*Get a named configuration blob.*

- void(∗ aborted_cached )(void ∗ctx)
  *Notify that cached PMK attempt was aborted.*
- const char ∗ opensc_engine_path
  *Path to the OpenSSL engine for opensc.*
- const char ∗ pkcs11_engine_path
  *Path to the OpenSSL engine for PKCS#11.*
- const char ∗ pkcs11_module_path
  *Path to the OpenSSL OpenSC/PKCS#11 module.*

## 5.7.1 Detailed Description

Global (for all networks) EAPOL state machine context.

Definition at line 78 of file eapol_sm.h.

## 5.7.2 Field Documentation

### 5.7.2.1 void(∗ eapol_ctx::aborted_cached)(void ∗ctx)

Notify that cached PMK attempt was aborted.

**Parameters:**

***ctx*** Callback context (ctx)

### 5.7.2.2 void(∗ eapol_ctx::cb)(struct eapol_sm ∗eapol, int success, void ∗ctx)

Function to be called when EAPOL negotiation has been completed.

**Parameters:**

***eapol*** Pointer to EAPOL state machine data

***success*** Whether the authentication was completed successfully

***ctx*** Pointer to context data (cb_ctx)

This optional callback function will be called when the EAPOL authentication has been completed. This allows the owner of the EAPOL state machine to process the key and terminate the EAPOL state machine. Currently, this is used only in RSN pre-authentication.

### 5.7.2.3 void(∗ eapol_ctx::eapol_done_cb)(void ∗ctx)

Function to be called at successful completion.

**Parameters:**

***ctx*** Callback context (ctx)

This function is called at the successful completion of EAPOL authentication. If dynamic WEP keys are used, this is called only after all the expected keys have been received.

#### 5.7.2.4 int(* eapol_ctx::eapol_send)(void *ctx, int type, const u8 *buf, size_t len)

Send EAPOL packets.

**Parameters:**

***ctx*** Callback context (eapol_send_ctx)

***type*** EAPOL type (IEEE802_1X_TYPE_*)

***buf*** Pointer to EAPOL payload

***len*** Length of the EAPOL payload

**Returns:**

0 on success, -1 on failure

#### 5.7.2.5 const struct wpa_config_blob*(* eapol_ctx::get_config_blob)(void *ctx, const char *name)

Get a named configuration blob.

**Parameters:**

***ctx*** Callback context (ctx)

***name*** Name of the blob

**Returns:**

Pointer to blob data or NULL if not found

#### 5.7.2.6 const char* eapol_ctx::opensc_engine_path

Path to the OpenSSL engine for opensc.

This is an OpenSSL specific configuration option for loading OpenSC engine (engine_opensc.so); if NULL, this engine is not loaded.

Definition at line 204 of file eapol_sm.h.

#### 5.7.2.7 const char* eapol_ctx::pkcs11_engine_path

Path to the OpenSSL engine for PKCS#11.

This is an OpenSSL specific configuration option for loading PKCS#11 engine (engine_pkcs11.so); if NULL, this engine is not loaded.

Definition at line 213 of file eapol_sm.h.

#### 5.7.2.8 const char* eapol_ctx::pkcs11_module_path

Path to the OpenSSL OpenSC/PKCS#11 module.

This is an OpenSSL specific configuration option for configuring path to OpenSC/PKCS#11 engine (opensc-pkcs11.so); if NULL, this module is not loaded.

Definition at line 223 of file eapol_sm.h.

#### 5.7.2.9 int eapol_ctx::preauth

IEEE 802.11i/RSN pre-authentication.

This EAPOL state machine is used for IEEE 802.11i/RSN pre-authentication

Definition at line 92 of file eapol_sm.h.

#### 5.7.2.10 void* eapol_ctx::scard_ctx

Callback context for PC/SC scard_*() function calls.

This context can be updated with eapol_sm_register_scard_ctx().

Definition at line 126 of file eapol_sm.h.

#### 5.7.2.11 void(* eapol_ctx::set_config_blob)(void *ctx, struct wpa_config_blob *blob)

Set or add a named configuration blob.

**Parameters:**

***ctx*** Callback context (ctx)

***blob*** New value for the blob

Adds a new configuration blob or replaces the current value of an existing blob.

#### 5.7.2.12 int(* eapol_ctx::set_wep_key)(void *ctx, int unicast, int keyidx, const u8 *key, size_t keylen)

Configure WEP keys.

**Parameters:**

***ctx*** Callback context (ctx)

***unicast*** Non-zero = unicast, 0 = multicast/broadcast key

***keyidx*** Key index (0..3)

***key*** WEP key

***keylen*** Length of the WEP key

**Returns:**

0 on success, -1 on failure

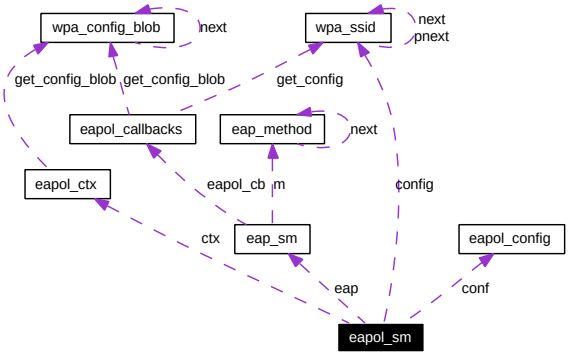
The documentation for this struct was generated from the following file:

- eapol_sm.h

## 5.8 eapol_sm Struct Reference

Internal data for EAPOL state machines.

Collaboration diagram for eapol_sm:




### Public Types

- enum {

  **SUPP_PAE_UNKNOWN** = 0, **SUPP_PAE_DISCONNECTED** = 1, **SUPP_PAE_LOGOFF** = 2, **SUPP_PAE_CONNECTING** = 3,

  **SUPP_PAE_AUTHENTICATING** = 4, **SUPP_PAE_AUTHENTICATED** = 5, **SUPP_PAE_-HELD** = 7, **SUPP_PAE_RESTART** = 8,

  **SUPP_PAE_S_FORCE_AUTH** = 9, **SUPP_PAE_S_FORCE_UNAUTH** = 10 }
- enum { **KEY_RX_UNKNOWN** = 0, **KEY_RX_NO_KEY_RECEIVE**, **KEY_RX_KEY_-RECEIVE** }
- enum {

  **SUPP_BE_UNKNOWN** = 0, **SUPP_BE_INITIALIZE** = 1, **SUPP_BE_IDLE** = 2, **SUPP_BE_-REQUEST** = 3,

  **SUPP_BE_RECEIVE** = 4, **SUPP_BE_RESPONSE** = 5, **SUPP_BE_FAIL** = 6, **SUPP_BE_-TIMEOUT** = 7,

  **SUPP_BE_SUCCESS** = 8 }
- enum { **EAPOL_CB_IN_PROGRESS** = 0, **EAPOL_CB_SUCCESS**, **EAPOL_CB_FAILURE** }

### Data Fields

- unsigned int **authWhile**
- unsigned int **heldWhile**
- unsigned int **startWhen**
- unsigned int **idleWhile**
- Boolean **eapFail**
- Boolean **eapolEap**
- Boolean **eapSuccess**
- Boolean **initialize**
- Boolean **keyDone**
- Boolean **keyRun**

- PortControl **portControl**
- Boolean **portEnabled**
- PortStatus **suppPortStatus**
- Boolean **portValid**
- Boolean **suppAbort**
- Boolean **suppFail**
- Boolean **suppStart**
- Boolean **suppSuccess**
- Boolean **suppTimeout**
- enum eapol_sm:: { ... } **SUPP_PAE_state**
- Boolean **userLogoff**
- Boolean **logoffSent**
- unsigned int **startCount**
- Boolean **eapRestart**
- PortControl **sPortMode**
- unsigned int **heldPeriod**
- unsigned int **startPeriod**
- unsigned int **maxStart**
- enum eapol_sm:: { ... } **KEY_RX_state**
- Boolean **rxKey**
- enum eapol_sm:: { ... } **SUPP_BE_state**
- Boolean **eapNoResp**
- Boolean **eapReq**
- Boolean **eapResp**
- unsigned int **authPeriod**
- unsigned int **dot1xSuppEapolFramesRx**
- unsigned int **dot1xSuppEapolFramesTx**
- unsigned int **dot1xSuppEapolStartFramesTx**
- unsigned int **dot1xSuppEapolLogoffFramesTx**
- unsigned int **dot1xSuppEapolRespFramesTx**
- unsigned int **dot1xSuppEapolReqIdFramesRx**
- unsigned int **dot1xSuppEapolReqFramesRx**
- unsigned int **dot1xSuppInvalidEapolFramesRx**
- unsigned int **dot1xSuppEapLengthErrorFramesRx**
- unsigned int **dot1xSuppLastEapolFrameVersion**
- unsigned char **dot1xSuppLastEapolFrameSource** [6]
- Boolean **changed**
- eap_sm * **eap**
- wpa_ssid * **config**
- Boolean **initial_req**
- u8 * **last_rx_key**
- size_t **last_rx_key_len**
- u8 * **eapReqData**
- size_t **eapReqDataLen**
- Boolean **altAccept**
- Boolean **altReject**
- Boolean **replay_counter_valid**
- u8 **last_replay_counter** [16]
- eapol_config **conf**
- eapol_ctx * **ctx**

- enum eapol_sm:: { ... } **cb_status**
- Boolean **cached_pmk**
- Boolean **unicast_key_received**
- Boolean **broadcast_key_received**

### 5.8.1 Detailed Description

Internal data for EAPOL state machines.

Definition at line 38 of file eapol_sm.c.

The documentation for this struct was generated from the following file:

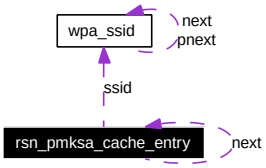
- eapol_sm.c

## 5.9 rsn_pmksa_cache_entry Struct Reference

PMKSA cache entry.

`#include <pmksa_cache.h>`

Collaboration diagram for rsn_pmksa_cache_entry:




### Data Fields

- rsn_pmksa_cache_entry * **next**
- u8 **pmkid** [PMKID_LEN]
- u8 **pmk** [PMK_LEN]
- size_t **pmk_len**
- os_time_t **expiration**
- int **akmp**
- u8 **aa** [ETH_ALEN]
- os_time_t **reauth_time**
- wpa_ssid * **ssid**
- int **opportunistic**

### 5.9.1 Detailed Description

PMKSA cache entry.

Definition at line 23 of file pmksa_cache.h.

The documentation for this struct was generated from the following file:

- pmksa_cache.h

## 5.10 tls_connection_params Struct Reference

Parameters for TLS connection.

`#include <tls.h>`

### Data Fields

- const char * **ca_cert**
- const u8 * **ca_cert_blob**
- size_t **ca_cert_blob_len**
- const char * **ca_path**
- const char * **subject_match**
- const char * **altsubject_match**
- const char * **client_cert**
- const u8 * **client_cert_blob**
- size_t **client_cert_blob_len**
- const char * **private_key**
- const u8 * **private_key_blob**
- size_t **private_key_blob_len**
- const char * **private_key_passwd**
- const char * **dh_file**
- const u8 * **dh_blob**
- size_t **dh_blob_len**
- int **tls_ia**
- int **engine**
- const char * **engine_id**
- const char * **pin**
- const char * **key_id**

### 5.10.1 Detailed Description

Parameters for TLS connection.

**Parameters:**

***ca_cert*** File or reference name for CA X.509 certificate in PEM or DER format

***ca_cert_blob*** ca_cert as inlined data or NULL if not used

***ca_cert_blob_len*** ca_cert_blob length

***ca_path*** Path to CA certificates (OpenSSL specific)

***subject_match*** String to match in the subject of the peer certificate or NULL to allow all subjects

***altsubject_match*** String to match in the alternative subject of the peer certificate or NULL to allow all alternative subjects

***client_cert*** File or reference name for client X.509 certificate in PEM or DER format

***client_cert_blob*** client_cert as inlined data or NULL if not used

***client_cert_blob_len*** client_cert_blob length

***private_key*** File or reference name for client private key in PEM or DER format (traditional format (RSA PRIVATE KEY) or PKCS#8 (PRIVATE KEY)

***private_key_blob*** private_key as inlined data or NULL if not used

***private_key_blob_len*** private_key_blob length

***private_key_passwd*** Passphrase for decrypted private key, NULL if no passphrase is used.

***dh_file*** File name for DH/DSA data in PEM format, or NULL if not used

***dh_blob*** dh_file as inlined data or NULL if not used

***dh_blob_len*** dh_blob length

***engine*** 1 = use engine (e.g., a smartcard) for private key operations (this is OpenSSL specific for now)

***engine_id*** engine id string (this is OpenSSL specific for now)

***ppin*** pointer to the pin variable in the configuration (this is OpenSSL specific for now)

***key_id*** the private key's key id (this is OpenSSL specific for now)

***tls_ia*** Whether to enable TLS/IA (for EAP-TTLSv1)

TLS connection parameters to be configured with tls_connection_set_params() and tls_global_set_-params().

Certificates and private key can be configured either as a reference name (file path or reference to certificate store) or by providing the same data as a pointer to the data in memory. Only one option will be used for each field.

Definition at line 79 of file tls.h.

The documentation for this struct was generated from the following file:

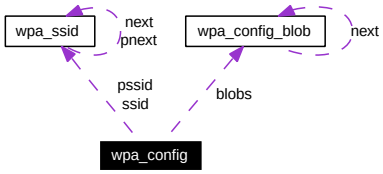
- tls.h

## 5.11 wpa_config Struct Reference

wpa_supplicant configuration data

`#include <config.h>`

Collaboration diagram for wpa_config:




### Data Fields

- wpa_ssid * ssid

  *Head of the global network list.*

- wpa_ssid ** pssid

  *Per-priority network lists (in priority order).*

- int num_prio

  *Number of different priorities used in the pssid lists.*

- int eapol_version

  *IEEE 802.1X/EAPOL version number.*

- int ap_scan

  *AP scanning/selection.*

- char * ctrl_interface

  *Parameters for the control interface.*

- char * ctrl_interface_group

  *Control interface group (DEPRECATED).*

- int fast_reauth

  *EAP fast re-authentication (session resumption).*

- char * opensc_engine_path

  *Path to the OpenSSL engine for opensc.*

- char * pkcs11_engine_path

  *Path to the OpenSSL engine for PKCS#11.*

- char * pkcs11_module_path

  *Path to the OpenSSL OpenSC/PKCS#11 module.*

- char * driver_param

  *Driver interface parameters.*

- unsigned int dot11RSNAConfigPMKLifetime

  *Maximum lifetime of a PMK.*

- unsigned int dot11RSNAConfigPMKReauthThreshold

  *PMK re-authentication threshold.*

- unsigned int dot11RSNAConfigSATimeout

  *Security association timeout.*

- int update_config

  *Is wpa_supplicant allowed to update configuration.*

- wpa_config_blob * blobs

  *Configuration blobs.*

## 5.11.1 Detailed Description

wpa_supplicant configuration data

This data structure is presents the per-interface (radio) configuration data. In many cases, there is only one struct wpa_config instance, but if more than one network interface is being controlled, one instance is used for each.

Definition at line 69 of file config.h.

## 5.11.2 Field Documentation

### 5.11.2.1 int wpa_config::ap_scan

AP scanning/selection.

By default, wpa_supplicant requests driver to perform AP scanning and then uses the scan results to select a suitable AP. Another alternative is to allow the driver to take care of AP scanning and selection and use wpa_supplicant just to process EAPOL frames based on IEEE 802.11 association information from the driver.

1: wpa_supplicant initiates scanning and AP selection (default).

0: Driver takes care of scanning, AP selection, and IEEE 802.11 association parameters (e.g., WPA IE generation); this mode can also be used with non-WPA drivers when using IEEE 802.1X mode; do not try to associate with APs (i.e., external program needs to control association). This mode must also be used when using wired Ethernet drivers.

2: like 0, but associate with APs using security policy and SSID (but not BSSID); this can be used, e.g., with ndiswrapper and NDIS drivers to enable operation with hidden SSIDs and optimized roaming; in this mode, the network blocks in the configuration are tried one by one until the driver reports successful association; each network block should have explicit security policy (i.e., only one option in the lists) for key_mgmt, pairwise, group, proto variables.

Definition at line 134 of file config.h.

#### 5.11.2.2 char* wpa_config::ctrl_interface

Parameters for the control interface.

If this is specified, wpa_supplicant will open a control interface that is available for external programs to manage wpa_supplicant. The meaning of this string depends on which control interface mechanism is used. For all cases, the existance of this parameter in configuration is used to determine whether the control interface is enabled.

For UNIX domain sockets (default on Linux and BSD): This is a directory that will be created for UNIX domain sockets for listening to requests from external programs (CLI/GUI, etc.) for status information and configuration. The socket file will be named based on the interface name, so multiple wpa_supplicant processes can be run at the same time if more than one interface is used. /var/run/wpa_supplicant is the recommended directory for sockets and by default, wpa_cli will use it when trying to connect with wpa_supplicant.

Access control for the control interface can be configured by setting the directory to allow only members of a group to use sockets. This way, it is possible to run wpa_supplicant as root (since it needs to change network configuration and open raw sockets) and still allow GUI/CLI components to be run as non-root users. However, since the control interface can be used to change the network configuration, this access needs to be protected in many cases. By default, wpa_supplicant is configured to use gid 0 (root). If you want to allow non-root users to use the control interface, add a new group and change this value to match with that group. Add users that should have control interface access to this group.

When configuring both the directory and group, use following format: DIR=/var/run/wpa_supplicant GROUP=wheel DIR=/var/run/wpa_supplicant GROUP=0 (group can be either group name or gid)

For UDP connections (default on Windows): The value will be ignored. This variable is just used to select that the control interface is to be created. The value can be set to, e.g., udp (ctrl_interface=udp).

For Windows Named Pipe: This value can be used to set the security descriptor for controlling access to the control interface. Security descriptor can be set using Security Descriptor String Format (see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthz/security/se descriptor_string_format.asp). The descriptor string needs to be prefixed with SDDL=. For example, ctrl_interface=SDDL=D: would set an empty DACL (which will reject all connections).

Definition at line 189 of file config.h.

#### 5.11.2.3 char* wpa_config::ctrl_interface_group

Control interface group (DEPRECATED).

This variable is only used for backwards compatibility. Group for UNIX domain sockets should now be specified using GROUP=<group> in ctrl_interface variable.

Definition at line 199 of file config.h.

#### 5.11.2.4 unsigned int wpa_config::dot11RSNAConfigPMKLifetime

Maximum lifetime of a PMK.

dot11 MIB variable for the maximum lifetime of a PMK in the PMK cache (unit: seconds).

Definition at line 258 of file config.h.

#### 5.11.2.5 unsigned int wpa_config::dot11RSNAConfigPMKReauthThreshold

PMK re-authentication threshold.

dot11 MIB variable for the percentage of the PMK lifetime that should expire before an IEEE 802.1X reauthentication occurs.

Definition at line 267 of file config.h.

#### 5.11.2.6 unsigned int wpa_config::dot11RSNAConfigSATimeout

Security association timeout.

dot11 MIB variable for the maximum time a security association shall take to set up (unit: seconds).

Definition at line 276 of file config.h.

#### 5.11.2.7 char* wpa_config::driver_param

Driver interface parameters.

This text string is passed to the selected driver interface with the optional struct wpa_driver_ops::set_-param() handler. This can be used to configure driver specific options without having to add new driver interface functionality.

Definition at line 249 of file config.h.

#### 5.11.2.8 int wpa_config::eapol_version

IEEE 802.1X/EAPOL version number.

wpa_supplicant is implemented based on IEEE Std 802.1X-2004 which defines EAPOL version 2. However, there are many APs that do not handle the new version number correctly (they seem to drop the frames completely). In order to make wpa_supplicant interoperate with these APs, the version number is set to 1 by default. This configuration value can be used to set it to the new version (2).

Definition at line 104 of file config.h.

#### 5.11.2.9 int wpa_config::fast_reauth

EAP fast re-authentication (session resumption).

By default, fast re-authentication is enabled for all EAP methods that support it. This variable can be used to disable fast re-authentication (by setting fast_reauth=0). Normally, there is no need to disable fast re-authentication.

Definition at line 210 of file config.h.

#### 5.11.2.10 int wpa_config::num_prio

Number of different priorities used in the pssid lists.

This indicates how many per-priority network lists are included in pssid.

Definition at line 91 of file config.h.

#### 5.11.2.11 char* wpa_config::opensc_engine_path

Path to the OpenSSL engine for opensc.

This is an OpenSSL specific configuration option for loading OpenSC engine (engine_opensc.so); if NULL, this engine is not loaded.

Definition at line 219 of file config.h.

#### 5.11.2.12 char* wpa_config::pkcs11_engine_path

Path to the OpenSSL engine for PKCS#11.

This is an OpenSSL specific configuration option for loading PKCS#11 engine (engine_pkcs11.so); if NULL, this engine is not loaded.

Definition at line 228 of file config.h.

#### 5.11.2.13 char* wpa_config::pkcs11_module_path

Path to the OpenSSL OpenSC/PKCS#11 module.

This is an OpenSSL specific configuration option for configuring path to OpenSC/PKCS#11 engine (opensc-pkcs11.so); if NULL, this module is not loaded.

Definition at line 238 of file config.h.

#### 5.11.2.14 struct wpa_ssid* wpa_config::ssid

Head of the global network list.

This is the head for the list of all the configured networks.

Definition at line 76 of file config.h.

#### 5.11.2.15 int wpa_config::update_config

Is wpa_supplicant allowed to update configuration.

This variable control whether wpa_supplicant is allow to re-write its configuration with wpa_config_write(). If this is zero, configuration data is only changed in memory and the external data is not overriden. If this is non-zero, wpa_supplicant will update the configuration data (e.g., a file) whenever configuration is changed. This update may replace the old configuration which can remove comments from it in case of a text file configuration.

Definition at line 290 of file config.h.

The documentation for this struct was generated from the following file:

- config.h

## 5.12 wpa_config_blob Struct Reference

Named configuration blob.

`#include <config.h>`

Collaboration diagram for wpa_config_blob:


### Data Fields

- char * name
  *Blob name.*
- u8 * data
  *Pointer to binary data.*
- size_t len
  *Length of binary data.*
- wpa_config_blob * next
  *Pointer to next blob in the configuration.*

### 5.12.1 Detailed Description

Named configuration blob.

This data structure is used to provide storage for binary objects to store abstract information like certificates and private keys inlined with the configuration data.

Definition at line 33 of file config.h.

The documentation for this struct was generated from the following file:

- config.h

## 5.13 wpa_ctrl Struct Reference

Internal structure for control interface library.

### 5.13.1 Detailed Description

Internal structure for control interface library.

This structure is used by the wpa_supplicant/hostapd control interface library to store internal data. Programs using the library should not touch this data directly. They can only use the pointer to the data structure as an identifier for the control interface connection and use this as one of the arguments for most of the control interface library functions.

Definition at line 43 of file wpa_ctrl.c.

The documentation for this struct was generated from the following file:

- wpa_ctrl.c

## 5.14 wpa_ctrl_dst Struct Reference

Internal data structure of control interface clients.

Collaboration diagram for wpa_ctrl_dst:


### Data Fields

- OVERLAPPED **overlap**
- wpa_ctrl_dst * **next**
- wpa_ctrl_dst * **prev**
- ctrl_iface_priv * **priv**
- HANDLE **pipe**
- int **attached**
- int **debug_level**
- int **errors**
- char **req_buf** [REQUEST_BUFSIZE]
- char * **rsp_buf**
- int **used**
- wpa_ctrl_dst * **next**
- sockaddr_in **addr**
- socklen_t **addrlen**
- wpa_ctrl_dst * **next**
- sockaddr_un **addr**

### 5.14.1 Detailed Description

Internal data structure of control interface clients.

This structure is used to store information about registered control interface monitors into struct wpa_supplicant. This data is private to ctrl_iface_named_pipe.c and should not be touched directly from other files.

Definition at line 68 of file ctrl_iface_named_pipe.c.

The documentation for this struct was generated from the following files:

- ctrl_iface_named_pipe.c
- ctrl_iface_udp.c
- ctrl_iface_unix.c

## 5.15 wpa_driver_associate_params Struct Reference

Association parameters.

`#include <driver.h>`

### Public Types

- enum { **NO_MGMT_FRAME_PROTECTION**, **MGMT_FRAME_PROTECTION_OPTIONAL**, **MGMT_FRAME_PROTECTION_REQUIRED** }

  *IEEE 802.11w management frame protection.*

### Data Fields

- const u8 * bssid

  *BSSID of the selected AP.*

- const u8 * ssid

  *The selected SSID.*

- size_t **ssid_len**
- int freq

  *Frequency of the channel the selected AP is using.*

- const u8 * wpa_ie

  *WPA information element for (Re)Association Request.*

- size_t wpa_ie_len

  *length of the wpa_ie*

- wpa_cipher **pairwise_suite**
- wpa_cipher **group_suite**
- wpa_key_mgmt **key_mgmt_suite**
- int auth_alg

  *Allowed authentication algorithms.*

- int mode

  *Operation mode (infra/ibss) IEEE80211_MODE_*.*

- const u8 * wep_key [4]

  *WEP keys for static WEP configuration.*

- size_t wep_key_len [4]

  *WEP key length for static WEP configuration.*

- int wep_tx_keyidx

  *WEP TX key index for static WEP configuration.*

- enum wpa_driver_associate_params:: { ... } mgmt_frame_protection

*IEEE 802.11w management frame protection.*

### 5.15.1 Detailed Description

Association parameters.

Data for struct wpa_driver_ops::associate().

Definition at line 77 of file driver.h.

### 5.15.2 Field Documentation

#### 5.15.2.1 int wpa_driver_associate_params::auth_alg

Allowed authentication algorithms.

Bit field of AUTH_ALG_*

Definition at line 134 of file driver.h.

#### 5.15.2.2 const u8* wpa_driver_associate_params::bssid

BSSID of the selected AP.

This can be NULL, if ap_scan=2 mode is used and the driver is responsible for selecting with which BSS to associate.

Definition at line 83 of file driver.h.

#### 5.15.2.3 int wpa_driver_associate_params::freq

Frequency of the channel the selected AP is using.

Frequency that the selected AP is using (in MHz as reported in the scan results)

Definition at line 98 of file driver.h.

#### 5.15.2.4 const u8* wpa_driver_associate_params::wpa_ie

WPA information element for (Re)Association Request.

WPA information element to be included in (Re)Association Request (including information element id and length). Use of this WPA IE is optional. If the driver generates the WPA IE, it can use pairwise_suite, group_suite, and key_mgmt_suite to select proper algorithms. In this case, the driver has to notify wpa_supplicant about the used WPA IE by generating an event that the interface code will convert into EVENT_ASSOCINFO data (see wpa_supplicant.h). When using WPA2/IEEE 802.11i, wpa_ie is used for RSN IE instead. The driver can determine which version is used by looking at the first byte of the IE (0xdd for WPA, 0x30 for WPA2/RSN).

Definition at line 116 of file driver.h.

The documentation for this struct was generated from the following file:

- driver.h

## 5.16 wpa_driver_capa Struct Reference

Driver capability information.

`#include <driver.h>`

### Data Fields

- unsigned int **key_mgmt**
- unsigned int **enc**
- unsigned int **auth**
- unsigned int **flags**

### 5.16.1 Detailed Description

Driver capability information.

Definition at line 175 of file driver.h.

The documentation for this struct was generated from the following file:

- driver.h

## 5.17 wpa_driver_ops Struct Reference

Driver interface API definition.

`#include <driver.h>`

### Data Fields

- const char * name
- const char * desc
- int(* get_bssid )(void *priv, u8 *bssid)

  *Get the current BSSID.*

- int(* get_ssid )(void *priv, u8 *ssid)

  *Get the current SSID.*

- int(* set_wpa )(void *priv, int enabled)

  *Enable/disable WPA support (OBSOLETE).*

- int(* set_key )(void *priv, wpa_alg alg, const u8 *addr, int key_idx, int set_tx, const u8 *seq, size_t seq_len, const u8 *key, size_t key_len)

  *Configure encryption key.*

- void *(* init )(void *ctx, const char *ifname)

  *Initialize driver interface.*

- void(* deinit )(void *priv)

  *Deinitialize driver interface.*

- int(* set_param )(void *priv, const char *param)

  *Set driver configuration parameters.*

- int(* set_countermeasures )(void *priv, int enabled)

  *Enable/disable TKIP countermeasures.*

- int(* set_drop_unencrypted )(void *priv, int enabled)

  *Enable/disable unencrypted frame filtering.*

- int(* scan )(void *priv, const u8 *ssid, size_t ssid_len)

  *Request the driver to initiate scan.*

- int(* get_scan_results )(void *priv, struct wpa_scan_result *results, size_t max_size)

  *Fetch the latest scan results.*

- int(* deauthenticate )(void *priv, const u8 *addr, int reason_code)

  *Request driver to deauthenticate.*

- int(* disassociate )(void *priv, const u8 *addr, int reason_code)

  *Request driver to disassociate.*

- int(* associate )(void *priv, struct wpa_driver_associate_params *params)
  *Request driver to associate.*
- int(* set_auth_alg )(void *priv, int auth_alg)
  *Set IEEE 802.11 authentication algorithm.*
- int(* add_pmkid )(void *priv, const u8 *bssid, const u8 *pmkid)
  *Add PMKSA cache entry to the driver.*
- int(* remove_pmkid )(void *priv, const u8 *bssid, const u8 *pmkid)
  *Remove PMKSA cache entry to the driver.*
- int(* flush_pmkid )(void *priv)
  *Flush PMKSA cache.*
- int(* get_capa )(void *priv, struct wpa_driver_capa *capa)
  *Flush PMKSA cache.*
- void(* poll )(void *priv)
  *Poll driver for association information.*
- const char *(* get_ifname )(void *priv)
  *Get interface name.*
- const u8 *(* get_mac_addr )(void *priv)
  *Get own MAC address.*
- int(* send_eapol )(void *priv, const u8 *dest, u16 proto, const u8 *data, size_t data_len)
  *Optional function for sending EAPOL packets.*
- int(* set_operstate )(void *priv, int state)
  *Sets device operating state to DORMANT or UP.*
- int(* mlme_setprotection )(void *priv, const u8 *addr, int protect_type, int key_type)
  *MLME-SETPROTECTION.request primitive.*
- wpa_hw_modes *(* get_hw_feature_data )(void *priv, u16 *num_modes, u16 *flags)
  *Get hardware support data (channels and rates).*
- int(* set_channel )(void *priv, wpa_hw_mode phymode, int chan, int freq)
  *Set channel.*
- int(* set_ssid )(void *priv, const u8 *ssid, size_t ssid_len)
  *Set SSID.*
- int(* set_bssid )(void *priv, const u8 *bssid)
  *Set BSSID.*
- int(* send_mlme )(void *priv, const u8 *data, size_t data_len)
  *Send management frame from MLME.*

- int(∗ mlme_add_sta )(void ∗priv, const u8 ∗addr, const u8 ∗supp_rates, size_t supp_rates_len)

  *Add a STA entry into the driver/netstack.*

- int(∗ mlme_remove_sta )(void ∗priv, const u8 ∗addr)

  *Remove a STA entry from the driver/netstack.*

## 5.17.1 Detailed Description

Driver interface API definition.

This structure defines the API that each driver interface needs to implement for core wpa_supplicant code. All driver specific functionality is captured in this wrapper.

Definition at line 255 of file driver.h.

## 5.17.2 Field Documentation

### 5.17.2.1 int(∗ wpa_driver_ops::add_pmkid)(void ∗priv, const u8 ∗bssid, const u8 ∗pmkid)

Add PMKSA cache entry to the driver.

**Parameters:**

***priv*** private driver interface data

***bssid*** BSSID for the PMKSA cache entry

***pmkid*** PMKID for the PMKSA cache entry

**Returns:**

0 on success, -1 on failure

This function is called when a new PMK is received, as a result of either normal authentication or RSN pre-authentication.

If the driver generates RSN IE, i.e., it does not use wpa_ie in associate(), add_pmkid() can be used to add new PMKSA cache entries in the driver. If the driver uses wpa_ie from wpa_supplicant, this driver_ops function does not need to be implemented. Likewise, if the driver does not support WPA, this function is not needed.

### 5.17.2.2 int(∗ wpa_driver_ops::associate)(void ∗priv, struct wpa_driver_associate_params ∗params)

Request driver to associate.

**Parameters:**

***priv*** private driver interface data

***params*** association parameters

**Returns:**

0 on success, -1 on failure

#### 5.17.2.3 int(* wpa_driver_ops::deauthenticate)(void *priv, const u8 *addr, int reason_code)

Request driver to deauthenticate.

**Parameters:**

***priv*** private driver interface data

***addr*** peer address (BSSID of the AP)

***reason_code*** 16-bit reason code to be sent in the deauthentication frame

**Returns:**

0 on success, -1 on failure

#### 5.17.2.4 void(* wpa_driver_ops::deinit)(void *priv)

Deinitialize driver interface.

**Parameters:**

***priv*** private driver interface data from init()

Shut down driver interface and processing of driver events. Free private data buffer if one was allocated in init() handler.

#### 5.17.2.5 const char* wpa_driver_ops::desc

One line description of the driver interface

Definition at line 259 of file driver.h.

#### 5.17.2.6 int(* wpa_driver_ops::disassociate)(void *priv, const u8 *addr, int reason_code)

Request driver to disassociate.

**Parameters:**

***priv*** private driver interface data

***addr*** peer address (BSSID of the AP)

***reason_code*** 16-bit reason code to be sent in the disassociation frame

**Returns:**

0 on success, -1 on failure

#### 5.17.2.7 int(* wpa_driver_ops::flush_pmkid)(void *priv)

Flush PMKSA cache.

**Parameters:**

***priv*** private driver interface data

**Returns:**

0 on success, -1 on failure

This function is called when the supplicant drops all PMKSA cache entries for any reason.

If the driver generates RSN IE, i.e., it does not use wpa_ie in associate(), remove_pmkid() can be used to synchronize PMKSA caches between the driver and wpa_supplicant. If the driver uses wpa_ie from wpa_supplicant, this driver_ops function does not need to be implemented. Likewise, if the driver does not support WPA, this function is not needed.

#### 5.17.2.8 int(* wpa_driver_ops::get_bssid)(void *priv, u8 *bssid)

Get the current BSSID.

**Parameters:**

***priv*** private driver interface data

***bssid*** buffer for BSSID (ETH_ALEN = 6 bytes)

**Returns:**

0 on success, -1 on failure

Query kernel driver for the current BSSID and copy it to bssid. Setting bssid to 00:00:00:00:00:00 is recommended if the STA is not associated.

#### 5.17.2.9 int(* wpa_driver_ops::get_capa)(void *priv, struct wpa_driver_capa *capa)

Flush PMKSA cache.

**Parameters:**

***priv*** private driver interface data

**Returns:**

0 on success, -1 on failure

Get driver/firmware/hardware capabilities.

#### 5.17.2.10 struct wpa_hw_modes*(* wpa_driver_ops::get_hw_feature_data)(void *priv, u16 *num_modes, u16 *flags)

Get hardware support data (channels and rates).

**Parameters:**

***priv*** Private driver interface data

***num_modes*** Variable for returning the number of returned modes flags: Variable for returning hardware feature flags

**Returns:**

Pointer to allocated hardware data on success or NULL on failure. Caller is responsible for freeing this.

This function is only needed for drivers that export MLME (management frame processing) to wpa_supplicant.

#### 5.17.2.11 const char*(* wpa_driver_ops::get_ifname)(void *priv)

Get interface name.

**Parameters:**
***priv*** private driver interface data

**Returns:**
Pointer to the interface name. This can differ from the interface name used in init() call.

This optional function can be used to allow the driver interface to replace the interface name with something else, e.g., based on an interface mapping from a more descriptive name.

#### 5.17.2.12 const u8*(* wpa_driver_ops::get_mac_addr)(void *priv)

Get own MAC address.

**Parameters:**
***priv*** private driver interface data

**Returns:**
Pointer to own MAC address or NULL on failure

This optional function can be used to get the own MAC address of the device from the driver interface code. This is only needed if the l2_packet implementation for the OS does not provide easy access to a MAC address.

#### 5.17.2.13 int(* wpa_driver_ops::get_scan_results)(void *priv, struct wpa_scan_result *results, size_t max_size)

Fetch the latest scan results.

**Parameters:**
***priv*** private driver interface data

***results*** pointer to buffer for scan results

***max_size*** maximum number of entries (buffer size)

**Returns:**
Number of scan result entries used on success, -1 on failure

If scan results include more than max_size BSSes, max_size will be returned and the remaining entries will not be included in the buffer.

#### 5.17.2.14 int(* wpa_driver_ops::get_ssid)(void *priv, u8 *ssid)

Get the current SSID.

**Parameters:**
***priv*** private driver interface data

***ssid*** buffer for SSID (at least 32 bytes)

**Returns:**
Length of the SSID on success, -1 on failure

Query kernel driver for the current SSID and copy it to ssid. Returning zero is recommended if the STA is not associated.

Note: SSID is an array of octets, i.e., it is not nul terminated and can, at least in theory, contain control characters (including nul) and as such, should be processed as binary data, not a printable string.

#### 5.17.2.15 void*(* wpa_driver_ops::init)(void *ctx, const char *ifname)

Initialize driver interface.

**Parameters:**
***ctx*** context to be used when calling wpa_supplicant functions, e.g., wpa_supplicant_event()

***ifname*** interface name, e.g., wlan0

**Returns:**
Pointer to private data, NULL on failure

Initialize driver interface, including event processing for kernel driver events (e.g., associated, scan results, Michael MIC failure). This function can allocate a private configuration data area for

**Parameters:**
***ctx*** file descriptor, interface name, etc. information that may be needed in future driver operations. If this is not used, non-NULL value will need to be returned because NULL is used to indicate failure. The returned value will be used as 'void *priv' data for all other driver_ops functions.

The main event loop (eloop.c) of wpa_supplicant can be used to register callback for read sockets (eloop_register_read_sock()).

See wpa_supplicant.h for more information about events and wpa_supplicant_event() function.

#### 5.17.2.16 int(* wpa_driver_ops::mlme_add_sta)(void *priv, const u8 *addr, const u8 *supp_rates, size_t supp_rates_len)

Add a STA entry into the driver/netstack.

**Parameters:**
***priv*** Private driver interface data

***addr*** MAC address of the STA (e.g., BSSID of the AP)

***supp_rates*** Supported rate set (from (Re)AssocResp); in IEEE 802.11 format (one octet per rate, 1 = 0.5 Mbps)

***supp_rates_len*** Number of entries in supp_rates

**Returns:**
0 on success, -1 on failure

This function is only needed for drivers that export MLME (management frame processing) to wpa_supplicant. When the MLME code completes association with an AP, this function is called to configure the driver/netstack with a STA entry for data frame processing (TX rate control, encryption/decryption).

#### 5.17.2.17 int(* wpa_driver_ops::mlme_remove_sta)(void *priv, const u8 *addr)

Remove a STA entry from the driver/netstack.

**Parameters:**

***priv*** Private driver interface data

***addr*** MAC address of the STA (e.g., BSSID of the AP)

**Returns:**

0 on success, -1 on failure

This function is only needed for drivers that export MLME (management frame processing) to wpa_supplicant.

#### 5.17.2.18 int(* wpa_driver_ops::mlme_setprotection)(void *priv, const u8 *addr, int protect_type, int key_type)

MLME-SETPROTECTION.request primitive.

**Parameters:**

***priv*** Private driver interface data

***addr*** Address of the station for which to set protection (may be NULL for group keys)

***protect_type*** MLME_SETPROTECTION_PROTECT_TYPE_*

***key_type*** MLME_SETPROTECTION_KEY_TYPE_*

**Returns:**

0 on success, -1 on failure

This is an optional function that can be used to set the driver to require protection for Tx and/or Rx frames. This uses the layer interface defined in IEEE 802.11i-2004 clause 10.3.22.1 (MLME-SETPROTECTION.request). Many drivers do not use explicit set protection operation; instead, they set protection implicitly based on configured keys.

#### 5.17.2.19 const char* wpa_driver_ops::name

Name of the driver interface

Definition at line 257 of file driver.h.

#### 5.17.2.20 void(* wpa_driver_ops::poll)(void *priv)

Poll driver for association information.

**Parameters:**

***priv*** private driver interface data

This is an option callback that can be used when the driver does not provide event mechanism for association events. This is called when receiving WPA EAPOL-Key messages that require association information. The driver interface is supposed to generate associnfo event before returning from this callback function. In addition, the driver interface should generate an association event after having sent out associnfo.

#### 5.17.2.21 int(* wpa_driver_ops::remove_pmkid)(void *priv, const u8 *bssid, const u8 *pmkid)

Remove PMKSA cache entry to the driver.

**Parameters:**

***priv*** private driver interface data

***bssid*** BSSID for the PMKSA cache entry

***pmkid*** PMKID for the PMKSA cache entry

**Returns:**

0 on success, -1 on failure

This function is called when the supplicant drops a PMKSA cache entry for any reason.

If the driver generates RSN IE, i.e., it does not use wpa_ie in associate(), remove_pmkid() can be used to synchronize PMKSA caches between the driver and wpa_supplicant. If the driver uses wpa_ie from wpa_supplicant, this driver_ops function does not need to be implemented. Likewise, if the driver does not support WPA, this function is not needed.

#### 5.17.2.22 int(* wpa_driver_ops::scan)(void *priv, const u8 *ssid, size_t ssid_len)

Request the driver to initiate scan.

**Parameters:**

***priv*** private driver interface data

***ssid*** specific SSID to scan for (ProbeReq) or NULL to scan for all SSIDs (either active scan with broadcast SSID or passive scan

***ssid_len*** length of the SSID

**Returns:**

0 on success, -1 on failure

Once the scan results are ready, the driver should report scan results event for wpa_supplicant which will eventually request the results with wpa_driver_get_scan_results().

#### 5.17.2.23 int(* wpa_driver_ops::send_eapol)(void *priv, const u8 *dest, u16 proto, const u8 *data, size_t data_len)

Optional function for sending EAPOL packets.

**Parameters:**

***priv*** private driver interface data

***dest*** Destination MAC address

***proto*** Ethertype

***data*** EAPOL packet starting with IEEE 802.1X header

***data_len*** Size of the EAPOL packet

**Returns:**

0 on success, -1 on failure

This optional function can be used to override l2_packet operations with driver specific functionality. If this function pointer is set, l2_packet module is not used at all and the driver interface code is responsible for receiving and sending all EAPOL packets. The received EAPOL packets are sent to core code by calling wpa_supplicant_rx_eapol(). The driver interface is required to implement get_mac_addr() handler if send_eapol() is used.

#### 5.17.2.24 int(* wpa_driver_ops::send_mlme)(void *priv, const u8 *data, size_t data_len)

Send management frame from MLME.

**Parameters:**

***priv*** Private driver interface data

***data*** IEEE 802.11 management frame with IEEE 802.11 header

***data_len*** Size of the management frame

**Returns:**

0 on success, -1 on failure

This function is only needed for drivers that export MLME (management frame processing) to wpa_supplicant.

#### 5.17.2.25 int(* wpa_driver_ops::set_auth_alg)(void *priv, int auth_alg)

Set IEEE 802.11 authentication algorithm.

**Parameters:**

***priv*** private driver interface data

***auth_alg*** bit field of AUTH_ALG_*

If the driver supports more than one authentication algorithm at the same time, it should configure all supported algorithms. If not, one algorithm needs to be selected arbitrarily. Open System authentication should be ok for most cases and it is recommended to be used if other options are not supported. Static WEP configuration may also use Shared Key authentication and LEAP requires its own algorithm number. For LEAP, user can make sure that only one algorithm is used at a time by configuring LEAP as the only supported EAP method. This information is also available in associate() params, so set_auth_alg may not be needed in case of most drivers.

**Returns:**

0 on success, -1 on failure

#### 5.17.2.26 int(* wpa_driver_ops::set_bssid)(void *priv, const u8 *bssid)

Set BSSID.

**Parameters:**

***priv*** Private driver interface data

***bssid*** BSSID

**Returns:**

0 on success, -1 on failure

This function is only needed for drivers that export MLME (management frame processing) to wpa_supplicant.

#### 5.17.2.27 int(∗ wpa_driver_ops::set_channel)(void ∗priv, wpa_hw_mode phymode, int chan, int freq)

Set channel.

**Parameters:**

***priv*** Private driver interface data

***phymode*** WPA_MODE_IEEE80211B, ..

***chan*** IEEE 802.11 channel number

***freq*** Frequency of the channel in MHz

**Returns:**

0 on success, -1 on failure

This function is only needed for drivers that export MLME (management frame processing) to wpa_supplicant.

#### 5.17.2.28 int(∗ wpa_driver_ops::set_countermeasures)(void ∗priv, int enabled)

Enable/disable TKIP countermeasures.

**Parameters:**

***priv*** private driver interface data

***enabled*** 1 = countermeasures enabled, 0 = disabled

**Returns:**

0 on success, -1 on failure

Configure TKIP countermeasures. When these are enabled, the driver should drop all received and queued frames that are using TKIP.

#### 5.17.2.29 int(∗ wpa_driver_ops::set_drop_unencrypted)(void ∗priv, int enabled)

Enable/disable unencrypted frame filtering.

**Parameters:**

***priv*** private driver interface data

***enabled*** 1 = unencrypted Tx/Rx frames will be dropped, 0 = disabled

**Returns:**

0 on success, -1 on failure

Configure the driver to drop all non-EAPOL frames (both receive and transmit paths). Unencrypted EAPOL frames (ethertype 0x888e) must still be allowed for key negotiation.

#### 5.17.2.30 int(∗ wpa_driver_ops::set_key)(void ∗priv, wpa_alg alg, const u8 ∗addr, int key_idx, int set_tx, const u8 ∗seq, size_t seq_len, const u8 ∗key, size_t key_len)

Configure encryption key.

**Parameters:**

***priv*** private driver interface data

***alg*** encryption algorithm (WPA_ALG_NONE, WPA_ALG_WEP, WPA_ALG_TKIP, WPA_ALG_-CCMP, WPA_ALG_IGTK, WPA_ALG_DHV); WPA_ALG_NONE clears the key.

***addr*** address of the peer STA or ff:ff:ff:ff:ff:ff for broadcast/default keys

***key_idx*** key index (0..3), usually 0 for unicast keys; 0..4095 for IGTK

***set_tx*** configure this key as the default Tx key (only used when driver does not support separate unicast/individual key

***seq*** sequence number/packet number, seq_len octets, the next packet number to be used for in replay protection; configured for Rx keys (in most cases, this is only used with broadcast keys and set to zero for unicast keys)

***seq_len*** length of the seq, depends on the algorithm: TKIP: 6 octets, CCMP: 6 octets, IGTK: 6 octets

***key*** key buffer; TKIP: 16-byte temporal key, 8-byte Tx Mic key, 8-byte Rx Mic Key

***key_len*** length of the key buffer in octets (WEP: 5 or 13, TKIP: 32, CCMP: 16, IGTK: 16, DHV: 16)

**Returns:**

0 on success, -1 on failure

Configure the given key for the kernel driver. If the driver supports separate individual keys (4 default keys + 1 individual), addr can be used to determine whether the key is default or individual. If only 4 keys are supported, the default key with key index 0 is used as the individual key. STA must be configured to use it as the default Tx key (set_tx is set) and accept Rx for all the key indexes. In most cases, WPA uses only key indexes 1 and 2 for broadcast keys, so key index 0 is available for this kind of configuration.

Please note that TKIP keys include separate TX and RX MIC keys and some drivers may expect them in different order than wpa_supplicant is using. If the TX/RX keys are swapped, all TKIP encrypted packets will tricker Michael MIC errors. This can be fixed by changing the order of MIC keys by swapping te bytes 16..23 and 24..31 of the key in driver_∗.c set_key() implementation, see driver_ndis.c for an example on how this can be done.

#### 5.17.2.31 int(∗ wpa_driver_ops::set_operstate)(void ∗priv, int state)

Sets device operating state to DORMANT or UP.

**Parameters:**

***priv*** private driver interface data

***state*** 0 = dormant, 1 = up

**Returns:**

0 on success, -1 on failure

This is an optional function that can be used on operating systems that support a concept of controlling network device state from user space applications. This function, if set, gets called with state = 1 when authentication has been completed and with state = 0 when connection is lost.

#### 5.17.2.32 int(∗ wpa_driver_ops::set_param)(void ∗priv, const char ∗param)

Set driver configuration parameters.

**Parameters:**

***priv*** private driver interface data from init()

***param*** driver specific configuration parameters

**Returns:**

0 on success, -1 on failure

Optional handler for notifying driver interface about configuration parameters (driver_param).

#### 5.17.2.33 int(∗ wpa_driver_ops::set_ssid)(void ∗priv, const u8 ∗ssid, size_t ssid_len)

Set SSID.

**Parameters:**

***priv*** Private driver interface data

***ssid*** SSID

***ssid_len*** SSID length

**Returns:**

0 on success, -1 on failure

This function is only needed for drivers that export MLME (management frame processing) to wpa_supplicant.

#### 5.17.2.34 int(∗ wpa_driver_ops::set_wpa)(void ∗priv, int enabled)

Enable/disable WPA support (OBSOLETE).

**Parameters:**

***priv*** private driver interface data

***enabled*** 1 = enable, 0 = disable

**Returns:**

0 on success, -1 on failure

Note: This function is included for backwards compatibility. This is called only just after init and just before deinit, so these functions can be used to implement same functionality and the driver interface need not define this function.

Configure the kernel driver to enable/disable WPA support. This may be empty function, if WPA support is always enabled. Common configuration items are WPA IE (clearing it when WPA support is disabled), Privacy flag configuration for capability field (note: this the value need to set in associate handler to allow plaintext mode to be used) when trying to associate with, roaming mode (can allow wpa_supplicant to control roaming if ap_scan=1 is used; however, drivers can also implement roaming if desired, especially ap_scan=2 mode is used for this).

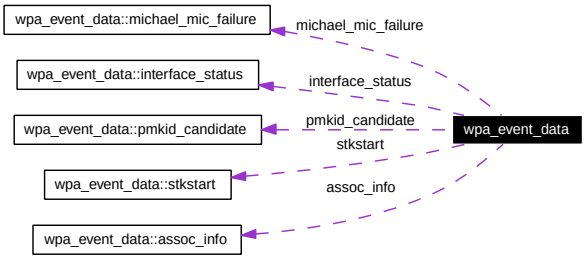
The documentation for this struct was generated from the following file:

- driver.h

# 5.18 wpa_event_data Union Reference

`#include <wpa_supplicant.h>`

Collaboration diagram for wpa_event_data:




## Data Fields

- wpa_event_data::assoc_info assoc_info
  *Data for EVENT_ASSOC and EVENT_ASSOCINFO events.*
- wpa_event_data::michael_mic_failure michael_mic_failure
  *Data for EVENT_MICHAEL_MIC_FAILURE.*
- wpa_event_data::interface_status interface_status
  *Data for EVENT_INTERFACE_STATUS.*
- wpa_event_data::pmkid_candidate pmkid_candidate
  *Data for EVENT_PMKID_CANDIDATE.*
- wpa_event_data::stkstart stkstart
  *Data for EVENT_STKSTART.*

## Data Structures

- struct assoc_info
  *Data for EVENT_ASSOC and EVENT_ASSOCINFO events.*
- struct interface_status
  *Data for EVENT_INTERFACE_STATUS.*
- struct michael_mic_failure
  *Data for EVENT_MICHAEL_MIC_FAILURE.*
- struct pmkid_candidate
  *Data for EVENT_PMKID_CANDIDATE.*
- struct stkstart
  *Data for EVENT_STKSTART.*

### 5.18.1 Detailed Description

union wpa_event_data - Additional data for wpa_supplicant_event() calls

Definition at line 151 of file wpa_supplicant.h.

### 5.18.2 Field Documentation

#### 5.18.2.1 struct wpa_event_data::assoc_info wpa_event_data::assoc_info

Data for EVENT_ASSOC and EVENT_ASSOCINFO events.

This structure is optional for EVENT_ASSOC calls and required for EVENT_ASSOCINFO calls. By using EVENT_ASSOC with this data, the driver interface does not need to generate separate EVENT_-ASSOCINFO calls.

The documentation for this union was generated from the following file:

- wpa_supplicant.h

## 5.19 wpa_event_data::assoc_info Struct Reference

Data for EVENT_ASSOC and EVENT_ASSOCINFO events.

`#include <wpa_supplicant.h>`

### Data Fields

- u8 * req_ies

  *(Re)Association Request IEs*

- size_t req_ies_len

  *Length of req_ies in bytes.*

- u8 * resp_ies

  *(Re)Association Response IEs*

- size_t resp_ies_len

  *Length of resp_ies in bytes.*

- u8 * beacon_ies

  *Beacon or Probe Response IEs.*

- size_t beacon_ies_len

  *Length of beacon_ies.*

### 5.19.1 Detailed Description

Data for EVENT_ASSOC and EVENT_ASSOCINFO events.

This structure is optional for EVENT_ASSOC calls and required for EVENT_ASSOCINFO calls. By using EVENT_ASSOC with this data, the driver interface does not need to generate separate EVENT_-ASSOCINFO calls.

Definition at line 161 of file wpa_supplicant.h.

### 5.19.2 Field Documentation

#### 5.19.2.1 u8* wpa_event_data::assoc_info::beacon_ies

Beacon or Probe Response IEs.

Optional Beacon/ProbeResp data: IEs included in Beacon or Probe Response frames from the current AP (i.e., the one that the client just associated with). This information is used to update WPA/RSN IE for the AP. If this field is not set, the results from previous scan will be used. If no data for the new AP is found, scan results will be requested again (without scan request). At this point, the driver is expected to provide WPA/RSN IE for the AP (if WPA/WPA2 is used).

This should start with the first IE (fixed fields before IEs are not included).

Definition at line 219 of file wpa_supplicant.h.

#### 5.19.2.2 u8* wpa_event_data::assoc_info::req_ies

(Re)Association Request IEs

If the driver generates WPA/RSN IE, this event data must be returned for WPA handshake to have needed information. If wpa_supplicant-generated WPA/RSN IE is used, this information event is optional.

This should start with the first IE (fixed fields before IEs are not included).

Definition at line 174 of file wpa_supplicant.h.

#### 5.19.2.3 u8* wpa_event_data::assoc_info::resp_ies

(Re)Association Response IEs

Optional association data from the driver. This data is not required WPA, but may be useful for some protocols and as such, should be reported if this is available to the driver interface.

This should start with the first IE (fixed fields before IEs are not included).

Definition at line 194 of file wpa_supplicant.h.

The documentation for this struct was generated from the following file:

- wpa_supplicant.h

## 5.20 wpa_event_data::interface_status Struct Reference

Data for EVENT_INTERFACE_STATUS.

`#include <wpa_supplicant.h>`

### Public Types

- enum { **EVENT_INTERFACE_ADDED**, **EVENT_INTERFACE_REMOVED** }

### Data Fields

- char **ifname** [100]
- enum wpa_event_data::interface_status:: { ... } **ievent**

### 5.20.1 Detailed Description

Data for EVENT_INTERFACE_STATUS.

Definition at line 239 of file wpa_supplicant.h.

The documentation for this struct was generated from the following file:

- wpa_supplicant.h

## 5.21 wpa_event_data::michael_mic_failure Struct Reference

Data for EVENT_MICHAEL_MIC_FAILURE.

`#include <wpa_supplicant.h>`

### Data Fields

- int **unicast**

### 5.21.1 Detailed Description

Data for EVENT_MICHAEL_MIC_FAILURE.

Definition at line 231 of file wpa_supplicant.h.

The documentation for this struct was generated from the following file:

- wpa_supplicant.h

## 5.22 wpa_event_data::pmkid_candidate Struct Reference

Data for EVENT_PMKID_CANDIDATE.

`#include <wpa_supplicant.h>`

### Data Fields

- u8 bssid [ETH_ALEN]
- int index
- int preauth

### 5.22.1 Detailed Description

Data for EVENT_PMKID_CANDIDATE.

Definition at line 250 of file wpa_supplicant.h.

### 5.22.2 Field Documentation

#### 5.22.2.1 u8 wpa_event_data::pmkid_candidate::bssid[ETH_ALEN]

BSSID of the PMKID candidate

Definition at line 252 of file wpa_supplicant.h.

#### 5.22.2.2 int wpa_event_data::pmkid_candidate::index

Smaller the index, higher the priority

Definition at line 254 of file wpa_supplicant.h.

#### 5.22.2.3 int wpa_event_data::pmkid_candidate::preauth

Whether RSN IE includes pre-authenticate flag

Definition at line 256 of file wpa_supplicant.h.

The documentation for this struct was generated from the following file:

- wpa_supplicant.h

## 5.23 wpa_event_data::stkstart Struct Reference

Data for EVENT_STKSTART.

`#include <wpa_supplicant.h>`

### Data Fields

- u8 **peer** [ETH_ALEN]

### 5.23.1 Detailed Description

Data for EVENT_STKSTART.

Definition at line 263 of file wpa_supplicant.h.

The documentation for this struct was generated from the following file:

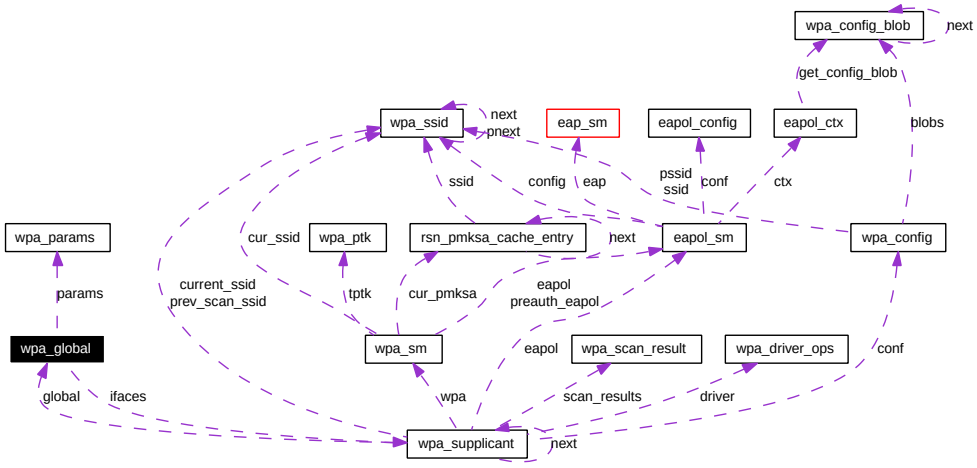
- wpa_supplicant.h

## 5.24 wpa_global Struct Reference

Internal, global data for all wpa_supplicant interfaces.

`#include <wpa_supplicant_i.h>`

Collaboration diagram for wpa_global:




### Data Fields

- wpa_supplicant * **ifaces**
- wpa_params **params**
- ctrl_iface_global_priv * **ctrl_iface**
- ctrl_iface_dbus_priv * **dbus_ctrl_iface**

### 5.24.1 Detailed Description

Internal, global data for all wpa_supplicant interfaces.

This structure is initialized by calling wpa_supplicant_init() when starting wpa_supplicant.

Definition at line 192 of file wpa_supplicant_i.h.

The documentation for this struct was generated from the following file:

- wpa_supplicant_i.h

## 5.25 wpa_interface Struct Reference

Parameters for wpa_supplicant_add_iface().

`#include <wpa_supplicant_i.h>`

### Data Fields

- const char * confname
  *Configuration name (file or profile) name.*
- const char * ctrl_interface
  *Control interface parameter.*
- const char * driver
  *Driver interface name, or NULL to use the default driver.*
- const char * driver_param
  *Driver interface parameters.*
- const char * ifname
  *Interface name.*
- const char * bridge_ifname
  *Optional bridge interface name.*

### 5.25.1 Detailed Description

Parameters for wpa_supplicant_add_iface().

Definition at line 45 of file wpa_supplicant_i.h.

### 5.25.2 Field Documentation

#### 5.25.2.1 const char* wpa_interface::bridge_ifname

Optional bridge interface name.

If the driver interface (ifname) is included in a Linux bridge device, the bridge interface may need to be used for receiving EAPOL frames. This can be enabled by setting this variable to enable receiving of EAPOL frames from an additional interface.

Definition at line 101 of file wpa_supplicant_i.h.

#### 5.25.2.2 const char* wpa_interface::confname

Configuration name (file or profile) name.

This can also be NULL when a configuration file is not used. In that case, ctrl_interface must be set to allow the interface to be configured.

Definition at line 54 of file wpa_supplicant_i.h.

#### 5.25.2.3 const char* wpa_interface::ctrl_interface

Control interface parameter.

If a configuration file is not used, this variable can be used to set the ctrl_interface parameter that would have otherwise been read from the configuration file. If both confname and ctrl_interface are set, ctrl_interface is used to override the value from configuration file.

Definition at line 66 of file wpa_supplicant_i.h.

#### 5.25.2.4 const char* wpa_interface::driver_param

Driver interface parameters.

If a configuration file is not used, this variable can be used to set the driver_param parameters that would have otherwise been read from the configuration file. If both confname and driver_param are set, driver_param is used to override the value from configuration file.

Definition at line 84 of file wpa_supplicant_i.h.

The documentation for this struct was generated from the following file:

- wpa_supplicant_i.h

## 5.26 wpa_params Struct Reference

Parameters for wpa_supplicant_init().

`#include <wpa_supplicant_i.h>`

### Data Fields

- int daemonize
  *Run wpa_supplicant in the background.*
- int wait_for_interface
  *Wait for the network interface to appear.*
- int wait_for_monitor
  *Wait for a monitor program before starting.*
- char * pid_file
  *Path to a PID (process ID) file.*
- int wpa_debug_level
  *Debugging verbosity level (e.g., MSG_INFO).*
- int wpa_debug_show_keys
  *Whether keying material is included in debug.*
- int wpa_debug_timestamp
  *Whether to include timestamp in debug messages.*
- char * ctrl_interface
  *Global ctrl_iface path/parameter.*
- int dbus_ctrl_interface
  *Enable the DBus control interface.*
- int wpa_debug_use_file
  *Write debug to a file (instead of stdout).*

### 5.26.1 Detailed Description

Parameters for wpa_supplicant_init().

Definition at line 108 of file wpa_supplicant_i.h.

### 5.26.2 Field Documentation

#### 5.26.2.1 char* wpa_params::pid_file

Path to a PID (process ID) file.

If this and daemonize are set, process ID of the background process will be written to the specified file.

Definition at line 140 of file wpa_supplicant_i.h.

#### 5.26.2.2 int wpa_params::wait_for_interface

Wait for the network interface to appear.

If set, wpa_supplicant will wait until all the configured network interfaces are available before starting processing. Please note that in many cases, a better alternative would be to start wpa_supplicant without network interfaces and add the interfaces dynamically whenever they become available.

Definition at line 125 of file wpa_supplicant_i.h.

#### 5.26.2.3 int wpa_params::wpa_debug_show_keys

Whether keying material is included in debug.

This parameter can be used to allow keying material to be included in debug messages. This is a security risk and this option should not be enabled in normal configuration. If needed during development or while troubleshooting, this option can provide more details for figuring out what is happening.

Definition at line 158 of file wpa_supplicant_i.h.

The documentation for this struct was generated from the following file:

- wpa_supplicant_i.h

## 5.27 wpa_ptk Struct Reference

WPA Pairwise Transient Key.

`#include <wpa_i.h>`

### Data Fields

- u8 **kck** [16]
- u8 **kek** [16]
- u8 **tk1** [16]
- union {
  u8 **tk2** [16]
  struct {
  u8 **tx_mic_key** [8]
  u8 **rx_mic_key** [8]
  } **auth**
  } **u**

### 5.27.1 Detailed Description

WPA Pairwise Transient Key.

IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy

Definition at line 30 of file wpa_i.h.

The documentation for this struct was generated from the following file:

- wpa_i.h

## 5.28 wpa_scan_result Struct Reference

Scan results.

`#include <driver.h>`

### Data Fields

- u8 **bssid** [ETH_ALEN]
- u8 **ssid** [32]
- size_t **ssid_len**
- u8 **wpa_ie** [SSID_MAX_WPA_IE_LEN]
- size_t **wpa_ie_len**
- u8 **rsn_ie** [SSID_MAX_WPA_IE_LEN]
- size_t **rsn_ie_len**
- int **freq**
- u16 **caps**
- int **qual**
- int **noise**
- int **level**
- int **maxrate**

### 5.28.1 Detailed Description

Scan results.

**Parameters:**

***bssid*** BSSID

***ssid*** SSID

***ssid_len*** length of the ssid

***wpa_ie*** WPA IE

***wpa_ie_len*** length of the wpa_ie

***rsn_ie*** RSN IE

***rsn_ie_len*** length of the RSN IE

***freq*** frequency of the channel in MHz (e.g., 2412 = channel 1)

***caps*** capability information field in host byte order

***qual*** signal quality

***noise*** noise level

***level*** signal level

***maxrate*** maximum supported rate

This structure is used as a generic format for scan results from the driver. Each driver interface implementation is responsible for converting the driver or OS specific scan results into this format.

Definition at line 56 of file driver.h.

The documentation for this struct was generated from the following file:

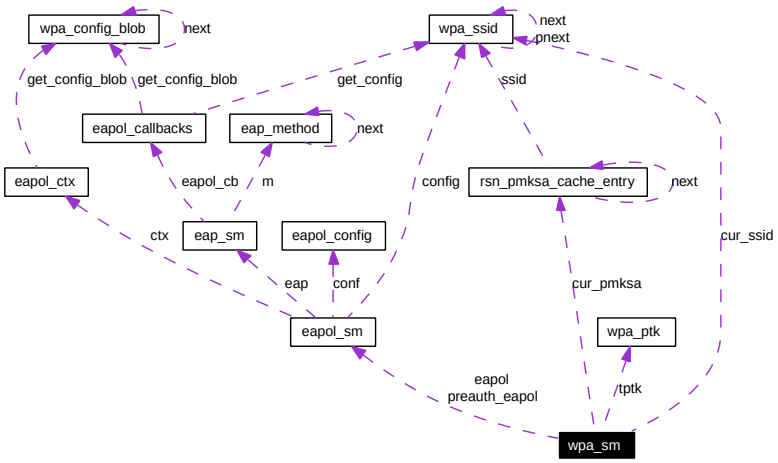
- driver.h

## 5.29 wpa_sm Struct Reference

Internal WPA state machine data.

`#include <wpa_i.h>`

Collaboration diagram for wpa_sm:




### Data Fields

- u8 **pmk** [PMK_LEN]
- size_t **pmk_len**
- wpa_ptk ptk **tptk**
- int **ptk_set**
- int **tptk_set**
- u8 **snonce** [WPA_NONCE_LEN]
- u8 **anonce** [WPA_NONCE_LEN]
- int **renew_snonce**
- u8 **rx_replay_counter** [WPA_REPLAY_COUNTER_LEN]
- int **rx_replay_counter_set**
- u8 **request_counter** [WPA_REPLAY_COUNTER_LEN]
- eapol_sm * **eapol**
- rsn_pmksa_cache * **pmksa**
- rsn_pmksa_cache_entry * **cur_pmksa**
- rsn_pmksa_candidate * **pmksa_candidates**
- l2_packet_data * **l2_preauth**
- l2_packet_data * **l2_preauth_br**
- u8 **preauth_bssid** [ETH_ALEN]
- eapol_sm * **preauth_eapol**
- wpa_sm_ctx * **ctx**
- void * **scard_ctx**
- int **fast_reauth**
- wpa_ssid * **cur_ssid**
- u8 **own_addr** [ETH_ALEN]
- const char * **ifname**

- const char * **bridge_ifname**
- u8 **bssid** [ETH_ALEN]
- unsigned int **dot11RSNAConfigPMKLifetime**
- unsigned int **dot11RSNAConfigPMKReauthThreshold**
- unsigned int **dot11RSNAConfigSATimeout**
- unsigned int **dot11RSNA4WayHandshakeFailures**
- unsigned int **proto**
- unsigned int **pairwise_cipher**
- unsigned int **group_cipher**
- unsigned int **key_mgmt**
- unsigned int **mgmt_group_cipher**
- u8 * **assoc_wpa_ie**
- size_t **assoc_wpa_ie_len**
- u8 * **ap_wpa_ie**
- u8 * **ap_rsn_ie**
- size_t **ap_wpa_ie_len**
- size_t **ap_rsn_ie_len**

### 5.29.1 Detailed Description

Internal WPA state machine data.

Definition at line 81 of file wpa_i.h.

The documentation for this struct was generated from the following file:

- wpa_i.h

## 5.30 wpa_ssid Struct Reference

Network configuration data.

`#include <config_ssid.h>`

Collaboration diagram for wpa_ssid:


### Data Fields

- wpa_ssid * next
  *Next network in global list.*
- wpa_ssid * pnext
  *Next network in per-priority list.*
- int id
  *Unique id for the network.*
- int priority
  *Priority group.*
- u8 * ssid
  *Service set identifier (network name).*
- size_t ssid_len
  *Length of the SSID.*
- u8 bssid [ETH_ALEN]
  *BSSID.*
- int bssid_set
  *Whether BSSID is configured for this network.*
- u8 psk [PMK_LEN]
  *WPA pre-shared key (256 bits).*
- int psk_set
  *Whether PSK field is configured.*
- char * passphrase
  *WPA ASCII passphrase.*
- int pairwise_cipher
  *Bitfield of allowed pairwise ciphers, WPA_CIPHER_*.*
- int group_cipher

*Bitfield of allowed group ciphers, WPA_CIPHER_∗.*

- int key_mgmt

  *Bitfield of allowed key management protocols.*

- int proto

  *Bitfield of allowed protocols, WPA_PROTO_∗.*

- int auth_alg

  *Bitfield of allowed authentication algorithms.*

- int scan_ssid

  *Scan this SSID with Probe Requests.*

- u8 ∗ identity

  *EAP Identity.*

- size_t identity_len

  *EAP Identity length.*

- u8 ∗ anonymous_identity

  *Anonymous EAP Identity.*

- size_t anonymous_identity_len

  *Length of anonymous_identity.*

- u8 ∗ eappsk

  *EAP-PSK/PAX/SAKE pre-shared key.*

- size_t eappsk_len

  *EAP-PSK/PAX/SAKE pre-shared key length.*

- u8 ∗ nai

  *User NAI (for EAP-PSK/PAX/SAKE).*

- size_t nai_len

  *Length of nai field.*

- u8 ∗ password

  *Password string for EAP.*

- size_t password_len

  *Length of password field.*

- u8 ∗ ca_cert

  *File path to CA certificate file (PEM/DER).*

- u8 ∗ ca_path

  *Directory path for CA certificate files (PEM).*

- u8 * client_cert
  *File path to client certificate file (PEM/DER).*
- u8 * private_key
  *File path to client private key file (PEM/DER/PFX).*
- u8 * private_key_passwd
  *Password for private key file.*
- u8 * dh_file
  *File path to DH/DSA parameters file (in PEM format).*
- u8 * subject_match
  *Constraint for server certificate subject.*
- u8 * **altsubject_match**
- u8 * ca_cert2
  *File path to CA certificate file (PEM/DER) (Phase 2).*
- u8 * ca_path2
  *Directory path for CA certificate files (PEM) (Phase 2).*
- u8 * client_cert2
  *File path to client certificate file.*
- u8 * private_key2
  *File path to client private key file.*
- u8 * private_key2_passwd
  *Password for private key file.*
- u8 * dh_file2
  *File path to DH/DSA parameters file (in PEM format).*
- u8 * subject_match2
  *Constraint for server certificate subject.*
- u8 * altsubject_match2
  *Constraint for server certificate alt. subject.*
- eap_method_type * eap_methods
  *Allowed EAP methods.*
- char * phase1
  *Phase 1 (outer authentication) parameters.*
- char * phase2
  *Phase2 (inner authentication with TLS tunnel) parameters.*
- char * pcsc

*Parameters for PC/SC smartcard interface for USIM and GSM SIM.*

- char * pin
  *PIN for USIM, GSM SIM, and smartcards.*
- int engine
  *Enable OpenSSL engine (e.g., for smartcard access).*
- char * engine_id
  *Engine ID for OpenSSL engine.*
- char * key_id
  *Key ID for OpenSSL engine.*
- int eapol_flags
  *Bit field of IEEE 802.1X/EAPOL options (EAPOL_FLAG_*).*
- u8 wep_key [NUM_WEP_KEYS][MAX_WEP_KEY_LEN]
  *WEP keys.*
- size_t wep_key_len [NUM_WEP_KEYS]
  *WEP key lengths.*
- int wep_tx_keyidx
  *Default key index for TX frames using WEP.*
- int proactive_key_caching
  *Enable proactive key caching.*
- u8 * otp
  *One-time-password.*
- size_t otp_len
  *Length of the otp field.*
- int pending_req_identity
  *Whether there is a pending identity request.*
- int pending_req_password
  *Whether there is a pending password request.*
- int pending_req_pin
  *Whether there is a pending PIN request.*
- int pending_req_new_password
  *Pending password update request.*
- int pending_req_passphrase
  *Pending passphrase request.*

- char * pending_req_otp
  *Whether there is a pending OTP request.*
- size_t pending_req_otp_len
  *Length of the pending OTP request.*
- int leap
  *Number of EAP methods using LEAP.*
- int non_leap
  *Number of EAP methods not using LEAP.*
- unsigned int eap_workaround
  *EAP workarounds enabled.*
- char * pac_file
  *File path or blob name for the PAC entries (EAP-FAST).*
- int mode
  *IEEE 802.11 operation mode (Infrastucture/IBSS).*
- int mschapv2_retry
  *MSCHAPv2 retry in progress.*
- u8 * new_password
  *New password for password update.*
- size_t new_password_len
  *Length of new_password field.*
- int disabled
  *Whether this network is currently disabled.*
- int peerkey
  *Whether PeerKey handshake for direct links is allowed.*
- int fragment_size
  *Maximum EAP fragment size in bytes (default 1398).*
- char * id_str
  *Network identifier string for external scripts.*

### 5.30.1 Detailed Description

Network configuration data.

This structure includes all the configuration variables for a network. This data is included in the per-interface configuration data as an element of the network list, struct wpa_config::ssid. Each network block in the configuration is mapped to a struct wpa_ssid instance.

Definition at line 70 of file config_ssid.h.

## 5.30.2 Field Documentation

### 5.30.2.1 u8* wpa_ssid::altsubject_match2

Constraint for server certificate alt. subject.

This field is like altsubject_match, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication.

Definition at line 529 of file config_ssid.h.

### 5.30.2.2 u8* wpa_ssid::anonymous_identity

Anonymous EAP Identity.

This field is used for unencrypted use with EAP types that support different tunnelled identity, e.g., EAP-TTLS, in order to reveal the real identity (identity field) only to the authentication server.

Definition at line 244 of file config_ssid.h.

### 5.30.2.3 int wpa_ssid::auth_alg

Bitfield of allowed authentication algorithms.

WPA_AUTH_ALG_*

Definition at line 210 of file config_ssid.h.

### 5.30.2.4 u8 wpa_ssid::bssid[ETH_ALEN]

BSSID.

If set, this network block is used only when associating with the AP using the configured BSSID

Definition at line 148 of file config_ssid.h.

### 5.30.2.5 u8* wpa_ssid::ca_cert

File path to CA certificate file (PEM/DER).

This file can have one or more trusted CA certificates. If ca_cert and ca_path are not included, server certificate will not be verified. This is insecure and a trusted CA certificate should always be configured when using EAP-TLS/TTLS/PEAP. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background.

Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

On Windows, trusted CA certificates can be loaded from the system certificate store by setting this to cert_store://<name>, e.g., ca_cert="cert_store://CA" or ca_cert="cert_store://ROOT". Note that when running wpa_supplicant as an application, the user certificate store (My user account) is used, whereas computer store (Computer account) is used when running wpasvc as a service.

Definition at line 312 of file config_ssid.h.

#### 5.30.2.6 u8* wpa_ssid::ca_cert2

File path to CA certificate file (PEM/DER) (Phase 2).

This file can have one or more trusted CA certificates. If ca_cert2 and ca_path2 are not included, server certificate will not be verified. This is insecure and a trusted CA certificate should always be configured. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background.

This field is like ca_cert, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication.

Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

Definition at line 445 of file config_ssid.h.

#### 5.30.2.7 u8* wpa_ssid::ca_path

Directory path for CA certificate files (PEM).

This path may contain multiple CA certificates in OpenSSL format. Common use for this is to point to system trusted CA list which is often installed into directory like /etc/ssl/certs. If configured, these certificates are added to the list of trusted CAs. ca_cert may also be included in that case, but it is not required.

Definition at line 324 of file config_ssid.h.

#### 5.30.2.8 u8* wpa_ssid::ca_path2

Directory path for CA certificate files (PEM) (Phase 2).

This path may contain multiple CA certificates in OpenSSL format. Common use for this is to point to system trusted CA list which is often installed into directory like /etc/ssl/certs. If configured, these certificates are added to the list of trusted CAs. ca_cert may also be included in that case, but it is not required.

This field is like ca_path, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication.

Definition at line 460 of file config_ssid.h.

#### 5.30.2.9 u8* wpa_ssid::client_cert

File path to client certificate file (PEM/DER).

This field is used with EAP method that use TLS authentication. Usually, this is only configured for EAP-TLS, even though this could in theory be used with EAP-TTLS and EAP-PEAP, too. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background.

Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

Definition at line 339 of file config_ssid.h.

#### 5.30.2.10 u8* wpa_ssid::client_cert2

File path to client certificate file.

This field is like client_cert, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background.

Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

Definition at line 474 of file config_ssid.h.

#### 5.30.2.11 u8* wpa_ssid::dh_file

File path to DH/DSA parameters file (in PEM format).

This is an optional configuration file for setting parameters for an ephemeral DH key exchange. In most cases, the default RSA authentication does not use this configuration. However, it is possible setup RSA to use ephemeral DH key exchange. In addition, ciphers with DSA keys always use ephemeral DH keys. This can be used to achieve forward secrecy. If the file is in DSA parameters format, it will be automatically converted into DH params. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background.

Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

Definition at line 394 of file config_ssid.h.

#### 5.30.2.12 u8* wpa_ssid::dh_file2

File path to DH/DSA parameters file (in PEM format).

This field is like dh_file, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background.

Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

Definition at line 511 of file config_ssid.h.

#### 5.30.2.13 int wpa_ssid::disabled

Whether this network is currently disabled.

0 = this network can be used (default). 1 = this network block is disabled (can be enabled through ctrl_iface, e.g., with wpa_cli or wpa_gui).

Definition at line 870 of file config_ssid.h.

#### 5.30.2.14 struct eap_method_type* wpa_ssid::eap_methods

Allowed EAP methods.

(vendor=EAP_VENDOR_IETF,method=EAP_TYPE_NONE) terminated list of allowed EAP methods or NULL if all methods are accepted.

Definition at line 538 of file config_ssid.h.

#### 5.30.2.15 unsigned int wpa_ssid::eap_workaround

EAP workarounds enabled.

wpa_supplicant supports number of "EAP workarounds" to work around interoperability issues with incorrectly behaving authentication servers. This is recommended to be enabled by default because some of the issues are present in large number of authentication servers.

Strict EAP conformance mode can be configured by disabling workarounds with eap_workaround = 0.

Definition at line 799 of file config_ssid.h.

#### 5.30.2.16 size_t wpa_ssid::eappsk_len

EAP-PSK/PAX/SAKE pre-shared key length.

This field is always 16 for the current version of EAP-PSK/PAX and 32 for EAP-SAKE.

Definition at line 265 of file config_ssid.h.

#### 5.30.2.17 int wpa_ssid::engine

Enable OpenSSL engine (e.g., for smartcard access).

This is used if private key operations for EAP-TLS are performed using a smartcard.

Definition at line 616 of file config_ssid.h.

#### 5.30.2.18 char* wpa_ssid::engine_id

Engine ID for OpenSSL engine.

"opensc" to select OpenSC engine or "pkcs11" to select PKCS#11 engine.

This is used if private key operations for EAP-TLS are performed using a smartcard.

Definition at line 628 of file config_ssid.h.

#### 5.30.2.19 int wpa_ssid::fragment_size

Maximum EAP fragment size in bytes (default 1398).

This value limits the fragment size for EAP methods that support fragmentation (e.g., EAP-TLS and EAP-PEAP). This value should be set small enough to make the EAP messages fit in MTU of the network interface used for EAPOL. The default value is suitable for most cases.

Definition at line 896 of file config_ssid.h.

#### 5.30.2.20 int wpa_ssid::id

Unique id for the network.

This identifier is used as a unique identifier for each network block when using the control interface. Each network is allocated an id when it is being created, either when reading the configuration file or when a new network is added through the control interface.

Definition at line 99 of file config_ssid.h.

#### 5.30.2.21 char* wpa_ssid::id_str

Network identifier string for external scripts.

This value is passed to external ctrl_iface monitors in WPA_EVENT_CONNECTED event and wpa_cli sets this as WPA_ID_STR environment variable for action scripts.

Definition at line 908 of file config_ssid.h.

#### 5.30.2.22 char* wpa_ssid::key_id

Key ID for OpenSSL engine.

This is used if private key operations for EAP-TLS are performed using a smartcard.

Definition at line 637 of file config_ssid.h.

#### 5.30.2.23 int wpa_ssid::key_mgmt

Bitfield of allowed key management protocols.

WPA_KEY_MGMT_*

Definition at line 196 of file config_ssid.h.

#### 5.30.2.24 int wpa_ssid::leap

Number of EAP methods using LEAP.

This field should be set to 1 if LEAP is enabled. This is used to select IEEE 802.11 authentication algorithm.

Definition at line 775 of file config_ssid.h.

#### 5.30.2.25 int wpa_ssid::mode

IEEE 802.11 operation mode (Infrastucture/IBSS).

0 = infrastructure (Managed) mode, i.e., associate with an AP.

1 = IBSS (ad-hoc, peer-to-peer)

Note: IBSS can only be used with key_mgmt NONE (plaintext and static WEP) and key_mgmt=WPA-NONE (fixed group key TKIP/CCMP). In addition, ap_scan has to be set to 2 for IBSS. WPA-None requires following network block options: proto=WPA, key_mgmt=WPA-NONE, pairwise=NONE, group=TKIP (or CCMP, but not both), and psk must also be set (either directly or using ASCII passphrase).

Definition at line 831 of file config_ssid.h.

#### 5.30.2.26 int wpa_ssid::mschapv2_retry

MSCHAPv2 retry in progress.

This field is used internally by EAP-MSCHAPv2 and should not be set as part of configuration.

Definition at line 842 of file config_ssid.h.

#### 5.30.2.27 u8* wpa_ssid::new_password

New password for password update.

This field is used during MSCHAPv2 password update. This is normally requested from the user through the control interface and not set from configuration.

Definition at line 852 of file config_ssid.h.

#### 5.30.2.28 struct wpa_ssid* wpa_ssid::next

Next network in global list.

This pointer can be used to iterate over all networks. The head of this list is stored in the ssid field of struct wpa_config.

Definition at line 78 of file config_ssid.h.

#### 5.30.2.29 int wpa_ssid::non_leap

Number of EAP methods not using LEAP.

This field should be set to >0 if any EAP method other than LEAP is enabled. This is used to select IEEE 802.11 authentication algorithm.

Definition at line 785 of file config_ssid.h.

#### 5.30.2.30 u8* wpa_ssid::otp

One-time-password.

This field should not be set in configuration step. It is only used internally when OTP is entered through the control interface.

Definition at line 694 of file config_ssid.h.

#### 5.30.2.31 char* wpa_ssid::pac_file

File path or blob name for the PAC entries (EAP-FAST).

wpa_supplicant will need to be able to create this file and write updates to it when PAC is being provisioned or refreshed. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background. Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

Definition at line 812 of file config_ssid.h.

#### 5.30.2.32 char* wpa_ssid::passphrase

WPA ASCII passphrase.

If this is set, psk will be generated using the SSID and passphrase configured for the network. ASCII passphrase must be between 8 and 63 characters (inclusive).

Definition at line 176 of file config_ssid.h.

#### 5.30.2.33 char* wpa_ssid::pcsc

Parameters for PC/SC smartcard interface for USIM and GSM SIM.

This field is used to configure PC/SC smartcard interface. Currently, the only configuration is whether this field is NULL (do not use PC/SC) or non-NULL (e.g., "") to enable PC/SC.

This field is used for EAP-SIM and EAP-AKA.

Definition at line 595 of file config_ssid.h.

#### 5.30.2.34 int wpa_ssid::peerkey

Whether PeerKey handshake for direct links is allowed.

This is only used when both RSN/WPA2 and IEEE 802.11e (QoS) are enabled.

0 = disabled (default) 1 = enabled

Definition at line 882 of file config_ssid.h.

#### 5.30.2.35 int wpa_ssid::pending_req_identity

Whether there is a pending identity request.

This field should not be set in configuration step. It is only used internally when control interface is used to request needed information.

Definition at line 710 of file config_ssid.h.

#### 5.30.2.36 int wpa_ssid::pending_req_new_password

Pending password update request.

This field should not be set in configuration step. It is only used internally when control interface is used to request needed information.

Definition at line 740 of file config_ssid.h.

#### 5.30.2.37 char* wpa_ssid::pending_req_otp

Whether there is a pending OTP request.

This field should not be set in configuration step. It is only used internally when control interface is used to request needed information.

Definition at line 760 of file config_ssid.h.

#### 5.30.2.38 int wpa_ssid::pending_req_passphrase

Pending passphrase request.

This field should not be set in configuration step. It is only used internally when control interface is used to request needed information.

Definition at line 750 of file config_ssid.h.

#### 5.30.2.39 int wpa_ssid::pending_req_password

Whether there is a pending password request.

This field should not be set in configuration step. It is only used internally when control interface is used to request needed information.

Definition at line 720 of file config_ssid.h.

#### 5.30.2.40 int wpa_ssid::pending_req_pin

Whether there is a pending PIN request.

This field should not be set in configuration step. It is only used internally when control interface is used to request needed information.

Definition at line 730 of file config_ssid.h.

#### 5.30.2.41 char* wpa_ssid::phase1

Phase 1 (outer authentication) parameters.

String with field-value pairs, e.g., "peapver=0" or "peapver=1 peaplabel=1".

'peapver' can be used to force which PEAP version (0 or 1) is used.

'peaplabel=1' can be used to force new label, "client PEAP encryption", to be used during key derivation when PEAPv1 or newer.

Most existing PEAPv1 implementation seem to be using the old label, "client EAP encryption", and wpa_supplicant is now using that as the default value.

Some servers, e.g., Radiator, may require peaplabel=1 configuration to interoperate with PEAPv1; see eap_testing.txt for more details.

'peap_outer_success=0' can be used to terminate PEAP authentication on tunneled EAP-Success. This is required with some RADIUS servers that implement draft-josefsson-pppext-eap-tls-eap-05.txt (e.g., Lucent NavisRadius v4.4.0 with PEAP in "IETF Draft 5" mode).

include_tls_length=1 can be used to force wpa_supplicant to include TLS Message Length field in all TLS messages even if they are not fragmented.

sim_min_num_chal=3 can be used to configure EAP-SIM to require three challenges (by default, it accepts 2 or 3).

fast_provisioning=1 can be used to enable in-line provisioning of EAP-FAST credentials (PAC)

Definition at line 574 of file config_ssid.h.

#### 5.30.2.42 char* wpa_ssid::phase2

Phase2 (inner authentication with TLS tunnel) parameters.

String with field-value pairs, e.g., "auth=MSCHAPV2" for EAP-PEAP or "autheap=MSCHAPV2 autheap=MD5" for EAP-TTLS.

Definition at line 583 of file config_ssid.h.

#### 5.30.2.43 char* wpa_ssid::pin

PIN for USIM, GSM SIM, and smartcards.

This field is used to configure PIN for SIM and smartcards for EAP-SIM and EAP-AKA. In addition, this is used with EAP-TLS if a smartcard is used for private key operations.

If left out, this will be asked through control interface.

Definition at line 607 of file config_ssid.h.

#### 5.30.2.44 struct wpa_ssid* wpa_ssid::pnext

Next network in per-priority list.

This pointer can be used to iterate over all networks in the same priority class. The heads of these list are stored in the pssid fields of struct wpa_config.

Definition at line 88 of file config_ssid.h.

#### 5.30.2.45 int wpa_ssid::priority

Priority group.

By default, all networks will get same priority group (0). If some of the networks are more desirable, this field can be used to change the order in which wpa_supplicant goes through the networks when selecting a BSS. The priority groups will be iterated in decreasing priority (i.e., the larger the priority value, the sooner the network is matched against the scan results). Within each priority group, networks will be selected based on security policy, signal strength, etc.

Please note that AP scanning with scan_ssid=1 and ap_scan=2 mode are not using this priority to select the order for scanning. Instead, they try the networks in the order that used in the configuration file.

Definition at line 119 of file config_ssid.h.

#### 5.30.2.46 u8* wpa_ssid::private_key

File path to client private key file (PEM/DER/PFX).

When PKCS#12/PFX file (.p12/.pfx) is used, client_cert should be commented out. Both the private key and certificate will be read from the PKCS#12 file in this case. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background.

Windows certificate store can be used by leaving client_cert out and configuring private_key in one of the following formats:

cert://substring_to_match

hash://certificate_thumbprint_in_hex

For example: private_key="hash://63093aa9c47f56ae88334c7b65a4"

Note that when running wpa_supplicant as an application, the user certificate store (My user account) is used, whereas computer store (Computer account) is used when running wpasvc as a service.

Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

Definition at line 367 of file config_ssid.h.

#### 5.30.2.47 u8* wpa_ssid::private_key2

File path to client private key file.

This field is like private_key, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication. Full path to the file should be used since working directory may change when wpa_supplicant is run in the background.

Alternatively, a named configuration blob can be used by setting this to blob://<blob name="">.

Definition at line 488 of file config_ssid.h.

#### 5.30.2.48 u8* wpa_ssid::private_key2_passwd

Password for private key file.

This field is like private_key_passwd, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication.

Definition at line 497 of file config_ssid.h.

#### 5.30.2.49 u8* wpa_ssid::private_key_passwd

Password for private key file.

If left out, this will be asked through control interface.

Definition at line 375 of file config_ssid.h.

#### 5.30.2.50 int wpa_ssid::proactive_key_caching

Enable proactive key caching.

This field can be used to enable proactive key caching which is also known as opportunistic PMKSA caching for WPA2. This is disabled (0) by default. Enable by setting this to 1.

Proactive key caching is used to make supplicant assume that the APs are using the same PMK and generate PMKSA cache entries without doing RSN pre-authentication. This requires support from the AP side and is normally used with wireless switches that co-locate the authenticator.

Definition at line 683 of file config_ssid.h.

#### 5.30.2.51 int wpa_ssid::scan_ssid

Scan this SSID with Probe Requests.

scan_ssid can be used to scan for APs using hidden SSIDs. Note: Many drivers do not support this. ap_-mode=2 can be used with such drivers to use hidden SSIDs.

Definition at line 220 of file config_ssid.h.

#### 5.30.2.52 u8* wpa_ssid::ssid

Service set identifier (network name).

This is the SSID for the network. For wireless interfaces, this is used to select which network will be used. If set to NULL (or ssid_len=0), any SSID can be used. For wired interfaces, this must be set to NULL. Note: SSID may contain any characters, even nul (ASCII 0) and as such, this should not be assumed to be a nul terminated string. ssid_len defines how many characters are valid and the ssid field is not guaranteed to be nul terminated.

Definition at line 133 of file config_ssid.h.

#### 5.30.2.53 u8* wpa_ssid::subject_match

Constraint for server certificate subject.

This substring is matched against the subject of the authentication server certificate. If this string is set, the server sertificate is only accepted if it contains this string in the subject. The subject string is in following format:

/C=US/ST=CA/L=San Francisco/CN=Test AS/emailAddress=as

.example.com

Definition at line 407 of file config_ssid.h.

#### 5.30.2.54 u8* wpa_ssid::subject_match2

Constraint for server certificate subject.

This field is like subject_match, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication.

Definition at line 520 of file config_ssid.h.

The documentation for this struct was generated from the following file:

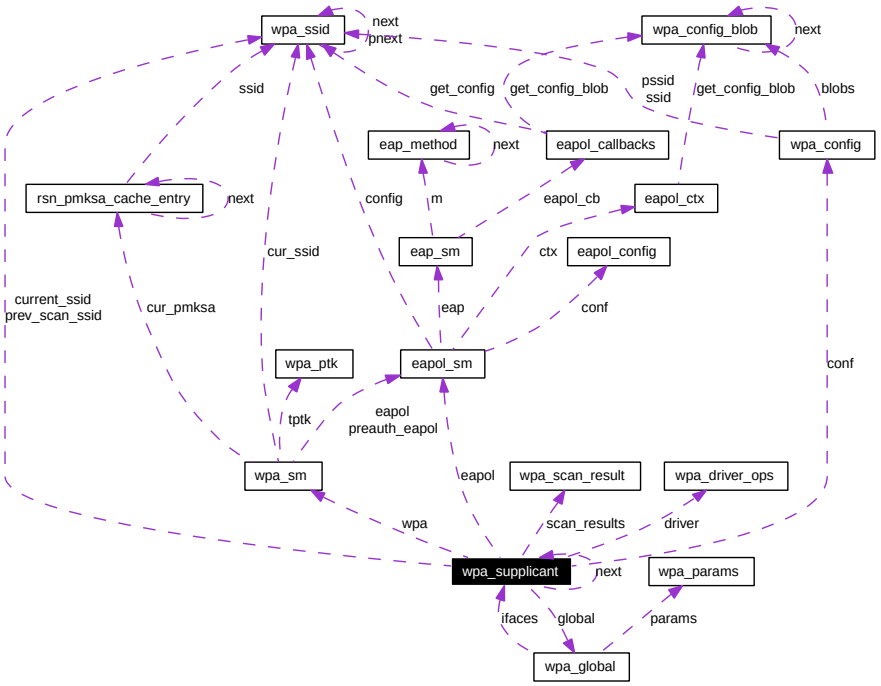
- config_ssid.h

# 5.31 wpa_supplicant Struct Reference

Internal data for wpa_supplicant interface.

`#include <wpa_supplicant_i.h>`

Collaboration diagram for wpa_supplicant:




## Data Fields

- wpa_global * **global**
- wpa_supplicant * **next**
- l2_packet_data * **l2**
- l2_packet_data * **l2_br**
- unsigned char **own_addr** [ETH_ALEN]
- char **ifname** [100]
- char **bridge_ifname** [16]
- char * **confname**
- wpa_config * **conf**
- int **countermeasures**
- os_time_t **last_michael_mic_error**
- u8 **bssid** [ETH_ALEN]
- u8 **pending_bssid** [ETH_ALEN]
- int **reassociate**
- int **disconnected**
- wpa_ssid * **current_ssid**
- int **pairwise_cipher**
- int **group_cipher**
- int **key_mgmt**

- int **mgmt_group_cipher**
- void * **drv_priv**
- wpa_ssid * **prev_scan_ssid**
- wpa_scan_result * **scan_results**
- int **num_scan_results**
- wpa_driver_ops * **driver**
- int **interface_removed**
- wpa_sm * **wpa**
- eapol_sm * **eapol**
- ctrl_iface_priv * **ctrl_iface**
- wpa_states **wpa_state**
- int **new_connection**
- int **reassociated_connection**
- int **eapol_received**
- scard_data * **scard**
- unsigned char **last_eapol_src** [ETH_ALEN]
- int **keys_cleared**
- wpa_blacklist * **blacklist**
- int **scan_req**
- int **scan_res_tried**
- wpa_client_mlme **mlme**
- int **use_client_mlme**

### 5.31.1 Detailed Description

Internal data for wpa_supplicant interface.

This structure contains the internal data for core wpa_supplicant code. This should be only used directly from the core code. However, a pointer to this data is used from other files as an arbitrary context pointer in calls to core functions.

Definition at line 291 of file wpa_supplicant_i.h.

The documentation for this struct was generated from the following file:

- wpa_supplicant_i.h

# Chapter 6

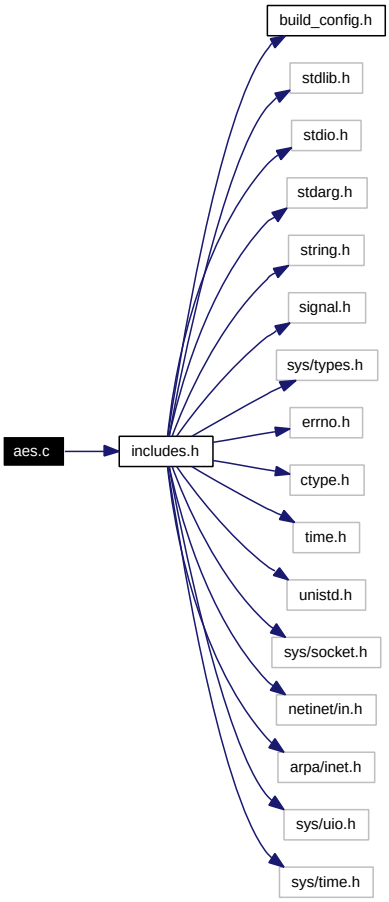
# wpa_supplicant File Documentation

## 6.1 aes.c File Reference

AES (Rijndael) cipher.

```
#include "includes.h"
```

Include dependency graph for aes.c:



## Defines

- #define **AES_SMALL_TABLES**
- #define **RCON**(i) (rcons[(i)] << 24)
- #define **TE0**(i) Te0[((i) >> 24) & 0xff]
- #define **TE1**(i) rotr(Te0[((i) >> 16) & 0xff], 8)
- #define **TE2**(i) rotr(Te0[((i) >> 8) & 0xff], 16)
- #define **TE3**(i) rotr(Te0[(i) & 0xff], 24)
- #define **TE41**(i) ((Te0[((i) >> 24) & 0xff] << 8) & 0xff000000)
- #define **TE42**(i) (Te0[((i) >> 16) & 0xff] & 0x00ff0000)
- #define **TE43**(i) (Te0[((i) >> 8) & 0xff] & 0x0000ff00)
- #define **TE44**(i) ((Te0[(i) & 0xff] >> 8) & 0x000000ff)
- #define **TE421**(i) ((Te0[((i) >> 16) & 0xff] << 8) & 0xff000000)
- #define **TE432**(i) (Te0[((i) >> 8) & 0xff] & 0x00ff0000)
- #define **TE443**(i) (Te0[(i) & 0xff] & 0x0000ff00)
- #define **TE414**(i) ((Te0[((i) >> 24) & 0xff] >> 8) & 0x000000ff)
- #define **TE4**(i) ((Te0[(i)] >> 8) & 0x000000ff)
- #define **TD0**(i) Td0[((i) >> 24) & 0xff]
- #define **TD1**(i) rotr(Td0[((i) >> 16) & 0xff], 8)
- #define **TD2**(i) rotr(Td0[((i) >> 8) & 0xff], 16)
- #define **TD3**(i) rotr(Td0[(i) & 0xff], 24)
- #define **TD41**(i) (Td4s[((i) >> 24) & 0xff] << 24)
- #define **TD42**(i) (Td4s[((i) >> 16) & 0xff] << 16)
- #define **TD43**(i) (Td4s[((i) >> 8) & 0xff] << 8)
- #define **TD44**(i) (Td4s[(i) & 0xff])
- #define **TD0_**(i) Td0[(i) & 0xff]
- #define **TD1_**(i) rotr(Td0[(i) & 0xff], 8)
- #define **TD2_**(i) rotr(Td0[(i) & 0xff], 16)
- #define **TD3_**(i) rotr(Td0[(i) & 0xff], 24)
- #define **SWAP**(x) (_lrotl(x, 8) & 0x00ff00ff | _lrotr(x, 8) & 0xff00ff00)
- #define **GETU32**(pt)
- #define **PUTU32**(ct, st)
- #define **ROUND**(i, d, s)
- #define **ROUND**(i, d, s)

## Functions

- void rijndaelKeySetupEnc (u32 rk[ ], const u8 cipherKey[ ])
- void rijndaelKeySetupDec (u32 rk[ ], const u8 cipherKey[ ])
- void **rijndaelEncrypt** (const u32 rk[ ], const u8 pt[16], u8 ct[16])
- void **rijndaelDecrypt** (const u32 rk[ ], const u8 ct[16], u8 pt[16])
- void * aes_encrypt_init (const u8 *key, size_t len)

  *Initialize AES for encryption.*

- void aes_encrypt (void *ctx, const u8 *plain, u8 *crypt)

  *Encrypt one AES block.*

- void aes_encrypt_deinit (void *ctx)

  *Deinitialize AES encryption.*

- void * aes_decrypt_init (const u8 *key, size_t len)
  *Initialize AES for decryption.*
- void aes_decrypt (void *ctx, const u8 *crypt, u8 *plain)
  *Decrypt one AES block.*
- void aes_decrypt_deinit (void *ctx)
  *Deinitialize AES decryption.*

### 6.1.1 Detailed Description

AES (Rijndael) cipher.

Modifications to public domain implementation:

- support only 128-bit keys
- cleanup
- use C pre-processor to make it easier to change S table access
- added option (AES_SMALL_TABLES) for reducing code size by about 8 kB at cost of reduced throughput (quite small difference on Pentium 4, 10-25% when using -O1 or -O2 optimization)

**Copyright**
Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file aes.c.

### 6.1.2 Define Documentation

#### 6.1.2.1 #define GETU32(pt)

**Value:**

```
(((u32)(pt)[0] << 24) ^ ((u32)(pt)[1] << 16) ^ \
((u32)(pt)[2] <<  8) ^ ((u32)(pt)[3]))
```

Definition at line 858 of file aes.c.

#### 6.1.2.2 #define PUTU32(ct, st)

**Value:**

```
{ \
(ct)[0] = (u8)((st) >> 24); (ct)[1] = (u8)((st) >> 16); \
(ct)[2] = (u8)((st) >>  8); (ct)[3] = (u8)(st); }
```

Definition at line 860 of file aes.c.

#### 6.1.2.3 #define ROUND(i, d, s)

**Value:**

```
d##0 = TD0(s##0) ^ TD1(s##3) ^ TD2(s##2) ^ TD3(s##1) ^ rk[4 * i]; \
d##1 = TD0(s##1) ^ TD1(s##0) ^ TD2(s##3) ^ TD3(s##2) ^ rk[4 * i + 1]; \
d##2 = TD0(s##2) ^ TD1(s##1) ^ TD2(s##0) ^ TD3(s##3) ^ rk[4 * i + 2]; \
d##3 = TD0(s##3) ^ TD1(s##2) ^ TD2(s##1) ^ TD3(s##0) ^ rk[4 * i + 3]
```

#### 6.1.2.4 #define ROUND(i, d, s)

**Value:**

```
d##0 = TE0(s##0) ^ TE1(s##1) ^ TE2(s##2) ^ TE3(s##3) ^ rk[4 * i]; \
d##1 = TE0(s##1) ^ TE1(s##2) ^ TE2(s##3) ^ TE3(s##0) ^ rk[4 * i + 1]; \
d##2 = TE0(s##2) ^ TE1(s##3) ^ TE2(s##0) ^ TE3(s##1) ^ rk[4 * i + 2]; \
d##3 = TE0(s##3) ^ TE1(s##0) ^ TE2(s##1) ^ TE3(s##2) ^ rk[4 * i + 3]
```

### 6.1.3 Function Documentation

#### 6.1.3.1 void aes_decrypt (void * *ctx*, const u8 * *crypt*, u8 * *plain*)

Decrypt one AES block.

**Parameters:**

***ctx*** Context pointer from aes_encrypt_init()

***crypt*** Encrypted data (16 bytes)

***plain*** Buffer for the decrypted data (16 bytes)

Definition at line 1099 of file aes.c.

#### 6.1.3.2 void aes_decrypt_deinit (void * *ctx*)

Deinitialize AES decryption.

**Parameters:**

***ctx*** Context pointer from aes_encrypt_init()

Definition at line 1105 of file aes.c.

#### 6.1.3.3 void* aes_decrypt_init (const u8 * *key*, size_t *len*)

Initialize AES for decryption.

**Parameters:**

***key*** Decryption key

***len*** Key length in bytes (usually 16, i.e., 128 bits)

**Returns:**

Pointer to context data or NULL on failure

Definition at line 1086 of file aes.c.

Here is the call graph for this function:


#### 6.1.3.4 void aes_encrypt (void * *ctx*, const u8 * *plain*, u8 * *crypt*)

Encrypt one AES block.

**Parameters:**

***ctx*** Context pointer from aes_encrypt_init()

***plain*** Plaintext data to be encrypted (16 bytes)

***crypt*** Buffer for the encrypted data (16 bytes)

Definition at line 1074 of file aes.c.

#### 6.1.3.5 void aes_encrypt_deinit (void * *ctx*)

Deinitialize AES encryption.

**Parameters:**

***ctx*** Context pointer from aes_encrypt_init()

Definition at line 1080 of file aes.c.

#### 6.1.3.6 void* aes_encrypt_init (const u8 * *key*, size_t *len*)

Initialize AES for encryption.

**Parameters:**

***key*** Encryption key

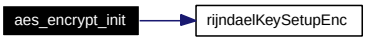
***len*** Key length in bytes (usually 16, i.e., 128 bits)

**Returns:**

Pointer to context data or NULL on failure

Definition at line 1061 of file aes.c.

Here is the call graph for this function:



#### 6.1.3.7 void rijndaelKeySetupDec (u32 *rk*[ ], const u8 *cipherKey*[ ])

Expand the cipher key into the decryption key schedule.

**Returns:**
the number of rounds for the given cipher key size.

Definition at line 896 of file aes.c.

Here is the call graph for this function:


#### 6.1.3.8 void rijndaelKeySetupEnc (u32 *rk*[ ], const u8 *cipherKey*[ ])

Expand the cipher key into the encryption key schedule.

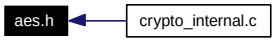
**Returns:**
the number of rounds for the given cipher key size.

Definition at line 870 of file aes.c.

## 6.2 aes.h File Reference

AES functions.

This graph shows which files directly or indirectly include this file:




### Functions

- void * **aes_encrypt_init** (const u8 *key, size_t len)
- void **aes_encrypt** (void *ctx, const u8 *plain, u8 *crypt)
- void **aes_encrypt_deinit** (void *ctx)
- void * **aes_decrypt_init** (const u8 *key, size_t len)
- void **aes_decrypt** (void *ctx, const u8 *crypt, u8 *plain)
- void **aes_decrypt_deinit** (void *ctx)

### 6.2.1 Detailed Description

AES functions.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

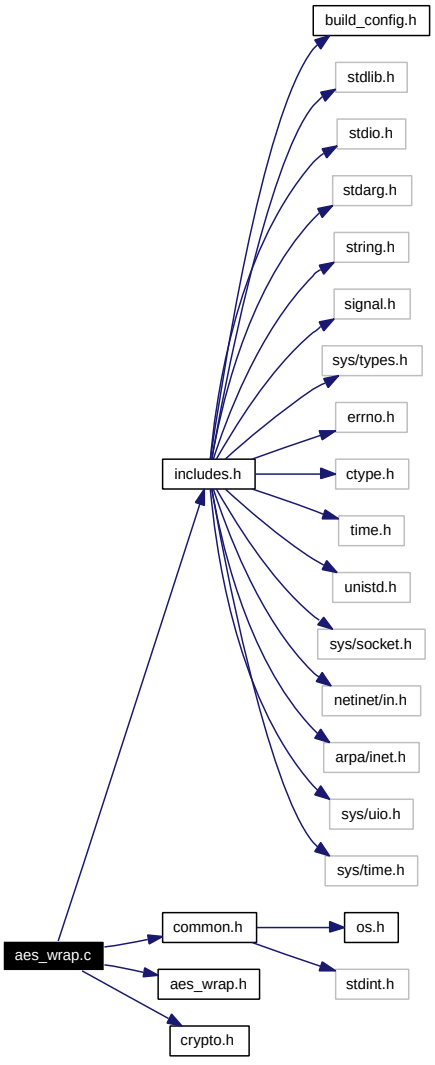
Definition in file aes.h.

## 6.3 aes_wrap.c File Reference

AES-based functions.

```
#include "includes.h"
#include "common.h"
#include "aes_wrap.h"
#include "crypto.h"
```

Include dependency graph for aes_wrap.c:




### Defines

- #define **BLOCK_SIZE** 16

## Functions

- int aes_wrap (const u8 *kek, int n, const u8 *plain, u8 *cipher)
  *Wrap keys with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).*
- int aes_unwrap (const u8 *kek, int n, const u8 *cipher, u8 *plain)
  *Unwrap key with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).*
- int omac1_aes_128 (const u8 *key, const u8 *data, size_t data_len, u8 *mac)
  *One-Key CBC MAC (OMAC1) hash with AES-128 (aka AES-CMAC).*
- int aes_128_encrypt_block (const u8 *key, const u8 *in, u8 *out)
  *Perform one AES 128-bit block operation.*
- int aes_128_ctr_encrypt (const u8 *key, const u8 *nonce, u8 *data, size_t data_len)
  *AES-128 CTR mode encryption.*
- int aes_128_eax_encrypt (const u8 *key, const u8 *nonce, size_t nonce_len, const u8 *hdr, size_t hdr_len, u8 *data, size_t data_len, u8 *tag)
  *AES-128 EAX mode encryption.*
- int aes_128_eax_decrypt (const u8 *key, const u8 *nonce, size_t nonce_len, const u8 *hdr, size_t hdr_len, u8 *data, size_t data_len, const u8 *tag)
  *AES-128 EAX mode decryption.*
- int aes_128_cbc_encrypt (const u8 *key, const u8 *iv, u8 *data, size_t data_len)
  *AES-128 CBC encryption.*
- int aes_128_cbc_decrypt (const u8 *key, const u8 *iv, u8 *data, size_t data_len)
  *AES-128 CBC decryption.*

### 6.3.1 Detailed Description

AES-based functions.

- AES Key Wrap Algorithm (128-bit KEK) (RFC3394)
  - One-Key CBC MAC (OMAC1) hash with AES-128
  - AES-128 CTR mode encryption
  - AES-128 EAX mode encryption/decryption
  - AES-128 CBC

**Copyright**
Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file aes_wrap.c.

## 6.3.2 Function Documentation

### 6.3.2.1 int aes_128_cbc_decrypt (const u8 * *key*, const u8 * *iv*, u8 * *data*, size_t *data_len*)

AES-128 CBC decryption.

**Parameters:**

***key*** Decryption key

***iv*** Decryption IV for CBC mode (16 bytes)

***data*** Data to decrypt in-place

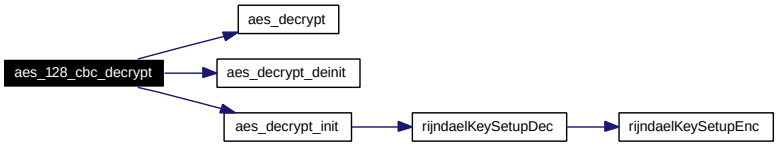
***data_len*** Length of data in bytes (must be divisible by 16)

**Returns:**

0 on success, -1 on failure

Definition at line 457 of file aes_wrap.c.

Here is the call graph for this function:




### 6.3.2.2 int aes_128_cbc_encrypt (const u8 * *key*, const u8 * *iv*, u8 * *data*, size_t *data_len*)

AES-128 CBC encryption.

**Parameters:**

***key*** Encryption key

***iv*** Encryption IV for CBC mode (16 bytes)

***data*** Data to encrypt in-place

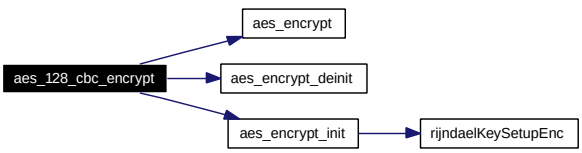
***data_len*** Length of data in bytes (must be divisible by 16)

**Returns:**

0 on success, -1 on failure

Definition at line 423 of file aes_wrap.c.

Here is the call graph for this function:



#### 6.3.2.3 int aes_128_ctr_encrypt (const u8 * *key*, const u8 * *nonce*, u8 * *data*, size_t *data_len*)

AES-128 CTR mode encryption.

**Parameters:**

***key*** Key for encryption (16 bytes)

***nonce*** Nonce for counter mode (16 bytes)

***data*** Data to encrypt in-place

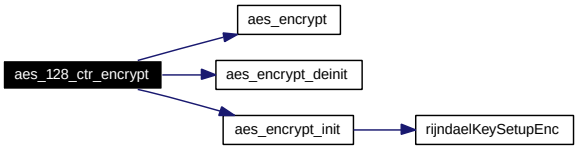
***data_len*** Length of data in bytes

**Returns:**

0 on success, -1 on failure

Definition at line 253 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.3.2.4 int aes_128_eax_decrypt (const u8 * *key*, const u8 * *nonce*, size_t *nonce_len*, const u8 * *hdr*, size_t *hdr_len*, u8 * *data*, size_t *data_len*, const u8 * *tag*)

AES-128 EAX mode decryption.

**Parameters:**

***key*** Key for decryption (16 bytes)

***nonce*** Nonce for counter mode

***nonce_len*** Nonce length in bytes

***hdr*** Header data to be authenticity protected

***hdr_len*** Length of the header data bytes

***data*** Data to encrypt in-place

***data_len*** Length of data in bytes

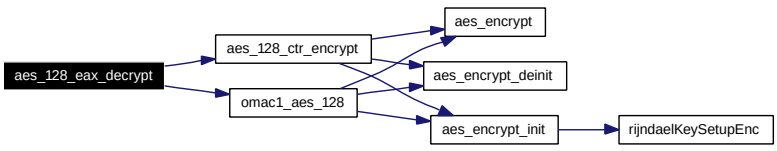
***tag*** 16-byte tag value

**Returns:**

0 on success, -1 on failure, -2 if tag does not match

Definition at line 362 of file aes_wrap.c.

Here is the call graph for this function:



#### 6.3.2.5 int aes_128_eax_encrypt (const u8 * *key*, const u8 * *nonce*, size_t *nonce_len*, const u8 * *hdr*, size_t *hdr_len*, u8 * *data*, size_t *data_len*, u8 * *tag*)

AES-128 EAX mode encryption.

**Parameters:**

***key*** Key for encryption (16 bytes)

***nonce*** Nonce for counter mode

***nonce_len*** Nonce length in bytes

***hdr*** Header data to be authenticity protected

***hdr_len*** Length of the header data bytes

***data*** Data to encrypt in-place

***data_len*** Length of data in bytes

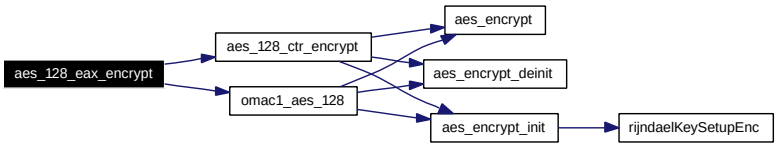
***tag*** 16-byte tag value

**Returns:**

0 on success, -1 on failure

Definition at line 304 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.3.2.6 int aes_128_encrypt_block (const u8 * *key*, const u8 * *in*, u8 * *out*)

Perform one AES 128-bit block operation.

**Parameters:**

***key*** Key for AES

***in*** Input data (16 bytes)

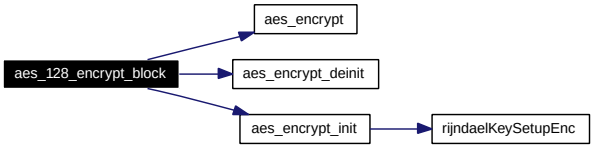
***out*** Output of the AES block operation (16 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 230 of file aes_wrap.c.

Here is the call graph for this function:



#### 6.3.2.7 int aes_unwrap (const u8 * *kek*, int *n*, const u8 * *cipher*, u8 * *plain*)

Unwrap key with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).

**Parameters:**

***kek*** Key encryption key (KEK)

***n*** Length of the wrapped key in 64-bit units; e.g., 2 = 128-bit = 16 bytes

***cipher*** Wrapped key to be unwrapped, (n + 1) * 64 bit

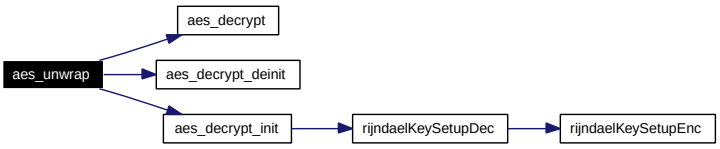
***plain*** Plaintext key, n * 64 bit

**Returns:**

0 on success, -1 on failure (e.g., integrity verification failed)

Definition at line 104 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.3.2.8 int aes_wrap (const u8 * *kek*, int *n*, const u8 * *plain*, u8 * *cipher*)

Wrap keys with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).

**Parameters:**

***kek*** Key encryption key (KEK)

***n*** Length of the wrapped key in 64-bit units; e.g., 2 = 128-bit = 16 bytes

***plain*** Plaintext key to be wrapped, n * 64 bit

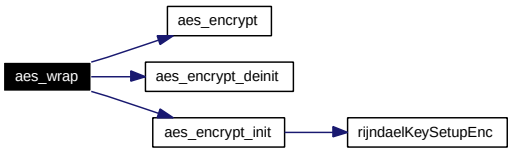
***cipher*** Wrapped key, (n + 1) * 64 bit

**Returns:**

0 on success, -1 on failure

Definition at line 45 of file aes_wrap.c.

Here is the call graph for this function:



#### 6.3.2.9 int omac1_aes_128 (const u8 * *key*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

One-Key CBC MAC (OMAC1) hash with AES-128 (aka AES-CMAC).

**Parameters:**

***key*** 128-bit key for the hash operation

***data*** Data buffer for which a MAC is determined

***data*** Length of data buffer in bytes

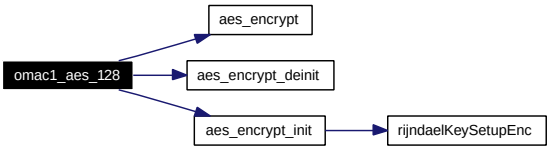
***mac*** Buffer for MAC (128 bits, i.e., 16 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 181 of file aes_wrap.c.

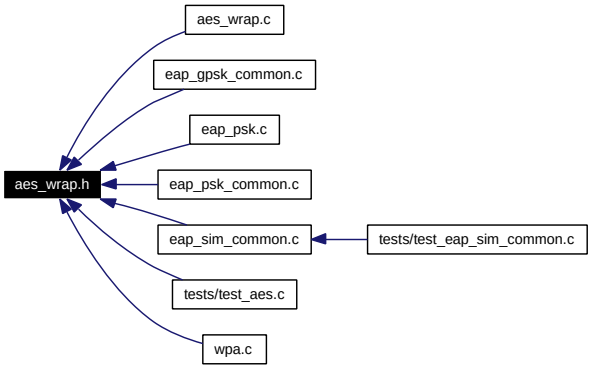
Here is the call graph for this function:



# 6.4 aes_wrap.h File Reference

AES-based functions.

This graph shows which files directly or indirectly include this file:




## Functions

- int aes_wrap (const u8 *kek, int n, const u8 *plain, u8 *cipher)

  *Wrap keys with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).*

- int aes_unwrap (const u8 *kek, int n, const u8 *cipher, u8 *plain)

  *Unwrap key with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).*

- int omac1_aes_128 (const u8 *key, const u8 *data, size_t data_len, u8 *mac)

  *One-Key CBC MAC (OMAC1) hash with AES-128 (aka AES-CMAC).*

- int aes_128_encrypt_block (const u8 *key, const u8 *in, u8 *out)

  *Perform one AES 128-bit block operation.*

- int aes_128_ctr_encrypt (const u8 *key, const u8 *nonce, u8 *data, size_t data_len)

  *AES-128 CTR mode encryption.*

- int aes_128_eax_encrypt (const u8 *key, const u8 *nonce, size_t nonce_len, const u8 *hdr, size_t hdr_len, u8 *data, size_t data_len, u8 *tag)

  *AES-128 EAX mode encryption.*

- int aes_128_eax_decrypt (const u8 *key, const u8 *nonce, size_t nonce_len, const u8 *hdr, size_t hdr_len, u8 *data, size_t data_len, const u8 *tag)

  *AES-128 EAX mode decryption.*

- int aes_128_cbc_encrypt (const u8 *key, const u8 *iv, u8 *data, size_t data_len)

  *AES-128 CBC encryption.*

- int aes_128_cbc_decrypt (const u8 *key, const u8 *iv, u8 *data, size_t data_len)

  *AES-128 CBC decryption.*

### 6.4.1 Detailed Description

AES-based functions.

- AES Key Wrap Algorithm (128-bit KEK) (RFC3394)
  - One-Key CBC MAC (OMAC1) hash with AES-128
  - AES-128 CTR mode encryption
  - AES-128 EAX mode encryption/decryption
  - AES-128 CBC

**Copyright**
Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file aes_wrap.h.

### 6.4.2 Function Documentation

#### 6.4.2.1 int aes_128_cbc_decrypt (const u8 * *key*, const u8 * *iv*, u8 * *data*, size_t *data_len*)

AES-128 CBC decryption.

**Parameters:**
***key*** Decryption key

***iv*** Decryption IV for CBC mode (16 bytes)

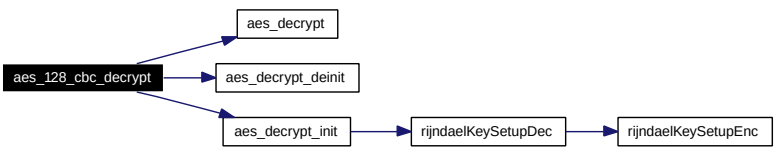
***data*** Data to decrypt in-place

***data_len*** Length of data in bytes (must be divisible by 16)

**Returns:**
0 on success, -1 on failure

Definition at line 457 of file aes_wrap.c.

Here is the call graph for this function:



#### 6.4.2.2 int aes_128_cbc_encrypt (const u8 * *key*, const u8 * *iv*, u8 * *data*, size_t *data_len*)

AES-128 CBC encryption.

**Parameters:**

***key*** Encryption key

***iv*** Encryption IV for CBC mode (16 bytes)

***data*** Data to encrypt in-place

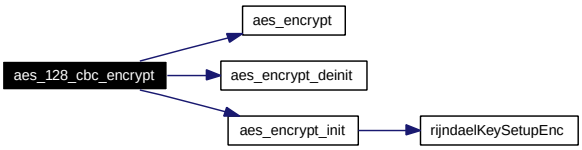
***data_len*** Length of data in bytes (must be divisible by 16)

**Returns:**

0 on success, -1 on failure

Definition at line 423 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.4.2.3 int aes_128_ctr_encrypt (const u8 * *key*, const u8 * *nonce*, u8 * *data*, size_t *data_len*)

AES-128 CTR mode encryption.

**Parameters:**

***key*** Key for encryption (16 bytes)

***nonce*** Nonce for counter mode (16 bytes)

***data*** Data to encrypt in-place

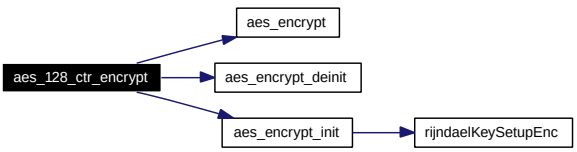
***data_len*** Length of data in bytes

**Returns:**

0 on success, -1 on failure

Definition at line 253 of file aes_wrap.c.

Here is the call graph for this function:



#### 6.4.2.4 int aes_128_eax_decrypt (const u8 * *key*, const u8 * *nonce*, size_t *nonce_len*, const u8 * *hdr*, size_t *hdr_len*, u8 * *data*, size_t *data_len*, const u8 * *tag*)

AES-128 EAX mode decryption.

**Parameters:**

***key*** Key for decryption (16 bytes)

***nonce*** Nonce for counter mode

***nonce_len*** Nonce length in bytes

***hdr*** Header data to be authenticity protected

***hdr_len*** Length of the header data bytes

***data*** Data to encrypt in-place

***data_len*** Length of data in bytes

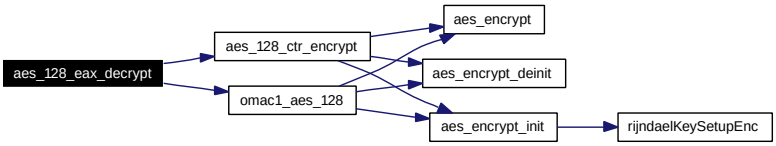
***tag*** 16-byte tag value

**Returns:**

0 on success, -1 on failure, -2 if tag does not match

Definition at line 362 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.4.2.5 int aes_128_eax_encrypt (const u8 * *key*, const u8 * *nonce*, size_t *nonce_len*, const u8 * *hdr*, size_t *hdr_len*, u8 * *data*, size_t *data_len*, u8 * *tag*)

AES-128 EAX mode encryption.

**Parameters:**

***key*** Key for encryption (16 bytes)

***nonce*** Nonce for counter mode

***nonce_len*** Nonce length in bytes

***hdr*** Header data to be authenticity protected

***hdr_len*** Length of the header data bytes

***data*** Data to encrypt in-place

***data_len*** Length of data in bytes

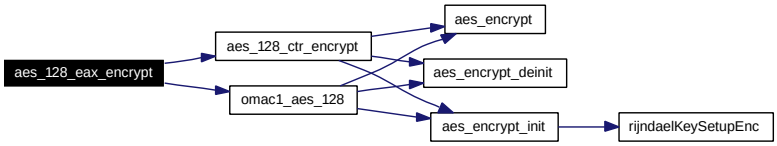
***tag*** 16-byte tag value

**Returns:**

0 on success, -1 on failure

Definition at line 304 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.4.2.6 int aes_128_encrypt_block (const u8 * *key*, const u8 * *in*, u8 * *out*)

Perform one AES 128-bit block operation.

**Parameters:**

***key*** Key for AES

***in*** Input data (16 bytes)

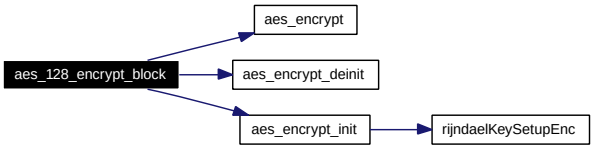
***out*** Output of the AES block operation (16 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 230 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.4.2.7 int aes_unwrap (const u8 * *kek*, int *n*, const u8 * *cipher*, u8 * *plain*)

Unwrap key with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).

**Parameters:**

***kek*** Key encryption key (KEK)

***n*** Length of the wrapped key in 64-bit units; e.g., 2 = 128-bit = 16 bytes

***cipher*** Wrapped key to be unwrapped, (n + 1) * 64 bit

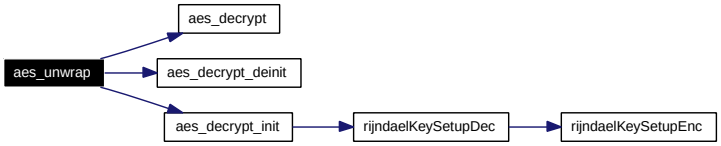
***plain*** Plaintext key, n * 64 bit

**Returns:**

0 on success, -1 on failure (e.g., integrity verification failed)

Definition at line 104 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.4.2.8 int aes_wrap (const u8 * *kek*, int *n*, const u8 * *plain*, u8 * *cipher*)

Wrap keys with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).

**Parameters:**

***kek*** Key encryption key (KEK)

***n*** Length of the wrapped key in 64-bit units; e.g., 2 = 128-bit = 16 bytes

***plain*** Plaintext key to be wrapped, n * 64 bit

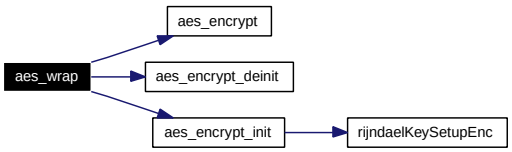
***cipher*** Wrapped key, (n + 1) * 64 bit

**Returns:**

0 on success, -1 on failure

Definition at line 45 of file aes_wrap.c.

Here is the call graph for this function:




#### 6.4.2.9 int omac1_aes_128 (const u8 * *key*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

One-Key CBC MAC (OMAC1) hash with AES-128 (aka AES-CMAC).

**Parameters:**

***key*** 128-bit key for the hash operation

***data*** Data buffer for which a MAC is determined

***data*** Length of data buffer in bytes

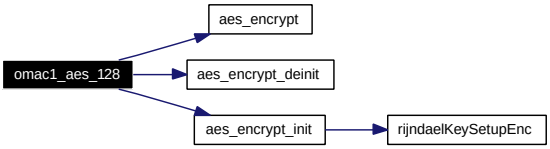
***mac*** Buffer for MAC (128 bits, i.e., 16 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 181 of file aes_wrap.c.

Here is the call graph for this function:


aes_encrypt
omac1_aes_128
aes_encrypt_deinit
aes_encrypt_init
rijndaelKeySetupEnc

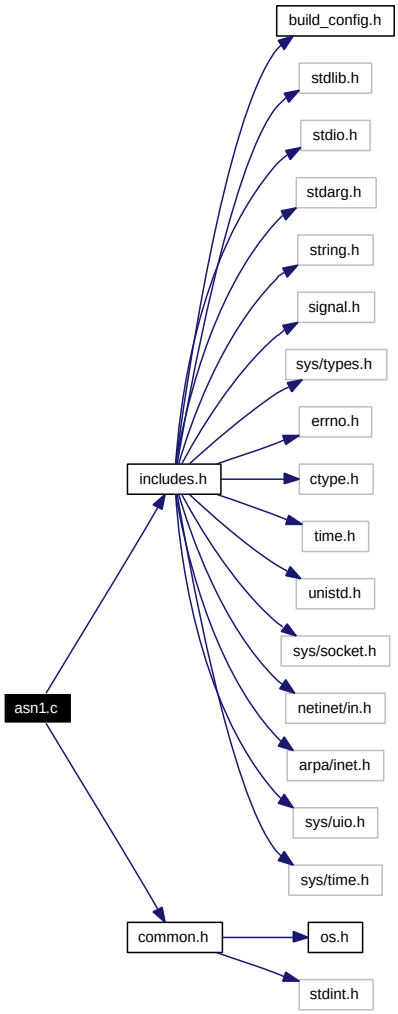
## 6.5 asn1.c File Reference

ASN.1 DER parsing.

```
#include "includes.h"
```

```
#include "common.h"
```

Include dependency graph for asn1.c:




### 6.5.1 Detailed Description

ASN.1 DER parsing.

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

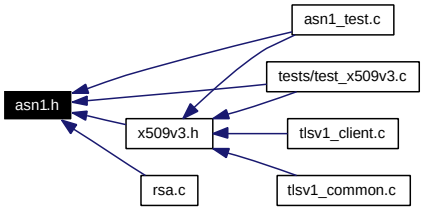
See README and COPYING for more details.

Definition in file asn1.c.

## 6.6 asn1.h File Reference

ASN.1 DER parsing.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **ASN1_TAG_EOC** 0x00
- #define **ASN1_TAG_BOOLEAN** 0x01
- #define **ASN1_TAG_INTEGER** 0x02
- #define **ASN1_TAG_BITSTRING** 0x03
- #define **ASN1_TAG_OCTETSTRING** 0x04
- #define **ASN1_TAG_NULL** 0x05
- #define **ASN1_TAG_OID** 0x06
- #define **ASN1_TAG_OBJECT_DESCRIPTOR** 0x07
- #define **ASN1_TAG_EXTERNAL** 0x08
- #define **ASN1_TAG_REAL** 0x09
- #define **ASN1_TAG_ENUMERATED** 0x0A
- #define **ASN1_TAG_UTF8STRING** 0x0C
- #define **ANS1_TAG_RELATIVE_OID** 0x0D
- #define **ASN1_TAG_SEQUENCE** 0x10
- #define **ASN1_TAG_SET** 0x11
- #define **ASN1_TAG_NUMERICSTRING** 0x12
- #define **ASN1_TAG_PRINTABLESTRING** 0x13
- #define **ASN1_TAG_TG1STRING** 0x14
- #define **ASN1_TAG_VIDEOTEXSTRING** 0x15
- #define **ASN1_TAG_IA5STRING** 0x16
- #define **ASN1_TAG_UTCTIME** 0x17
- #define **ASN1_TAG_GENERALIZEDTIME** 0x18
- #define **ASN1_TAG_GRAPHICSTRING** 0x19
- #define **ASN1_TAG_VISIBLESTRING** 0x1A
- #define **ASN1_TAG_GENERALSTRING** 0x1B
- #define **ASN1_TAG_UNIVERSALSTRING** 0x1C
- #define **ASN1_TAG_BMPSTRING** 0x1D
- #define **ASN1_CLASS_UNIVERSAL** 0
- #define **ASN1_CLASS_APPLICATION** 1
- #define **ASN1_CLASS_CONTEXT_SPECIFIC** 2
- #define **ASN1_CLASS_PRIVATE** 3
- #define **ASN1_MAX_OID_LEN** 20

## Functions

- int **asn1_get_next** (const u8 *buf, size_t len, struct asn1_hdr *hdr)
- int **asn1_get_oid** (const u8 *buf, size_t len, struct asn1_oid *oid, const u8 **next)
- void **asn1_oid_to_str** (struct asn1_oid *oid, char *buf, size_t len)
- unsigned long **asn1_bit_string_to_long** (const u8 *buf, size_t len)

### 6.6.1 Detailed Description

ASN.1 DER parsing.

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

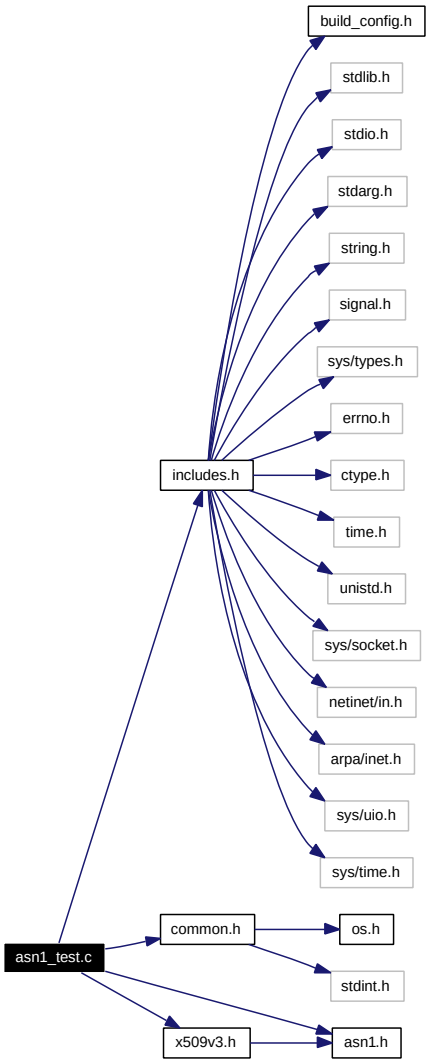
Definition in file asn1.h.

## 6.7 asn1_test.c File Reference

Testing tool for ASN.1/X.509v3 routines.

```
#include "includes.h"
#include "common.h"
#include "asn1.h"
#include "x509v3.h"
```

Include dependency graph for asn1_test.c:




### Functions

- int **asn1_parse** (const u8 *buf, size_t len, int level)
- int **main** (int argc, char *argv[ ])

## Variables

- int **wpa_debug_level**

### 6.7.1 Detailed Description

Testing tool for ASN.1/X.509v3 routines.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

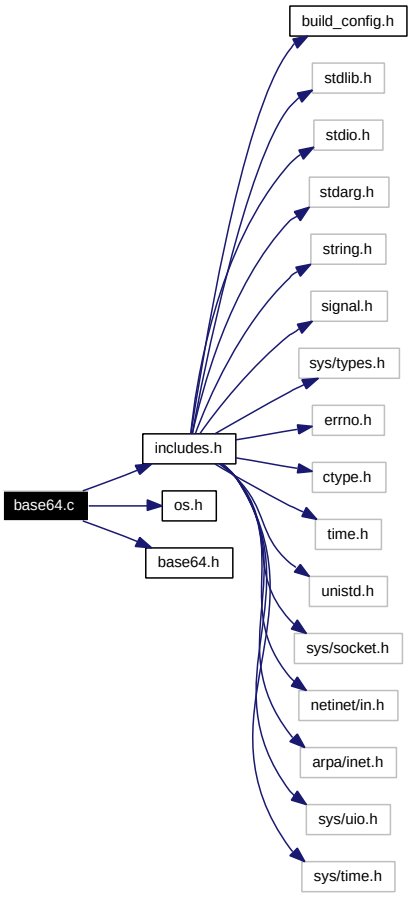
Definition in file asn1_test.c.

## 6.8 base64.c File Reference

Base64 encoding/decoding (RFC1341).

```
#include "includes.h"
#include "os.h"
#include "base64.h"
```

Include dependency graph for base64.c:




### Functions

- unsigned char * base64_encode (const unsigned char *src, size_t len, size_t *out_len)
  *Base64 encode.*
- unsigned char * base64_decode (const unsigned char *src, size_t len, size_t *out_len)
  *Base64 decode.*

### 6.8.1 Detailed Description

Base64 encoding/decoding (RFC1341).

**Copyright**
Copyright (c) 2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file base64.c.

## 6.8.2 Function Documentation

### 6.8.2.1 unsigned char* base64_decode (const unsigned char * *src*, size_t *len*, size_t * *out_len*)

Base64 decode.

**Parameters:**
***src*** Data to be decoded
***len*** Length of the data to be decoded
***out_len*** Pointer to output length variable

**Returns:**
Allocated buffer of out_len bytes of decoded data, or NULL on failure

Caller is responsible for freeing the returned buffer.

Definition at line 104 of file base64.c.

### 6.8.2.2 unsigned char* base64_encode (const unsigned char * *src*, size_t *len*, size_t * *out_len*)

Base64 encode.

**Parameters:**
***src*** Data to be encoded
***len*** Length of the data to be encoded
***out_len*** Pointer to output length variable, or NULL if not used

**Returns:**
Allocated buffer of out_len bytes of encoded data, or NULL on failure

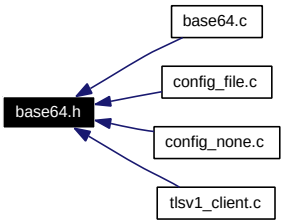
Caller is responsible for freeing the returned buffer. Returned buffer is nul terminated to make it easier to use as a C string. The nul terminator is not included in out_len.

Definition at line 37 of file base64.c.

## 6.9 base64.h File Reference

Base64 encoding/decoding (RFC1341).

This graph shows which files directly or indirectly include this file:




### Functions

- unsigned char * base64_encode (const unsigned char *src, size_t len, size_t *out_len)

  *Base64 encode.*

- unsigned char * base64_decode (const unsigned char *src, size_t len, size_t *out_len)

  *Base64 decode.*

### 6.9.1 Detailed Description

Base64 encoding/decoding (RFC1341).

**Copyright**
Copyright (c) 2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file base64.h.

### 6.9.2 Function Documentation

#### 6.9.2.1 unsigned char* base64_decode (const unsigned char * *src*, size_t *len*, size_t * *out_len*)

Base64 decode.

**Parameters:**

***src*** Data to be decoded

***len*** Length of the data to be decoded

***out_len*** Pointer to output length variable

**Returns:**
Allocated buffer of out_len bytes of decoded data, or NULL on failure

Caller is responsible for freeing the returned buffer.

Definition at line 104 of file base64.c.

#### 6.9.2.2 unsigned char* base64_encode (const unsigned char * *src*, size_t *len*, size_t * *out_len*)

Base64 encode.

**Parameters:**
***src*** Data to be encoded

***len*** Length of the data to be encoded

***out_len*** Pointer to output length variable, or NULL if not used

**Returns:**
Allocated buffer of out_len bytes of encoded data, or NULL on failure

Caller is responsible for freeing the returned buffer. Returned buffer is nul terminated to make it easier to use as a C string. The nul terminator is not included in out_len.

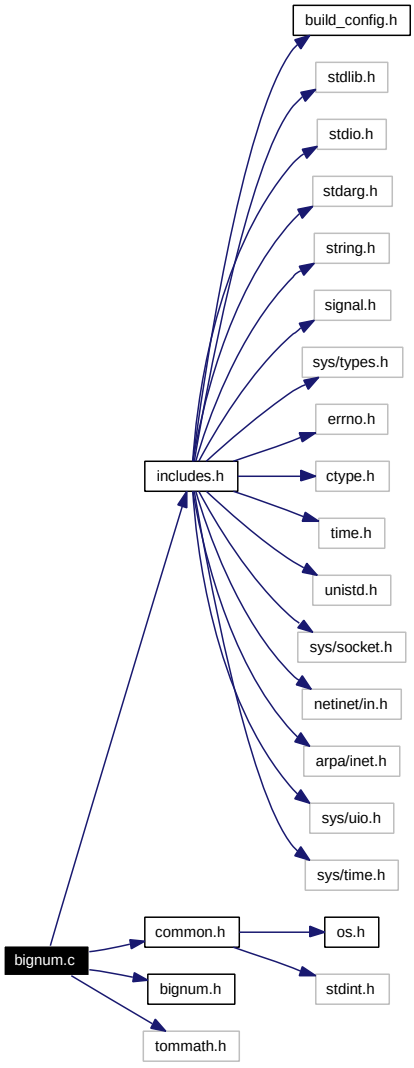
Definition at line 37 of file base64.c.

## 6.10 bignum.c File Reference

Big number math.

```
#include "includes.h"
#include "common.h"
#include "bignum.h"
#include <tommath.h>
```

Include dependency graph for bignum.c:




### Functions

- bignum * bignum_init (void)

  *Allocate memory for bignum.*

- void bignum_deinit (struct bignum *n)

*Free bignum.*

- size_t bignum_get_unsigned_bin_len (struct bignum *n)

  *Get length of bignum as an unsigned binary buffer.*

- int bignum_get_unsigned_bin (const struct bignum *n, u8 *buf, size_t *len)

  *Set binary buffer to unsigned bignum.*

- int bignum_set_unsigned_bin (struct bignum *n, const u8 *buf, size_t len)

  *Set bignum based on unsigned binary buffer.*

- int bignum_cmp (const struct bignum *a, const struct bignum *b)

  *Signed comparison.*

- int bignum_cmp_d (const struct bignum *a, unsigned long b)

  *Compare bignum to standard integer.*

- int bignum_add (const struct bignum *a, const struct bignum *b, struct bignum *c)

  *c = a + b*

- int bignum_sub (const struct bignum *a, const struct bignum *b, struct bignum *c)

  *c = a - b*

- int bignum_mul (const struct bignum *a, const struct bignum *b, struct bignum *c)

  *c = a * b*

- int bignum_mulmod (const struct bignum *a, const struct bignum *b, const struct bignum *c, struct bignum *d)

  *d = a * b (mod c)*

- int bignum_exptmod (const struct bignum *a, const struct bignum *b, const struct bignum *c, struct bignum *d)

  *Modular exponentiation: d = a^b (mod c).*

### 6.10.1 Detailed Description

Big number math.

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file bignum.c.

## 6.10.2 Function Documentation

### 6.10.2.1 int bignum_add (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a + b

**Parameters:**

***a*** Bignum from bignum_init()

***b*** Bignum from bignum_init()

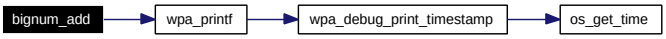
***c*** Bignum from bignum_init(); used to store the result of a + b

**Returns:**

0 on success, -1 on failure

Definition at line 155 of file bignum.c.

Here is the call graph for this function:




### 6.10.2.2 int bignum_cmp (const struct bignum * *a*, const struct bignum * *b*)

Signed comparison.

**Parameters:**

***a*** Bignum from bignum_init()

***b*** Bignum from bignum_init()

**Returns:**

0 on success, -1 on failure

Definition at line 128 of file bignum.c.

### 6.10.2.3 int bignum_cmp_d (const struct bignum * *a*, unsigned long *b*)

Compare bignum to standard integer.

**Parameters:**

***a*** Bignum from bignum_init()

***b*** Small integer

**Returns:**

0 on success, -1 on failure

Definition at line 141 of file bignum.c.

#### 6.10.2.4 void bignum_deinit (struct bignum * *n*)

Free bignum.

**Parameters:**
***n*** Bignum from bignum_init()

Definition at line 56 of file bignum.c.

#### 6.10.2.5 int bignum_exptmod (const struct bignum * *a*, const struct bignum * *b*, const struct bignum * *c*, struct bignum * *d*)

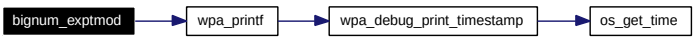
Modular exponentiation: d = a^b (mod c).

**Parameters:**
***a*** Bignum from bignum_init(); base
***b*** Bignum from bignum_init(); exponent
***c*** Bignum from bignum_init(); modulus
***d*** Bignum from bignum_init(); used to store the result of a^b (mod c)

**Returns:**
0 on success, -1 on failure

Definition at line 234 of file bignum.c.

Here is the call graph for this function:




#### 6.10.2.6 int bignum_get_unsigned_bin (const struct bignum * *n*, u8 * *buf*, size_t * *len*)

Set binary buffer to unsigned bignum.

**Parameters:**
***n*** Bignum from bignum_init()
***buf*** Buffer for the binary number
***len*** Length of the buffer, can be NULL if buffer is known to be long enough. Set to used buffer length on success if not NULL.

**Returns:**
0 on success, -1 on failure

Definition at line 86 of file bignum.c.

Here is the call graph for this function:

#### 6.10.2.7 size_t bignum_get_unsigned_bin_len (struct bignum * *n*)

Get length of bignum as an unsigned binary buffer.

**Parameters:**
***n*** Bignum from bignum_init()

**Returns:**
Length of n if written to a binary buffer

Definition at line 71 of file bignum.c.

#### 6.10.2.8 struct bignum* bignum_init (void)

Allocate memory for bignum.

**Returns:**
Pointer to allocated bignum or NULL on failure

Definition at line 38 of file bignum.c.

Here is the call graph for this function:


#### 6.10.2.9 int bignum_mul (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a * b

**Parameters:**
***a*** Bignum from bignum_init()

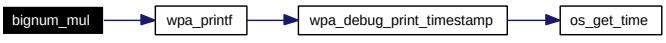
***b*** Bignum from bignum_init()

***c*** Bignum from bignum_init(); used to store the result of a * b

**Returns:**
0 on success, -1 on failure

Definition at line 193 of file bignum.c.

Here is the call graph for this function:



#### 6.10.2.10 int bignum_mulmod (const struct bignum * *a*, const struct bignum * *b*, const struct bignum * *c*, struct bignum * *d*)

d = a * b (mod c)

**Parameters:**

***a*** Bignum from bignum_init()

***b*** Bignum from bignum_init()

***c*** Bignum from bignum_init(); modulus

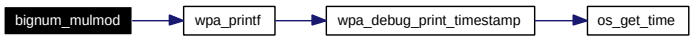
***d*** Bignum from bignum_init(); used to store the result of a * b (mod c)

**Returns:**

0 on success, -1 on failure

Definition at line 213 of file bignum.c.

Here is the call graph for this function:




#### 6.10.2.11 int bignum_set_unsigned_bin (struct bignum * *n*, const u8 * *buf*, size_t *len*)

Set bignum based on unsigned binary buffer.

**Parameters:**

***a*** Bignum from bignum_init(); to be set to the given value

***buf*** Buffer with unsigned binary value

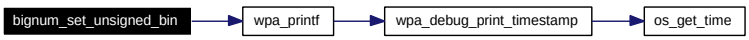
***len*** Length of buf in octets

**Returns:**

0 on success, -1 on failure

Definition at line 111 of file bignum.c.

Here is the call graph for this function:




#### 6.10.2.12 int bignum_sub (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a - b

**Parameters:**

***a*** Bignum from bignum_init()

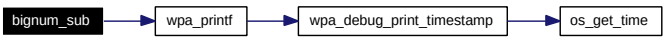
***b*** Bignum from bignum_init()

***c*** Bignum from bignum_init(); used to store the result of a - b

**Returns:**
0 on success, -1 on failure

Definition at line 174 of file bignum.c.

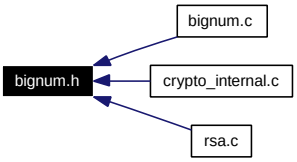
Here is the call graph for this function:



# 6.11 bignum.h File Reference

Big number math.

This graph shows which files directly or indirectly include this file:




## Functions

- bignum * bignum_init (void)
  *Allocate memory for bignum.*
- void bignum_deinit (struct bignum *n)
  *Free bignum.*
- size_t bignum_get_unsigned_bin_len (struct bignum *n)
  *Get length of bignum as an unsigned binary buffer.*
- int bignum_get_unsigned_bin (const struct bignum *n, u8 *buf, size_t *len)
  *Set binary buffer to unsigned bignum.*
- int bignum_set_unsigned_bin (struct bignum *n, const u8 *buf, size_t len)
  *Set bignum based on unsigned binary buffer.*
- int bignum_cmp (const struct bignum *a, const struct bignum *b)
  *Signed comparison.*
- int bignum_cmp_d (const struct bignum *a, unsigned long b)
  *Compare bignum to standard integer.*
- int bignum_add (const struct bignum *a, const struct bignum *b, struct bignum *c)
  *c = a + b*
- int bignum_sub (const struct bignum *a, const struct bignum *b, struct bignum *c)
  *c = a - b*
- int bignum_mul (const struct bignum *a, const struct bignum *b, struct bignum *c)
  *c = a * b*
- int bignum_mulmod (const struct bignum *a, const struct bignum *b, const struct bignum *c, struct bignum *d)
  *d = a * b (mod c)*
- int bignum_exptmod (const struct bignum *a, const struct bignum *b, const struct bignum *c, struct bignum *d)

*Modular exponentiation: d = a^b (mod c).*

### 6.11.1 Detailed Description

Big number math.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file bignum.h.

### 6.11.2 Function Documentation

#### 6.11.2.1 int bignum_add (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

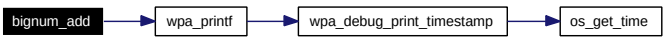
c = a + b

**Parameters:**
- ***a*** Bignum from bignum_init()
- ***b*** Bignum from bignum_init()
- ***c*** Bignum from bignum_init(); used to store the result of a + b

**Returns:**
0 on success, -1 on failure

Definition at line 155 of file bignum.c.

Here is the call graph for this function:




#### 6.11.2.2 int bignum_cmp (const struct bignum * *a*, const struct bignum * *b*)

Signed comparison.

**Parameters:**
- ***a*** Bignum from bignum_init()
- ***b*** Bignum from bignum_init()

**Returns:**
0 on success, -1 on failure

Definition at line 128 of file bignum.c.

#### 6.11.2.3 int bignum_cmp_d (const struct bignum * *a*, unsigned long *b*)

Compare bignum to standard integer.

**Parameters:**
- ***a*** Bignum from bignum_init()
- ***b*** Small integer

**Returns:**
0 on success, -1 on failure

Definition at line 141 of file bignum.c.

#### 6.11.2.4 void bignum_deinit (struct bignum * *n*)

Free bignum.

**Parameters:**
- ***n*** Bignum from bignum_init()

Definition at line 56 of file bignum.c.

#### 6.11.2.5 int bignum_exptmod (const struct bignum * *a*, const struct bignum * *b*, const struct bignum * *c*, struct bignum * *d*)

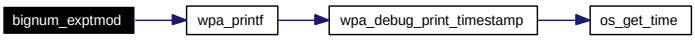
Modular exponentiation: d = a^b (mod c).

**Parameters:**
- ***a*** Bignum from bignum_init(); base
- ***b*** Bignum from bignum_init(); exponent
- ***c*** Bignum from bignum_init(); modulus
- ***d*** Bignum from bignum_init(); used to store the result of a^b (mod c)

**Returns:**
0 on success, -1 on failure

Definition at line 234 of file bignum.c.

Here is the call graph for this function:




#### 6.11.2.6 int bignum_get_unsigned_bin (const struct bignum * *n*, u8 * *buf*, size_t * *len*)

Set binary buffer to unsigned bignum.

**Parameters:**
- ***n*** Bignum from bignum_init()

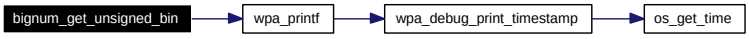
***buf*** Buffer for the binary number

***len*** Length of the buffer, can be NULL if buffer is known to be long enough. Set to used buffer length on success if not NULL.

**Returns:**
0 on success, -1 on failure

Definition at line 86 of file bignum.c.

Here is the call graph for this function:




#### 6.11.2.7 size_t bignum_get_unsigned_bin_len (struct bignum * *n*)

Get length of bignum as an unsigned binary buffer.

**Parameters:**
***n*** Bignum from bignum_init()

**Returns:**
Length of n if written to a binary buffer

Definition at line 71 of file bignum.c.

#### 6.11.2.8 struct bignum* bignum_init (void)

Allocate memory for bignum.

**Returns:**
Pointer to allocated bignum or NULL on failure

Definition at line 38 of file bignum.c.

Here is the call graph for this function:


#### 6.11.2.9 int bignum_mul (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a * b

**Parameters:**
***a*** Bignum from bignum_init()

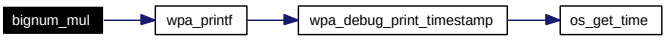
***b*** Bignum from bignum_init()

***c*** Bignum from bignum_init(); used to store the result of a * b

**Returns:**
0 on success, -1 on failure

Definition at line 193 of file bignum.c.

Here is the call graph for this function:




#### 6.11.2.10 int bignum_mulmod (const struct bignum * *a*, const struct bignum * *b*, const struct bignum * *c*, struct bignum * *d*)

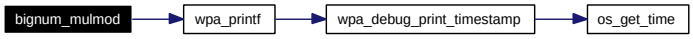
d = a * b (mod c)

**Parameters:**
***a*** Bignum from bignum_init()
***b*** Bignum from bignum_init()
***c*** Bignum from bignum_init(); modulus
***d*** Bignum from bignum_init(); used to store the result of a * b (mod c)

**Returns:**
0 on success, -1 on failure

Definition at line 213 of file bignum.c.

Here is the call graph for this function:




#### 6.11.2.11 int bignum_set_unsigned_bin (struct bignum * *n*, const u8 * *buf*, size_t *len*)

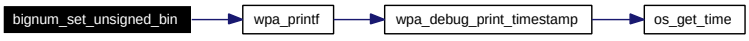
Set bignum based on unsigned binary buffer.

**Parameters:**
***a*** Bignum from bignum_init(); to be set to the given value
***buf*** Buffer with unsigned binary value
***len*** Length of buf in octets

**Returns:**
0 on success, -1 on failure

Definition at line 111 of file bignum.c.

Here is the call graph for this function:



#### 6.11.2.12 int bignum_sub (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a - b

**Parameters:**

***a*** Bignum from bignum_init()

***b*** Bignum from bignum_init()

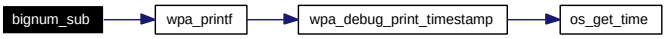
***c*** Bignum from bignum_init(); used to store the result of a - b

**Returns:**

0 on success, -1 on failure

Definition at line 174 of file bignum.c.

Here is the call graph for this function:



## 6.12 build_config.h File Reference

wpa_supplicant/hostapd - Build time configuration defines

This graph shows which files directly or indirectly include this file:


### 6.12.1 Detailed Description

wpa_supplicant/hostapd - Build time configuration defines

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This header file can be used to define configuration defines that were originally defined in Makefile. This is mainly meant for IDE use or for systems that do not have suitable 'make' tool. In these cases, it may be easier to have a single place for defining all the needed C pre-processor defines.

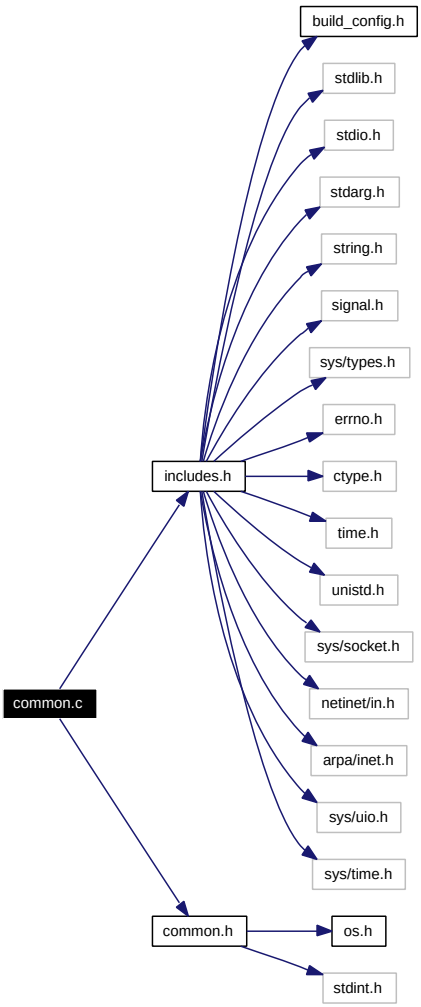
Definition in file build_config.h.

## 6.13 common.c File Reference

wpa_supplicant/hostapd / common helper functions, etc.

```
#include "includes.h"
#include "common.h"
```

Include dependency graph for common.c:




### Functions

- int hwaddr_aton (const char *txt, u8 *addr)

  *Convert ASCII string to MAC address.*

- int hexstr2bin (const char *hex, u8 *buf, size_t len)

  *Convert ASCII hex string into binary data.*

- void inc_byte_array (u8 *counter, size_t len)

  *Increment arbitrary length byte array by one.*

- void **wpa_get_ntp_timestamp** (u8 *buf)
- void wpa_debug_print_timestamp (void)

  *Print timestamp for debug output.*

- void wpa_printf (int level, char *fmt,...)

  *conditional printf*

- void wpa_hexdump (int level, const char *title, const u8 *buf, size_t len)

  *conditional hex dump*

- void wpa_hexdump_key (int level, const char *title, const u8 *buf, size_t len)

  *conditional hex dump, hide keys*

- void wpa_hexdump_ascii (int level, const char *title, const u8 *buf, size_t len)

  *conditional hex dump*

- void wpa_hexdump_ascii_key (int level, const char *title, const u8 *buf, size_t len)

  *conditional hex dump, hide keys*

- int **wpa_debug_open_file** (void)
- void **wpa_debug_close_file** (void)
- void wpa_msg_register_cb (wpa_msg_cb_func func)

  *Register callback function for wpa_msg() messages.*

- void **wpa_msg** (void *ctx, int level, char *fmt,...)
- int wpa_snprintf_hex (char *buf, size_t buf_size, const u8 *data, size_t len)

  *Print data as a hex string into a buffer.*

- int wpa_snprintf_hex_uppercase (char *buf, size_t buf_size, const u8 *data, size_t len)

  *Print data as a upper case hex string into buf.*

- const char * wpa_ssid_txt (u8 *ssid, size_t ssid_len)

  *Convert SSID to a printable string.*

## Variables

- int **wpa_debug_use_file** = 0
- int **wpa_debug_level** = MSG_INFO
- int **wpa_debug_show_keys** = 0
- int **wpa_debug_timestamp** = 0

### 6.13.1 Detailed Description

wpa_supplicant/hostapd / common helper functions, etc.

**Copyright**

Copyright (c) 2002-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file common.c.

### 6.13.2 Function Documentation

#### 6.13.2.1 int hexstr2bin (const char * *hex*, u8 * *buf*, size_t *len*)

Convert ASCII hex string into binary data.

**Parameters:**

***hex*** ASCII hex string (e.g., "01ab")

***buf*** Buffer for the binary data

***len*** Length of the text to convert in bytes (of buf); hex will be double this size

**Returns:**

0 on success, -1 on failure (invalid hex string)

Definition at line 93 of file common.c.

#### 6.13.2.2 int hwaddr_aton (const char * *txt*, u8 * *addr*)

Convert ASCII string to MAC address.

**Parameters:**

***txt*** MAC address as a string (e.g., "00:11:22:33:44:55")

***addr*** Buffer for the MAC address (ETH_ALEN = 6 bytes)

**Returns:**

0 on success, -1 on failure (e.g., string not a MAC address)

Definition at line 62 of file common.c.

#### 6.13.2.3 void inc_byte_array (u8 * *counter*, size_t *len*)

Increment arbitrary length byte array by one.

**Parameters:**

***counter*** Pointer to byte array

***len*** Length of the counter in bytes

This function increments the last byte of the counter by one and continues rolling over to more significant bytes if the byte was incremented from 0xff to 0x00.

Definition at line 121 of file common.c.

#### 6.13.2.4 void wpa_debug_print_timestamp (void)

Print timestamp for debug output.

This function prints a timestamp in <seconds from 1970>.<microsoconds> format if debug output has been configured to include timestamps in debug messages.

Definition at line 152 of file common.c.

Here is the call graph for this function:


#### 6.13.2.5 void wpa_hexdump (int *level*, const char * *title*, const u8 * *buf*, size_t *len*)

conditional hex dump

**Parameters:**

***level*** priority level (MSG_*) of the message

***title*** title of for the message

***buf*** data buffer to be dumped

***len*** length of the buf

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. The contents of buf is printed out has hex dump.

Definition at line 242 of file common.c.

#### 6.13.2.6 void wpa_hexdump_ascii (int *level*, const char * *title*, const u8 * *buf*, size_t *len*)

conditional hex dump

**Parameters:**

***level*** priority level (MSG_*) of the message

***title*** title of for the message

***buf*** data buffer to be dumped

***len*** length of the buf

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. The contents of buf is printed out has hex dump with both the hex numbers and ASCII characters (for printable range) are shown. 16 bytes per line will be shown.

Definition at line 339 of file common.c.

#### 6.13.2.7 void wpa_hexdump_ascii_key (int *level*, const char * *title*, const u8 * *buf*, size_t *len*)

conditional hex dump, hide keys

**Parameters:**

***level*** priority level (MSG_∗) of the message

***title*** title of for the message

***buf*** data buffer to be dumped

***len*** length of the buf

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. The contents of buf is printed out has hex dump with both the hex numbers and ASCII characters (for printable range) are shown. 16 bytes per line will be shown. This works like wpa_hexdump_ascii(), but by default, does not include secret keys (passwords, etc.) in debug output.

Definition at line 345 of file common.c.

#### 6.13.2.8 void wpa_hexdump_key (int *level*, const char ∗ *title*, const u8 ∗ *buf*, size_t *len*)

conditional hex dump, hide keys

**Parameters:**

***level*** priority level (MSG_∗) of the message

***title*** title of for the message

***buf*** data buffer to be dumped

***len*** length of the buf

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. The contents of buf is printed out has hex dump. This works like wpa_hexdump(), but by default, does not include secret keys (passwords, etc.) in debug output.

Definition at line 248 of file common.c.

#### 6.13.2.9 void wpa_msg_register_cb (wpa_msg_cb_func *func*)

Register callback function for wpa_msg() messages.

**Parameters:**

***func*** Callback function (NULL to unregister)

Definition at line 390 of file common.c.

#### 6.13.2.10 void wpa_printf (int *level*, char ∗ *fmt*, ...)

conditional printf

**Parameters:**

***level*** priority level (MSG_∗) of the message

***fmt*** printf format string, followed by optional arguments

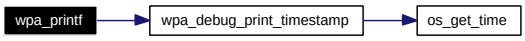
This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration.

Note: New line ’

’ is added to the end of the text when printing to stdout.

Definition at line 182 of file common.c.

Here is the call graph for this function:




#### 6.13.2.11 int wpa_snprintf_hex (char * *buf*, size_t *buf_size*, const u8 * *data*, size_t *len*)

Print data as a hex string into a buffer.

**Parameters:**

***buf*** Memory area to use as the output buffer

***buf_size*** Maximum buffer size in bytes (should be at least 2 * len + 1)

***data*** Data to be printed

***len*** Length of data in bytes

**Returns:**

Number of bytes written

Definition at line 450 of file common.c.

#### 6.13.2.12 int wpa_snprintf_hex_uppercase (char * *buf*, size_t *buf_size*, const u8 * *data*, size_t *len*)

Print data as a upper case hex string into buf.

**Parameters:**

***buf*** Memory area to use as the output buffer

***buf_size*** Maximum buffer size in bytes (should be at least 2 * len + 1)

***data*** Data to be printed

***len*** Length of data in bytes

**Returns:**

Number of bytes written

Definition at line 465 of file common.c.

#### 6.13.2.13 const char* wpa_ssid_txt (u8 * *ssid*, size_t *ssid_len*)

Convert SSID to a printable string.

**Parameters:**

***ssid*** SSID (32-octet string)

***ssid_len*** Length of ssid in octets

**Returns:**
Pointer to a printable string

This function can be used to convert SSIDs into printable form. In most cases, SSIDs do not use unprintable characters, but IEEE 802.11 standard does not limit the used character set, so anything could be used in an SSID.

This function uses a static buffer, so only one call can be used at the time, i.e., this is not re-entrant and the returned buffer must be used before calling this again.

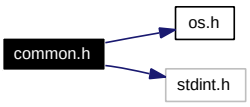
Definition at line 598 of file common.c.

## 6.14 common.h File Reference

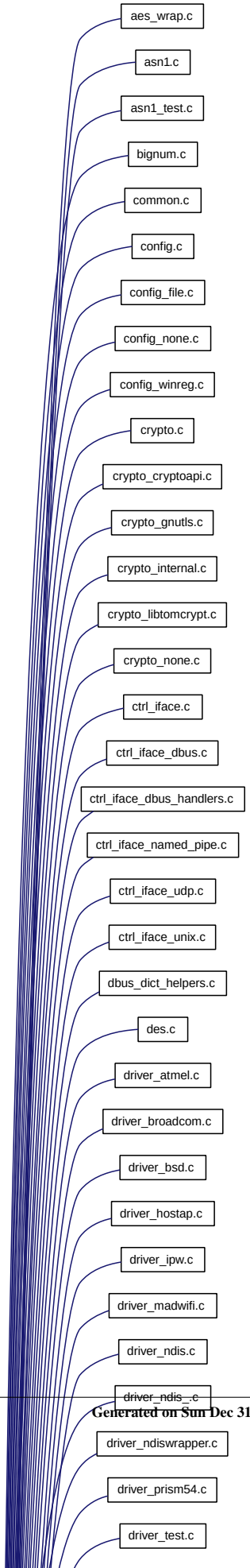
wpa_supplicant/hostapd / common helper functions, etc.

```
#include "os.h"
#include <stdint.h>
```

Include dependency graph for common.h:




This graph shows which files directly or indirectly include this file:


aes_wrap.c
asn1.c
asn1_test.c
bignum.c
common.c
config.c
config_file.c
config_none.c
config_winreg.c
crypto.c
crypto_cryptoapi.c
crypto_gnutls.c
crypto_internal.c
crypto_libtomcrypt.c
crypto_none.c
ctrl_iface.c
ctrl_iface_dbus.c
ctrl_iface_dbus_handlers.c
ctrl_iface_named_pipe.c
ctrl_iface_udp.c
ctrl_iface_unix.c
dbus_dict_helpers.c
des.c
driver_atmel.c
driver_broadcom.c
driver_bsd.c
driver_hostap.c
driver_ipw.c
driver_madwifi.c
driver_ndis.c
driver_ndis_.c
driver_ndiswrapper.c
driver_prism54.c
driver_test.c

## Defines

- #define **__LITTLE_ENDIAN** 1234
- #define **__BIG_ENDIAN** 4321
- #define **WPA_GET_BE16**(a) ((u16) (((a)[0] << 8) | (a)[1]))
- #define **WPA_PUT_BE16**(a, val)
- #define **WPA_GET_LE16**(a) ((u16) (((a)[1] << 8) | (a)[0]))
- #define **WPA_PUT_LE16**(a, val)
- #define **WPA_GET_BE24**(a)
- #define **WPA_PUT_BE24**(a, val)
- #define **WPA_GET_BE32**(a)
- #define **WPA_PUT_BE32**(a, val)
- #define **WPA_PUT_BE64**(a, val)
- #define **ETH_ALEN** 6
- #define **WPA_TYPES_DEFINED**
- #define **hostapd_get_rand** os_get_random
- #define **PRINTF_FORMAT**(a, b)
- #define **STRUCT_PACKED**
- #define **WPA_ASSERT**(a) do { } while (0)
- #define **wpa_zalloc**(s) os_zalloc((s))
- #define **wpa_unicode2ascii_inplace**(s) do { } while (0)
- #define **wpa_strdup_tchar**(s) strdup((s))

## Typedefs

- typedef uint64_t **u64**
- typedef uint32_t **u32**
- typedef uint16_t **u16**
- typedef uint8_t **u8**
- typedef int64_t **s64**
- typedef int32_t **s32**
- typedef int16_t **s16**
- typedef int8_t **s8**
- typedef u32 **__be32**
- typedef u64 **__be64**

## Enumerations

- enum {

  **MSG_MSGDUMP**, **MSG_DEBUG**, **MSG_INFO**, **MSG_WARNING**,

  **MSG_ERROR** }

## Functions

- int hwaddr_aton (const char *txt, u8 *addr)
  *Convert ASCII string to MAC address.*
- int hexstr2bin (const char *hex, u8 *buf, size_t len)
  *Convert ASCII hex string into binary data.*
- void inc_byte_array (u8 *counter, size_t len)
  *Increment arbitrary length byte array by one.*
- void **wpa_get_ntp_timestamp** (u8 *buf)
- int **wpa_debug_open_file** (void)
- void **wpa_debug_close_file** (void)
- void wpa_debug_print_timestamp (void)
  *Print timestamp for debug output.*
- void wpa_printf (int level, char *fmt,...) PRINTF_FORMAT(2
  *conditional printf*
- void void wpa_hexdump (int level, const char *title, const u8 *buf, size_t len)
  *conditional hex dump*
- void wpa_hexdump_key (int level, const char *title, const u8 *buf, size_t len)
  *conditional hex dump, hide keys*
- void wpa_hexdump_ascii (int level, const char *title, const u8 *buf, size_t len)
  *conditional hex dump*
- void wpa_hexdump_ascii_key (int level, const char *title, const u8 *buf, size_t len)
  *conditional hex dump, hide keys*
- void wpa_msg (void *ctx, int level, char *fmt,...) PRINTF_FORMAT(3
  *Conditional printf for default target and ctrl_iface monitors.*
- void wpa_msg_register_cb (wpa_msg_cb_func func)
  *Register callback function for wpa_msg() messages.*
- int wpa_snprintf_hex (char *buf, size_t buf_size, const u8 *data, size_t len)
  *Print data as a hex string into a buffer.*
- int wpa_snprintf_hex_uppercase (char *buf, size_t buf_size, const u8 *data, size_t len)
  *Print data as a upper case hex string into buf.*
- const char * wpa_ssid_txt (u8 *ssid, size_t ssid_len)
  *Convert SSID to a printable string.*

## Variables

- void typedef void(* **wpa_msg_cb_func** )(void *ctx, int level, const char *txt, size_t len)

### 6.14.1 Detailed Description

wpa_supplicant/hostapd / common helper functions, etc.

**Copyright**
Copyright (c) 2002-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file common.h.

### 6.14.2 Define Documentation

#### 6.14.2.1 #define WPA_GET_BE24(a)

**Value:**

```
((((u32) (a)[0]) << 16) | (((u32) (a)[1]) << 8) | \
                          ((u32) (a)[2]))
```

Definition at line 146 of file common.h.

#### 6.14.2.2 #define WPA_GET_BE32(a)

**Value:**

```
((((u32) (a)[0]) << 24) | (((u32) (a)[1]) << 16) | \
                          (((u32) (a)[2]) << 8) | ((u32) (a)[3]))
```

Definition at line 155 of file common.h.

#### 6.14.2.3 #define WPA_PUT_BE16(a, val)

**Value:**

```
do {                                    \
                (a)[0] = ((u16) (val)) >> 8;    \
                (a)[1] = ((u16) (val)) & 0xff;  \
        } while (0)
```

Definition at line 133 of file common.h.

#### 6.14.2.4 #define WPA_PUT_BE24(a, val)

**Value:**

```
do {                                            \
                (a)[0] = (u8) (((u32) (val)) >> 16);    \
                (a)[1] = (u8) (((u32) (val)) >> 8);     \
                (a)[2] = (u8) (((u32) (val)) & 0xff);   \
        } while (0)
```

Definition at line 148 of file common.h.

#### 6.14.2.5 #define WPA_PUT_BE32(a, val)

**Value:**

```
do {                                            \
                (a)[0] = (u8) (((u32) (val)) >> 24);    \
                (a)[1] = (u8) (((u32) (val)) >> 16);    \
                (a)[2] = (u8) (((u32) (val)) >> 8);     \
                (a)[3] = (u8) (((u32) (val)) & 0xff);   \
        } while (0)
```

Definition at line 157 of file common.h.

#### 6.14.2.6 #define WPA_PUT_BE64(a, val)

**Value:**

```
do {                                            \
                (a)[0] = (u8) (((u64) (val)) >> 56);    \
                (a)[1] = (u8) (((u64) (val)) >> 48);    \
                (a)[2] = (u8) (((u64) (val)) >> 40);    \
                (a)[3] = (u8) (((u64) (val)) >> 32);    \
                (a)[4] = (u8) (((u64) (val)) >> 24);    \
                (a)[5] = (u8) (((u64) (val)) >> 16);    \
                (a)[6] = (u8) (((u64) (val)) >> 8);     \
                (a)[7] = (u8) (((u64) (val)) & 0xff);   \
        } while (0)
```

Definition at line 165 of file common.h.

#### 6.14.2.7 #define WPA_PUT_LE16(a, val)

**Value:**

```
do {                                    \
                (a)[1] = ((u16) (val)) >> 8;    \
                (a)[0] = ((u16) (val)) & 0xff;  \
        } while (0)
```

Definition at line 140 of file common.h.

### 6.14.3 Function Documentation

#### 6.14.3.1 int hexstr2bin (const char * *hex*, u8 * *buf*, size_t *len*)

Convert ASCII hex string into binary data.

**Parameters:**

***hex*** ASCII hex string (e.g., "01ab")

***buf*** Buffer for the binary data

***len*** Length of the text to convert in bytes (of buf); hex will be double this size

**Returns:**

0 on success, -1 on failure (invalid hex string)

Definition at line 93 of file common.c.

#### 6.14.3.2 int hwaddr_aton (const char * *txt*, u8 * *addr*)

Convert ASCII string to MAC address.

**Parameters:**

***txt*** MAC address as a string (e.g., "00:11:22:33:44:55")

***addr*** Buffer for the MAC address (ETH_ALEN = 6 bytes)

**Returns:**

0 on success, -1 on failure (e.g., string not a MAC address)

Definition at line 62 of file common.c.

#### 6.14.3.3 void inc_byte_array (u8 * *counter*, size_t *len*)

Increment arbitrary length byte array by one.

**Parameters:**

***counter*** Pointer to byte array

***len*** Length of the counter in bytes

This function increments the last byte of the counter by one and continues rolling over to more significant bytes if the byte was incremented from 0xff to 0x00.

Definition at line 121 of file common.c.

#### 6.14.3.4 void wpa_debug_print_timestamp (void)

Print timestamp for debug output.

This function prints a timestamp in <seconds from 1970>.<microsoconds> format if debug output has been configured to include timestamps in debug messages.

Definition at line 152 of file common.c.

Here is the call graph for this function:

#### 6.14.3.5 void void wpa_hexdump (int *level*, const char * *title*, const u8 * *buf*, size_t *len*)

conditional hex dump

**Parameters:**

***level*** priority level (MSG_*) of the message

***title*** title of for the message

***buf*** data buffer to be dumped

***len*** length of the buf

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. The contents of buf is printed out has hex dump.

Definition at line 242 of file common.c.

#### 6.14.3.6 void wpa_hexdump_ascii (int *level*, const char * *title*, const u8 * *buf*, size_t *len*)

conditional hex dump

**Parameters:**

***level*** priority level (MSG_*) of the message

***title*** title of for the message

***buf*** data buffer to be dumped

***len*** length of the buf

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. The contents of buf is printed out has hex dump with both the hex numbers and ASCII characters (for printable range) are shown. 16 bytes per line will be shown.

Definition at line 339 of file common.c.

#### 6.14.3.7 void wpa_hexdump_ascii_key (int *level*, const char * *title*, const u8 * *buf*, size_t *len*)

conditional hex dump, hide keys

**Parameters:**

***level*** priority level (MSG_*) of the message

***title*** title of for the message

***buf*** data buffer to be dumped

***len*** length of the buf

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. The contents of buf is printed out has hex dump with both the hex numbers and ASCII characters (for printable range) are shown. 16 bytes per line will be shown. This works like wpa_hexdump_ascii(), but by default, does not include secret keys (passwords, etc.) in debug output.

Definition at line 345 of file common.c.

#### 6.14.3.8 void wpa_hexdump_key (int *level*, const char * *title*, const u8 * *buf*, size_t *len*)

conditional hex dump, hide keys

**Parameters:**

***level*** priority level (MSG_*) of the message

***title*** title of for the message

***buf*** data buffer to be dumped

***len*** length of the buf

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. The contents of buf is printed out has hex dump. This works like wpa_hexdump(), but by default, does not include secret keys (passwords, etc.) in debug output.

Definition at line 248 of file common.c.

#### 6.14.3.9 void wpa_msg (void * *ctx*, int *level*, char * *fmt*, ...)

Conditional printf for default target and ctrl_iface monitors.

**Parameters:**

***ctx*** Pointer to context data; this is the ctx variable registered with struct wpa_driver_ops::init()

***level*** priority level (MSG_*) of the message

***fmt*** printf format string, followed by optional arguments

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. This function is like wpa_printf(), but it also sends the same message to all attached ctrl_iface monitors.

Note: New line ’

’ is added to the end of the text when printing to stdout.

#### 6.14.3.10 void wpa_msg_register_cb (wpa_msg_cb_func *func*)

Register callback function for wpa_msg() messages.

**Parameters:**

***func*** Callback function (NULL to unregister)

Definition at line 390 of file common.c.

#### 6.14.3.11 void wpa_printf (int *level*, char * *fmt*, ...)

conditional printf

**Parameters:**

***level*** priority level (MSG_*) of the message

***fmt*** printf format string, followed by optional arguments

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration.

Note: New line '

' is added to the end of the text when printing to stdout.

#### 6.14.3.12 int wpa_snprintf_hex (char * *buf*, size_t *buf_size*, const u8 * *data*, size_t *len*)

Print data as a hex string into a buffer.

**Parameters:**

***buf*** Memory area to use as the output buffer

***buf_size*** Maximum buffer size in bytes (should be at least 2 * len + 1)

***data*** Data to be printed

***len*** Length of data in bytes

**Returns:**

Number of bytes written

Definition at line 450 of file common.c.

#### 6.14.3.13 int wpa_snprintf_hex_uppercase (char * *buf*, size_t *buf_size*, const u8 * *data*, size_t *len*)

Print data as a upper case hex string into buf.

**Parameters:**

***buf*** Memory area to use as the output buffer

***buf_size*** Maximum buffer size in bytes (should be at least 2 * len + 1)

***data*** Data to be printed

***len*** Length of data in bytes

**Returns:**

Number of bytes written

Definition at line 465 of file common.c.

#### 6.14.3.14 const char* wpa_ssid_txt (u8 * *ssid*, size_t *ssid_len*)

Convert SSID to a printable string.

**Parameters:**

***ssid*** SSID (32-octet string)

***ssid_len*** Length of ssid in octets

**Returns:**

Pointer to a printable string

This function can be used to convert SSIDs into printable form. In most cases, SSIDs do not use unprintable characters, but IEEE 802.11 standard does not limit the used character set, so anything could be used in an SSID.

This function uses a static buffer, so only one call can be used at the time, i.e., this is not re-entrant and the returned buffer must be used before calling this again.

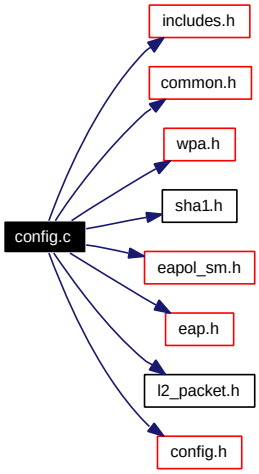
Definition at line 598 of file common.c.

## 6.15 config.c File Reference

WPA Supplicant / Configuration parser and common functions.

```
#include "includes.h"
#include "common.h"
#include "wpa.h"
#include "sha1.h"
#include "eapol_sm.h"
#include "eap.h"
#include "l2_packet.h"
#include "config.h"
```

Include dependency graph for config.c:




### Defines

- #define **OFFSET**(v) ((void ∗) &((struct wpa_ssid ∗) 0) → v)
- #define **_STR**(f) #f, wpa_config_parse_str, wpa_config_write_str, OFFSET(f)
- #define **STR**(f) _STR(f), NULL, NULL, NULL, 0
- #define **STR_KEY**(f) _STR(f), NULL, NULL, NULL, 1
- #define **_STR_LEN**(f) _STR(f), OFFSET(f ## _len)
- #define **STR_LEN**(f) _STR_LEN(f), NULL, NULL, 0
- #define **STR_LEN_KEY**(f) _STR_LEN(f), NULL, NULL, 1
- #define **_STR_RANGE**(f, min, max) _STR_LEN(f), (void ∗) (min), (void ∗) (max)
- #define **STR_RANGE**(f, min, max) _STR_RANGE(f, min, max), 0
- #define **STR_RANGE_KEY**(f, min, max) _STR_RANGE(f, min, max), 1
- #define **_INT**(f)
- #define **INT**(f) _INT(f), NULL, NULL, 0
- #define **INT_RANGE**(f, min, max) _INT(f), (void ∗) (min), (void ∗) (max), 0
- #define **_FUNC**(f)
- #define **FUNC**(f) _FUNC(f), 0

- #define **FUNC_KEY**(f) _FUNC(f), 1
- #define **NUM_SSID_FIELDS** (sizeof(ssid_fields) / sizeof(ssid_fields[0]))

## Functions

- int wpa_config_add_prio_network (struct wpa_config *config, struct wpa_ssid *ssid)
  *Add a network to priority lists.*
- void wpa_config_free_ssid (struct wpa_ssid *ssid)
  *Free network/ssid configuration data.*
- void wpa_config_free (struct wpa_config *config)
  *Free configuration data.*
- int wpa_config_allowed_eap_method (struct wpa_ssid *ssid, int vendor, u32 method)
  *Check whether EAP method is allowed.*
- wpa_ssid * wpa_config_get_network (struct wpa_config *config, int id)
  *Get configured network based on id.*
- wpa_ssid * wpa_config_add_network (struct wpa_config *config)
  *Add a new network with empty configuration.*
- int wpa_config_remove_network (struct wpa_config *config, int id)
  *Remove a configured network based on id.*
- void wpa_config_set_network_defaults (struct wpa_ssid *ssid)
  *Set network default values.*
- int wpa_config_set (struct wpa_ssid *ssid, const char *var, const char *value, int line)
  *Set a variable in network configuration.*
- char * wpa_config_get (struct wpa_ssid *ssid, const char *var)
  *Get a variable in network configuration.*
- char * wpa_config_get_no_key (struct wpa_ssid *ssid, const char *var)
  *Get a variable in network configuration (no keys).*
- void wpa_config_update_psk (struct wpa_ssid *ssid)
  *Update WPA PSK based on passphrase and SSID.*
- const struct wpa_config_blob * wpa_config_get_blob (struct wpa_config *config, const char *name)
  *Get a named configuration blob.*
- void wpa_config_set_blob (struct wpa_config *config, struct wpa_config_blob *blob)
  *Set or add a named configuration blob.*
- void wpa_config_free_blob (struct wpa_config_blob *blob)

*Free blob data.*

- int wpa_config_remove_blob (struct wpa_config *config, const char *name)

  *Remove a named configuration blob.*

- wpa_config * wpa_config_alloc_empty (const char *ctrl_interface, const char *driver_param)

  *Allocate an empty configuration.*

- void wpa_config_debug_dump_networks (struct wpa_config *config)

  *Debug dump of configured networks.*

### 6.15.1 Detailed Description

WPA Supplicant / Configuration parser and common functions.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file config.c.

### 6.15.2 Define Documentation

#### 6.15.2.1 #define _FUNC(f)

**Value:**

```
#f, wpa_config_parse_ ## f, wpa_config_write_ ## f, \
        NULL, NULL, NULL, NULL
```

Definition at line 1072 of file config.c.

#### 6.15.2.2 #define _INT(f)

**Value:**

```
#f, wpa_config_parse_int, wpa_config_write_int, \
        OFFSET(f), (void *) 0
```

Definition at line 1061 of file config.c.

### 6.15.3 Function Documentation

#### 6.15.3.1 struct wpa_ssid* wpa_config_add_network (struct wpa_config * *config*)

Add a new network with empty configuration.

**Parameters:**
***config*** Configuration data from wpa_config_read()

**Returns:**
The new network configuration or NULL if operation failed

Definition at line 1413 of file config.c.

Here is the call graph for this function:


#### 6.15.3.2 int wpa_config_add_prio_network (struct wpa_config * *config*, struct wpa_ssid * *ssid*)

Add a network to priority lists.

**Parameters:**
***config*** Configuration data from wpa_config_read()

***ssid*** Pointer to the network configuration to be added to the list

**Returns:**
0 on success, -1 on failure

This function is used to add a network block to the priority list of networks. This must be called for each network when reading in the full configuration. In addition, this can be used indirectly when updating priorities by calling wpa_config_update_prio_list().

Definition at line 1194 of file config.c.

#### 6.15.3.3 struct wpa_config* wpa_config_alloc_empty (const char * *ctrl_interface*, const char * *driver_param*)

Allocate an empty configuration.

**Parameters:**
***ctrl_interface*** Control interface parameters, e.g., path to UNIX domain socket

***driver_param*** Driver parameters

**Returns:**
Pointer to allocated configuration data or NULL on failure

Definition at line 1731 of file config.c.

Here is the call graph for this function:


#### 6.15.3.4 int wpa_config_allowed_eap_method (struct wpa_ssid * *ssid*, int *vendor*, u32 *method*)

Check whether EAP method is allowed.

**Parameters:**

***ssid*** Pointer to configuration data

***vendor*** Vendor-Id for expanded types or 0 = IETF for legacy types

***method*** EAP type

**Returns:**

1 = allowed EAP method, 0 = not allowed

Definition at line 1365 of file config.c.

#### 6.15.3.5 void wpa_config_debug_dump_networks (struct wpa_config * *config*)

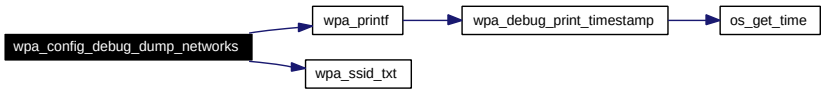
Debug dump of configured networks.

**Parameters:**

***config*** Configuration data from wpa_config_read()

Definition at line 1758 of file config.c.

Here is the call graph for this function:




#### 6.15.3.6 void wpa_config_free (struct wpa_config * *config*)

Free configuration data.

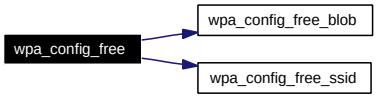
**Parameters:**

***config*** Configuration data from wpa_config_read()

This function frees all resources allocated for the configuration data by wpa_config_read().

Definition at line 1326 of file config.c.

Here is the call graph for this function:




#### 6.15.3.7 void wpa_config_free_blob (struct wpa_config_blob * *blob*)

Free blob data.

**Parameters:**
***blob*** Pointer to blob to be freed

Definition at line 1685 of file config.c.

#### 6.15.3.8 void wpa_config_free_ssid (struct wpa_ssid * *ssid*)

Free network/ssid configuration data.

**Parameters:**
***ssid*** Configuration data for the network

This function frees all resources allocated for the network configuration data.

Definition at line 1275 of file config.c.

#### 6.15.3.9 char* wpa_config_get (struct wpa_ssid * *ssid*, const char * *var*)

Get a variable in network configuration.

**Parameters:**
***ssid*** Pointer to network configuration data

***var*** Variable name, e.g., "ssid"

**Returns:**
Value of the variable or NULL on failure

This function can be used to get network configuration variables. The returned value is a copy of the configuration variable in text format, i.e,. the same format that the text-based configuration file and wpa_config_set() are using for the value. The caller is responsible for freeing the returned value.

Definition at line 1557 of file config.c.

#### 6.15.3.10 const struct wpa_config_blob* wpa_config_get_blob (struct wpa_config * *config*, const char * *name*)

Get a named configuration blob.

**Parameters:**
***config*** Configuration data from wpa_config_read()

***name*** Name of the blob

**Returns:**
Pointer to blob data or NULL if not found

Definition at line 1648 of file config.c.

#### 6.15.3.11 struct wpa_ssid* wpa_config_get_network (struct wpa_config * *config*, int *id*)

Get configured network based on id.

**Parameters:**
***config*** Configuration data from wpa_config_read()

***id*** Unique network id to search for

**Returns:**
Network configuration or NULL if not found

Definition at line 1392 of file config.c.

#### 6.15.3.12 char* wpa_config_get_no_key (struct wpa_ssid * *ssid*, const char * *var*)

Get a variable in network configuration (no keys).

**Parameters:**
***ssid*** Pointer to network configuration data

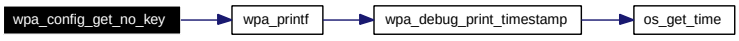
***var*** Variable name, e.g., "ssid"

**Returns:**
Value of the variable or NULL on failure

This function can be used to get network configuration variable like wpa_config_get(). The only difference is that this functions does not expose key/password material from the configuration. In case a key/password field is requested, the returned value is an empty string or NULL if the variable is not set or "*" if the variable is set (regardless of its value). The returned value is a copy of the configuration variable in text format, i.e,. the same format that the text-based configuration file and wpa_config_set() are using for the value. The caller is responsible for freeing the returned value.

Definition at line 1591 of file config.c.

Here is the call graph for this function:




#### 6.15.3.13 int wpa_config_remove_blob (struct wpa_config * *config*, const char * *name*)

Remove a named configuration blob.

**Parameters:**

***config*** Configuration data from wpa_config_read()

***name*** Name of the blob to remove

**Returns:**

0 if blob was removed or -1 if blob was not found

Definition at line 1702 of file config.c.

Here is the call graph for this function:


#### 6.15.3.14 int wpa_config_remove_network (struct wpa_config * *config*, int *id*)

Remove a configured network based on id.

**Parameters:**

***config*** Configuration data from wpa_config_read()

***id*** Unique network id to search for

**Returns:**

0 on success, or -1 if the network was not found

Definition at line 1450 of file config.c.

Here is the call graph for this function:


#### 6.15.3.15 int wpa_config_set (struct wpa_ssid * *ssid*, const char * *var*, const char * *value*, int *line*)

Set a variable in network configuration.

**Parameters:**

***ssid*** Pointer to network configuration data

***var*** Variable name, e.g., "ssid"

***value*** Variable value

***line*** Line number in configuration file or 0 if not used

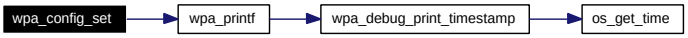
**Returns:**

0 on success, -1 on failure

This function can be used to set network configuration variables based on both the configuration file and management interface input. The value parameter must be in the same format as the text-based configuration file is using. For example, strings are using double quotation marks.

Definition at line 1509 of file config.c.

Here is the call graph for this function:




#### 6.15.3.16 void wpa_config_set_blob (struct wpa_config * *config*, struct wpa_config_blob * *blob*)

Set or add a named configuration blob.

**Parameters:**

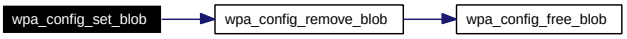
***config*** Configuration data from wpa_config_read()

***blob*** New value for the blob

Adds a new configuration blob or replaces the current value of an existing blob.

Definition at line 1671 of file config.c.

Here is the call graph for this function:




#### 6.15.3.17 void wpa_config_set_network_defaults (struct wpa_ssid * *ssid*)

Set network default values.

**Parameters:**

***ssid*** Pointer to network configuration data

Definition at line 1481 of file config.c.

#### 6.15.3.18 void wpa_config_update_psk (struct wpa_ssid * *ssid*)

Update WPA PSK based on passphrase and SSID.

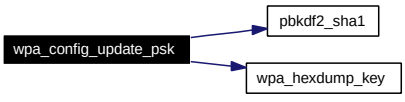
**Parameters:**

***ssid*** Pointer to network configuration data

This function must be called to update WPA PSK when either SSID or the passphrase has changed for the network configuration.

Definition at line 1630 of file config.c.

Here is the call graph for this function:

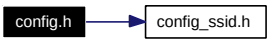


## 6.16 config.h File Reference

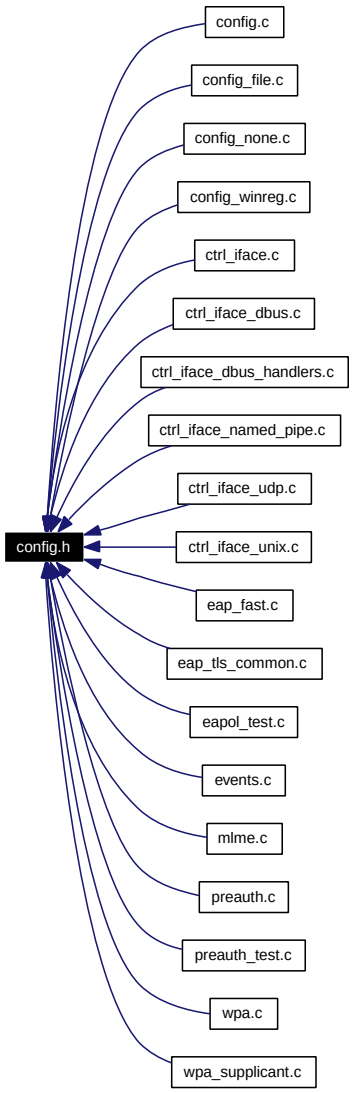
WPA Supplicant / Configuration file structures.

`#include "config_ssid.h"`

Include dependency graph for config.h:




This graph shows which files directly or indirectly include this file:




### Defines

- #define **DEFAULT_EAPOL_VERSION** 1

- #define **DEFAULT_AP_SCAN** 1
- #define **DEFAULT_FAST_REAUTH** 1

## Functions

- void wpa_config_free (struct wpa_config *ssid)

  *Free configuration data.*

- void wpa_config_free_ssid (struct wpa_ssid *ssid)

  *Free network/ssid configuration data.*

- wpa_ssid * wpa_config_get_network (struct wpa_config *config, int id)

  *Get configured network based on id.*

- wpa_ssid * wpa_config_add_network (struct wpa_config *config)

  *Add a new network with empty configuration.*

- int wpa_config_remove_network (struct wpa_config *config, int id)

  *Remove a configured network based on id.*

- void wpa_config_set_network_defaults (struct wpa_ssid *ssid)

  *Set network default values.*

- int wpa_config_set (struct wpa_ssid *ssid, const char *var, const char *value, int line)

  *Set a variable in network configuration.*

- char * wpa_config_get (struct wpa_ssid *ssid, const char *var)

  *Get a variable in network configuration.*

- char * wpa_config_get_no_key (struct wpa_ssid *ssid, const char *var)

  *Get a variable in network configuration (no keys).*

- void wpa_config_update_psk (struct wpa_ssid *ssid)

  *Update WPA PSK based on passphrase and SSID.*

- int wpa_config_add_prio_network (struct wpa_config *config, struct wpa_ssid *ssid)

  *Add a network to priority lists.*

- const struct wpa_config_blob * wpa_config_get_blob (struct wpa_config *config, const char *name)

  *Get a named configuration blob.*

- void wpa_config_set_blob (struct wpa_config *config, struct wpa_config_blob *blob)

  *Set or add a named configuration blob.*

- void wpa_config_free_blob (struct wpa_config_blob *blob)

  *Free blob data.*

- int wpa_config_remove_blob (struct wpa_config *config, const char *name)

*Remove a named configuration blob.*

- wpa_config * wpa_config_alloc_empty (const char *ctrl_interface, const char *driver_param)

  *Allocate an empty configuration.*

- void wpa_config_debug_dump_networks (struct wpa_config *config)

  *Debug dump of configured networks.*

- wpa_config * wpa_config_read (const char *name)

  *Read and parse configuration database.*

- int wpa_config_write (const char *name, struct wpa_config *config)

  *Write or update configuration data.*

## 6.16.1 Detailed Description

WPA Supplicant / Configuration file structures.

**Copyright**
Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file config.h.

## 6.16.2 Function Documentation

### 6.16.2.1 struct wpa_ssid* wpa_config_add_network (struct wpa_config * *config*)

Add a new network with empty configuration.

**Parameters:**
***config*** Configuration data from wpa_config_read()

**Returns:**
The new network configuration or NULL if operation failed

Definition at line 1413 of file config.c.

Here is the call graph for this function:

#### 6.16.2.2 int wpa_config_add_prio_network (struct wpa_config * *config*, struct wpa_ssid * *ssid*)

Add a network to priority lists.

**Parameters:**

***config*** Configuration data from wpa_config_read()

***ssid*** Pointer to the network configuration to be added to the list

**Returns:**

0 on success, -1 on failure

This function is used to add a network block to the priority list of networks. This must be called for each network when reading in the full configuration. In addition, this can be used indirectly when updating priorities by calling wpa_config_update_prio_list().

Definition at line 1194 of file config.c.

#### 6.16.2.3 struct wpa_config* wpa_config_alloc_empty (const char * *ctrl_interface*, const char * *driver_param*)

Allocate an empty configuration.

**Parameters:**

***ctrl_interface*** Control interface parameters, e.g., path to UNIX domain socket

***driver_param*** Driver parameters

**Returns:**

Pointer to allocated configuration data or NULL on failure

Definition at line 1731 of file config.c.

Here is the call graph for this function:


#### 6.16.2.4 void wpa_config_debug_dump_networks (struct wpa_config * *config*)

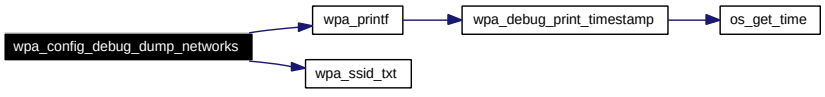
Debug dump of configured networks.

**Parameters:**

***config*** Configuration data from wpa_config_read()

Definition at line 1758 of file config.c.

Here is the call graph for this function:



#### 6.16.2.5 void wpa_config_free (struct wpa_config * *config*)

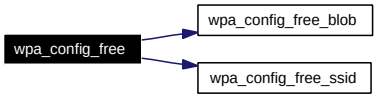
Free configuration data.

**Parameters:**
***config*** Configuration data from wpa_config_read()

This function frees all resources allocated for the configuration data by wpa_config_read().

Definition at line 1326 of file config.c.

Here is the call graph for this function:




#### 6.16.2.6 void wpa_config_free_blob (struct wpa_config_blob * *blob*)

Free blob data.

**Parameters:**
***blob*** Pointer to blob to be freed

Definition at line 1685 of file config.c.

#### 6.16.2.7 void wpa_config_free_ssid (struct wpa_ssid * *ssid*)

Free network/ssid configuration data.

**Parameters:**
***ssid*** Configuration data for the network

This function frees all resources allocated for the network configuration data.

Definition at line 1275 of file config.c.

#### 6.16.2.8 char* wpa_config_get (struct wpa_ssid * *ssid*, const char * *var*)

Get a variable in network configuration.

**Parameters:**
***ssid*** Pointer to network configuration data
***var*** Variable name, e.g., "ssid"

**Returns:**
Value of the variable or NULL on failure

This function can be used to get network configuration variables. The returned value is a copy of the configuration variable in text format, i.e,. the same format that the text-based configuration file and wpa_config_set() are using for the value. The caller is responsible for freeing the returned value.

Definition at line 1557 of file config.c.

#### 6.16.2.9 const struct wpa_config_blob* wpa_config_get_blob (struct wpa_config * *config*, const char * *name*)

Get a named configuration blob.

**Parameters:**

***config*** Configuration data from wpa_config_read()

***name*** Name of the blob

**Returns:**

Pointer to blob data or NULL if not found

Definition at line 1648 of file config.c.

#### 6.16.2.10 struct wpa_ssid* wpa_config_get_network (struct wpa_config * *config*, int *id*)

Get configured network based on id.

**Parameters:**

***config*** Configuration data from wpa_config_read()

***id*** Unique network id to search for

**Returns:**

Network configuration or NULL if not found

Definition at line 1392 of file config.c.

#### 6.16.2.11 char* wpa_config_get_no_key (struct wpa_ssid * *ssid*, const char * *var*)

Get a variable in network configuration (no keys).

**Parameters:**

***ssid*** Pointer to network configuration data

***var*** Variable name, e.g., "ssid"

**Returns:**

Value of the variable or NULL on failure

This function can be used to get network configuration variable like wpa_config_get(). The only difference is that this functions does not expose key/password material from the configuration. In case a key/password field is requested, the returned value is an empty string or NULL if the variable is not set or "*" if the variable is set (regardless of its value). The returned value is a copy of the configuration variable in text format, i.e,. the same format that the text-based configuration file and wpa_config_set() are using for the value. The caller is responsible for freeing the returned value.

Definition at line 1591 of file config.c.

Here is the call graph for this function:

#### 6.16.2.12 struct wpa_config* wpa_config_read (const char * *name*)

Read and parse configuration database.

**Parameters:**
***name*** Name of the configuration (e.g., path and file name for the configuration file)

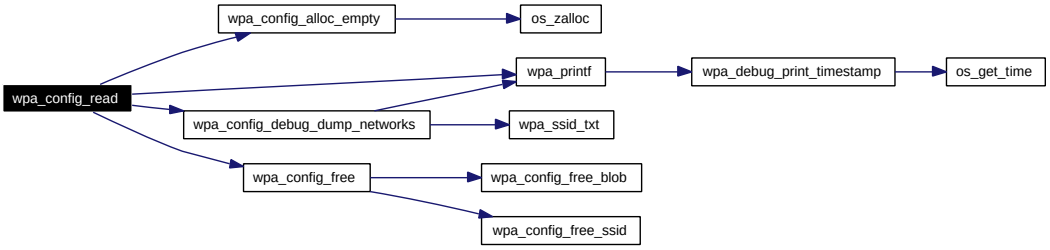
**Returns:**
Pointer to allocated configuration data or NULL on failure

This function reads configuration data, parses its contents, and allocates data structures needed for storing configuration information. The allocated data can be freed with wpa_config_free().

Each configuration backend needs to implement this function.

Definition at line 245 of file config_file.c.

Here is the call graph for this function:




#### 6.16.2.13 int wpa_config_remove_blob (struct wpa_config * *config*, const char * *name*)

Remove a named configuration blob.

**Parameters:**
***config*** Configuration data from wpa_config_read()
***name*** Name of the blob to remove

**Returns:**
0 if blob was removed or -1 if blob was not found

Definition at line 1702 of file config.c.

Here is the call graph for this function:


#### 6.16.2.14 int wpa_config_remove_network (struct wpa_config * *config*, int *id*)

Remove a configured network based on id.

**Parameters:**

***config*** Configuration data from wpa_config_read()

***id*** Unique network id to search for

**Returns:**

0 on success, or -1 if the network was not found

Definition at line 1450 of file config.c.

Here is the call graph for this function:


#### 6.16.2.15 int wpa_config_set (struct wpa_ssid * *ssid*, const char * *var*, const char * *value*, int *line*)

Set a variable in network configuration.

**Parameters:**

***ssid*** Pointer to network configuration data

***var*** Variable name, e.g., "ssid"

***value*** Variable value

***line*** Line number in configuration file or 0 if not used

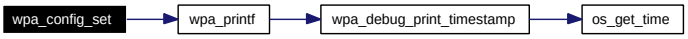
**Returns:**

0 on success, -1 on failure

This function can be used to set network configuration variables based on both the configuration file and management interface input. The value parameter must be in the same format as the text-based configuration file is using. For example, strings are using double quotation marks.

Definition at line 1509 of file config.c.

Here is the call graph for this function:




#### 6.16.2.16 void wpa_config_set_blob (struct wpa_config * *config*, struct wpa_config_blob * *blob*)

Set or add a named configuration blob.

**Parameters:**

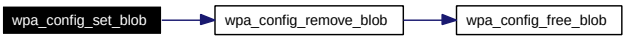
***config*** Configuration data from wpa_config_read()

***blob*** New value for the blob

Adds a new configuration blob or replaces the current value of an existing blob.

Definition at line 1671 of file config.c.

Here is the call graph for this function:




#### 6.16.2.17 void wpa_config_set_network_defaults (struct wpa_ssid * *ssid*)

Set network default values.

**Parameters:**
***ssid*** Pointer to network configuration data

Definition at line 1481 of file config.c.

#### 6.16.2.18 void wpa_config_update_psk (struct wpa_ssid * *ssid*)

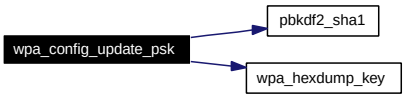
Update WPA PSK based on passphrase and SSID.

**Parameters:**
***ssid*** Pointer to network configuration data

This function must be called to update WPA PSK when either SSID or the passphrase has changed for the network configuration.

Definition at line 1630 of file config.c.

Here is the call graph for this function:




#### 6.16.2.19 int wpa_config_write (const char * *name*, struct wpa_config * *config*)

Write or update configuration data.

**Parameters:**
***name*** Name of the configuration (e.g., path and file name for the configuration file)

***config*** Configuration data from wpa_config_read()

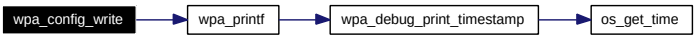
**Returns:**
0 on success, -1 on failure

This function write all configuration data into an external database (e.g., a text file) in a format that can be read with wpa_config_read(). This can be used to allow wpa_supplicant to update its configuration, e.g., when a new network is added or a password is changed.

Each configuration backend needs to implement this function.

Definition at line 692 of file config_file.c.

Here is the call graph for this function:

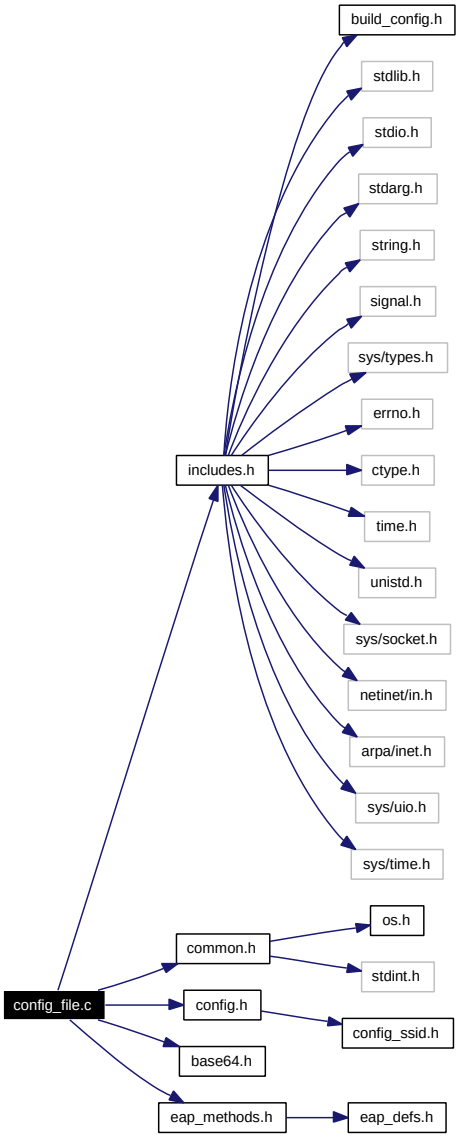


## 6.17 config_file.c File Reference

WPA Supplicant / Configuration backend: text file.

```
#include "includes.h"
#include "common.h"
#include "config.h"
#include "base64.h"
#include "eap_methods.h"
```

Include dependency graph for config_file.c:



## Defines

- #define **STR**(t) write_str(f, #t, ssid)
- #define **INT**(t) write_int(f, #t, ssid → t, 0)
- #define **INT_DEF**(t, def) write_int(f, #t, ssid → t, def)

## Functions

- wpa_config * wpa_config_read (const char *name)
  *Read and parse configuration database.*
- int wpa_config_write (const char *name, struct wpa_config *config)
  *Write or update configuration data.*

### 6.17.1 Detailed Description

WPA Supplicant / Configuration backend: text file.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements a configuration backend for text files. All the configuration information is stored in a text file that uses a format described in the sample configuration file, wpa_supplicant.conf.

Definition in file config_file.c.

### 6.17.2 Function Documentation

#### 6.17.2.1 struct wpa_config* wpa_config_read (const char * *name*)

Read and parse configuration database.

**Parameters:**
***name*** Name of the configuration (e.g., path and file name for the configuration file)

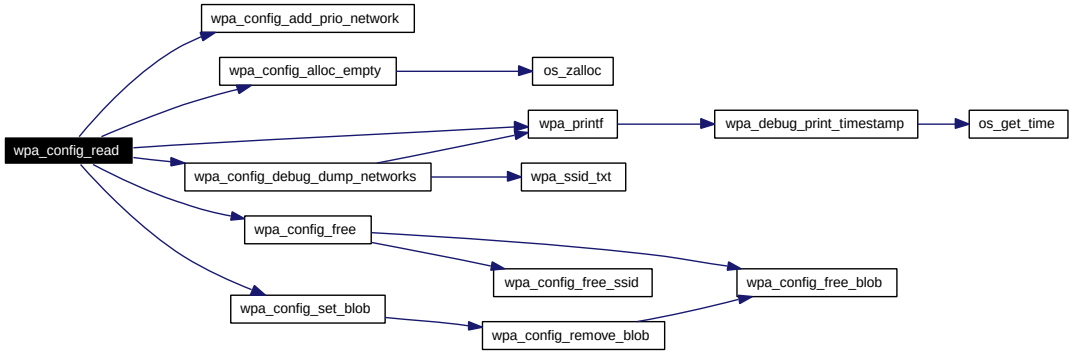
**Returns:**
Pointer to allocated configuration data or NULL on failure

This function reads configuration data, parses its contents, and allocates data structures needed for storing configuration information. The allocated data can be freed with wpa_config_free().

Each configuration backend needs to implement this function.

Definition at line 245 of file config_file.c.

Here is the call graph for this function:




#### 6.17.2.2 int wpa_config_write (const char * *name*, struct wpa_config * *config*)

Write or update configuration data.

**Parameters:**

***name*** Name of the configuration (e.g., path and file name for the configuration file)

***config*** Configuration data from wpa_config_read()

**Returns:**

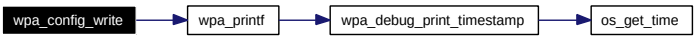
0 on success, -1 on failure

This function write all configuration data into an external database (e.g., a text file) in a format that can be read with wpa_config_read(). This can be used to allow wpa_supplicant to update its configuration, e.g., when a new network is added or a password is changed.

Each configuration backend needs to implement this function.

Definition at line 692 of file config_file.c.

Here is the call graph for this function:

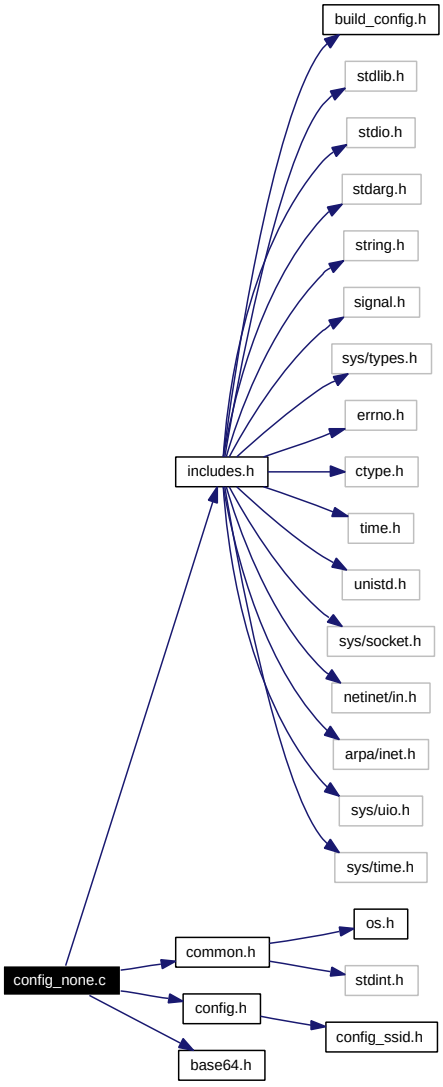


## 6.18 config_none.c File Reference

WPA Supplicant / Configuration backend: empty starting point.

```
#include "includes.h"
#include "common.h"
#include "config.h"
#include "base64.h"
```

Include dependency graph for config_none.c:




## Functions

- wpa_config * wpa_config_read (const char *name)

  *Read and parse configuration database.*

- int wpa_config_write (const char *name, struct wpa_config *config)

  *Write or update configuration data.*

## 6.18.1 Detailed Description

WPA Supplicant / Configuration backend: empty starting point.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements dummy example of a configuration backend. None of the functions are actually implemented so this can be used as a simple compilation test or a starting point for a new configuration backend.

Definition in file config_none.c.

## 6.18.2 Function Documentation

### 6.18.2.1 struct wpa_config* wpa_config_read (const char * *name*)

Read and parse configuration database.

**Parameters:**

***name*** Name of the configuration (e.g., path and file name for the configuration file)

**Returns:**

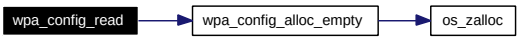
Pointer to allocated configuration data or NULL on failure

This function reads configuration data, parses its contents, and allocates data structures needed for storing configuration information. The allocated data can be freed with wpa_config_free().

Each configuration backend needs to implement this function.

Definition at line 27 of file config_none.c.

Here is the call graph for this function:




### 6.18.2.2 int wpa_config_write (const char * *name*, struct wpa_config * *config*)

Write or update configuration data.

**Parameters:**

***name*** Name of the configuration (e.g., path and file name for the configuration file)

***config*** Configuration data from wpa_config_read()

**Returns:**

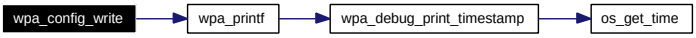
0 on success, -1 on failure

This function write all configuration data into an external database (e.g., a text file) in a format that can be read with wpa_config_read(). This can be used to allow wpa_supplicant to update its configuration, e.g., when a new network is added or a password is changed.

Each configuration backend needs to implement this function.

Definition at line 39 of file config_none.c.

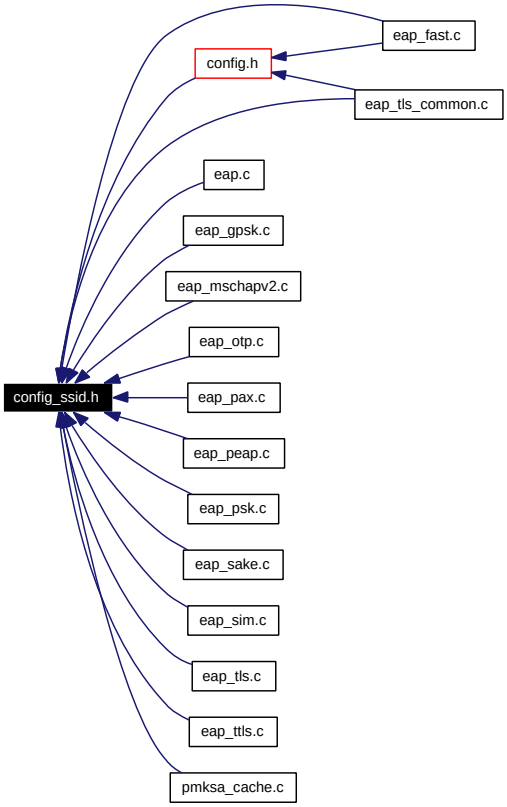
Here is the call graph for this function:



## 6.19 config_ssid.h File Reference

WPA Supplicant / Network configuration structures.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **BIT**(n) (1 << (n))
- #define **WPA_CIPHER_NONE** BIT(0)
- #define **WPA_CIPHER_WEP40** BIT(1)
- #define **WPA_CIPHER_WEP104** BIT(2)
- #define **WPA_CIPHER_TKIP** BIT(3)
- #define **WPA_CIPHER_CCMP** BIT(4)
- #define **WPA_KEY_MGMT_IEEE8021X** BIT(0)
- #define **WPA_KEY_MGMT_PSK** BIT(1)
- #define **WPA_KEY_MGMT_NONE** BIT(2)
- #define **WPA_KEY_MGMT_IEEE8021X_NO_WPA** BIT(3)
- #define **WPA_KEY_MGMT_WPA_NONE** BIT(4)
- #define **WPA_PROTO_WPA** BIT(0)
- #define **WPA_PROTO_RSN** BIT(1)
- #define **WPA_AUTH_ALG_OPEN** BIT(0)
- #define **WPA_AUTH_ALG_SHARED** BIT(1)
- #define **WPA_AUTH_ALG_LEAP** BIT(2)

- #define **MAX_SSID_LEN** 32
- #define **PMK_LEN** 32
- #define **EAP_PSK_LEN_MIN** 16
- #define **EAP_PSK_LEN_MAX** 32
- #define **DEFAULT_EAP_WORKAROUND** ((unsigned int) -1)
- #define **DEFAULT_EAPOL_FLAGS**
- #define **DEFAULT_PROTO** (WPA_PROTO_WPA | WPA_PROTO_RSN)
- #define **DEFAULT_KEY_MGMT** (WPA_KEY_MGMT_PSK | WPA_KEY_MGMT_IEEE8021X)
- #define **DEFAULT_PAIRWISE** (WPA_CIPHER_CCMP | WPA_CIPHER_TKIP)
- #define **DEFAULT_GROUP**
- #define **DEFAULT_FRAGMENT_SIZE** 1398
- #define **EAPOL_FLAG_REQUIRE_KEY_UNICAST** BIT(0)
- #define **EAPOL_FLAG_REQUIRE_KEY_BROADCAST** BIT(1)
- #define **NUM_WEP_KEYS** 4
- #define **MAX_WEP_KEY_LEN** 16

## Functions

- int wpa_config_allowed_eap_method (struct wpa_ssid *ssid, int vendor, u32 method)

  *Check whether EAP method is allowed.*

### 6.19.1 Detailed Description

WPA Supplicant / Network configuration structures.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file config_ssid.h.

### 6.19.2 Define Documentation

#### 6.19.2.1 #define DEFAULT_EAPOL_FLAGS

**Value:**

```
(EAPOL_FLAG_REQUIRE_KEY_UNICAST | \
                            EAPOL_FLAG_REQUIRE_KEY_BROADCAST)
```

Definition at line 52 of file config_ssid.h.

#### 6.19.2.2 #define DEFAULT_GROUP

**Value:**

```
(WPA_CIPHER_CCMP | WPA_CIPHER_TKIP | \
                      WPA_CIPHER_WEP104 | WPA_CIPHER_WEP40)
```

Definition at line 57 of file config_ssid.h.

### 6.19.3 Function Documentation

#### 6.19.3.1 int wpa_config_allowed_eap_method (struct wpa_ssid * *ssid*, int *vendor*, u32 *method*)

Check whether EAP method is allowed.

**Parameters:**

***ssid*** Pointer to configuration data

***vendor*** Vendor-Id for expanded types or 0 = IETF for legacy types

***method*** EAP type

**Returns:**

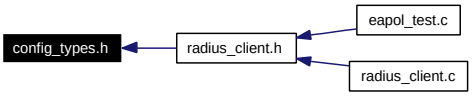
1 = allowed EAP method, 0 = not allowed

Definition at line 1365 of file config.c.

## 6.20 config_types.h File Reference

hostapd / Shared configuration file defines

This graph shows which files directly or indirectly include this file:




### 6.20.1 Detailed Description

hostapd / Shared configuration file defines

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

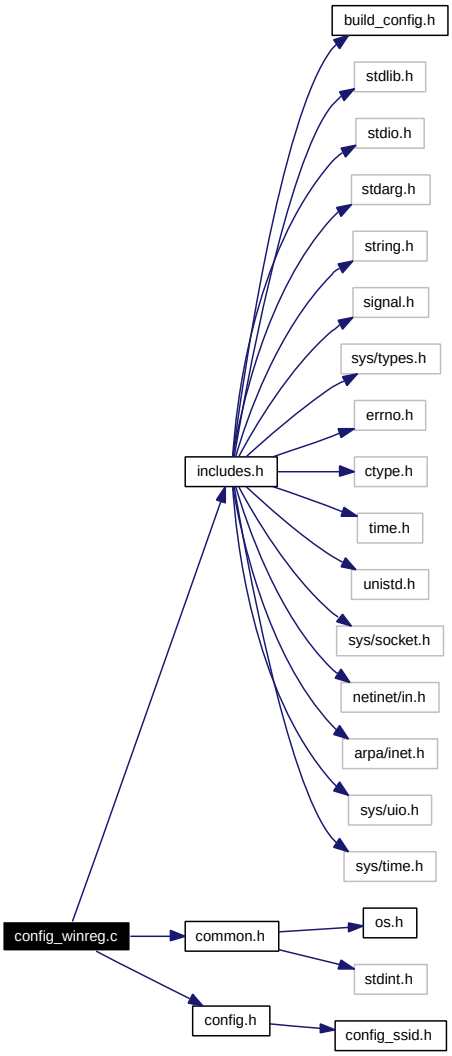
Definition in file config_types.h.

## 6.21 config_winreg.c File Reference

WPA Supplicant / Configuration backend: Windows registry.

```
#include "includes.h"
#include "common.h"
#include "config.h"
```

Include dependency graph for config_winreg.c:




### Defines

- #define **WPA_KEY_ROOT** HKEY_LOCAL_MACHINE
- #define **WPA_KEY_PREFIX** TEXT("SOFTWARE\\wpa_supplicant")
- #define **TSTR** "%s"
- #define **TNAMELEN** 255
- #define **STR**(t) write_str(netw, #t, ssid)

- #define **INT**(t) write_int(netw, #t, ssid → t, 0)
- #define **INT_DEF**(t, def) write_int(netw, #t, ssid → t, def)

## Functions

- wpa_config * wpa_config_read (const char *name)

  *Read and parse configuration database.*

- int wpa_config_write (const char *name, struct wpa_config *config)

  *Write or update configuration data.*

### 6.21.1 Detailed Description

WPA Supplicant / Configuration backend: Windows registry.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements a configuration backend for Windows registry.. All the configuration information is stored in the registry and the format for network configuration fields is same as described in the sample configuration file, wpa_supplicant.conf.

Configuration data is in HKEY_LOCAL_MACHINE key. Each configuration profile has its own key under this. In terms of text files, each profile would map to a separate text file with possibly multiple networks. Under each profile, there is a networks key that lists all networks as a subkey. Each network has set of values in the same way as network block in the configuration file. In addition, blobs subkey has possible blobs as values.

HKEY_LOCAL_MACHINE ssid="example" key_mgmt=WPA-PSK

Definition in file config_winreg.c.

### 6.21.2 Function Documentation

#### 6.21.2.1 struct wpa_config* wpa_config_read (const char * *name*)

Read and parse configuration database.

**Parameters:**
***name*** Name of the configuration (e.g., path and file name for the configuration file)

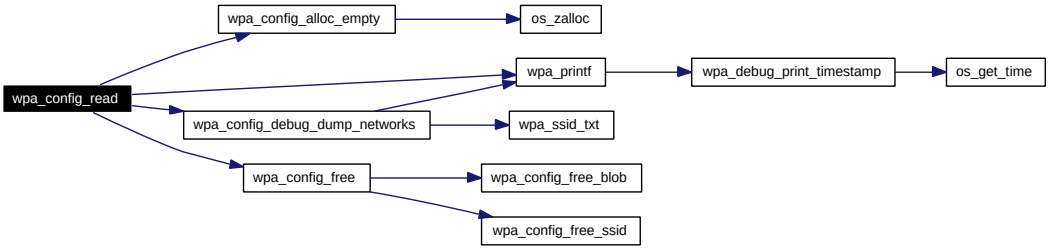
**Returns:**
Pointer to allocated configuration data or NULL on failure

This function reads configuration data, parses its contents, and allocates data structures needed for storing configuration information. The allocated data can be freed with wpa_config_free().

Each configuration backend needs to implement this function.

Definition at line 360 of file config_winreg.c.

Here is the call graph for this function:




### 6.21.2.2 int wpa_config_write (const char * *name*, struct wpa_config * *config*)

Write or update configuration data.

**Parameters:**

***name*** Name of the configuration (e.g., path and file name for the configuration file)

***config*** Configuration data from wpa_config_read()

**Returns:**

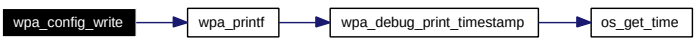
0 on success, -1 on failure

This function write all configuration data into an external database (e.g., a text file) in a format that can be read with wpa_config_read(). This can be used to allow wpa_supplicant to update its configuration, e.g., when a new network is added or a password is changed.

Each configuration backend needs to implement this function.

Definition at line 834 of file config_winreg.c.

Here is the call graph for this function:

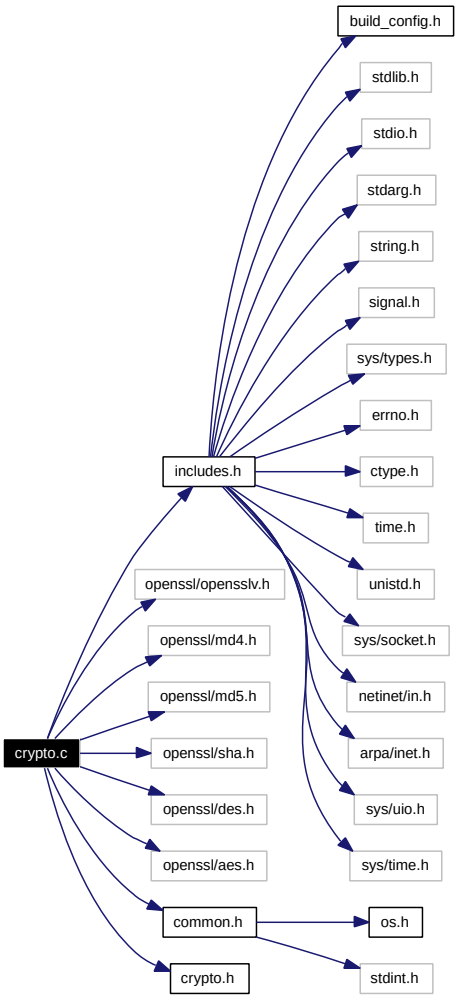


## 6.22 crypto.c File Reference

WPA Supplicant / wrapper functions for libcrypto.

```
#include "includes.h"
#include <openssl/opensslv.h>
#include <openssl/md4.h>
#include <openssl/md5.h>
#include <openssl/sha.h>
#include <openssl/des.h>
#include <openssl/aes.h>
#include "common.h"
#include "crypto.h"
```

Include dependency graph for crypto.c:



## Defines

- #define **DES_key_schedule** des_key_schedule
- #define **DES_cblock** des_cblock
- #define **DES_set_key**(key, schedule) des_set_key((key), *(schedule))
- #define **DES_ecb_encrypt**(input, output, ks, enc) des_ecb_encrypt((input), (output), *(ks), (enc))

## Functions

- void md4_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *MD4 hash for data vector.*

- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)

  *Encrypt one block with DES.*

### 6.22.1 Detailed Description

WPA Supplicant / wrapper functions for libcrypto.

**Copyright**

Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file crypto.c.

### 6.22.2 Function Documentation

#### 6.22.2.1 void des_encrypt (const u8 * *clear*, const u8 * *key*, u8 * *cypher*)

Encrypt one block with DES.

**Parameters:**

***clear*** 8 octets (in)

***key*** 7 octets (in) (no parity bits included)

***cypher*** 8 octets (out)

Definition at line 48 of file crypto.c.

#### 6.22.2.2 void md4_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

MD4 hash for data vector.

**Parameters:**

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

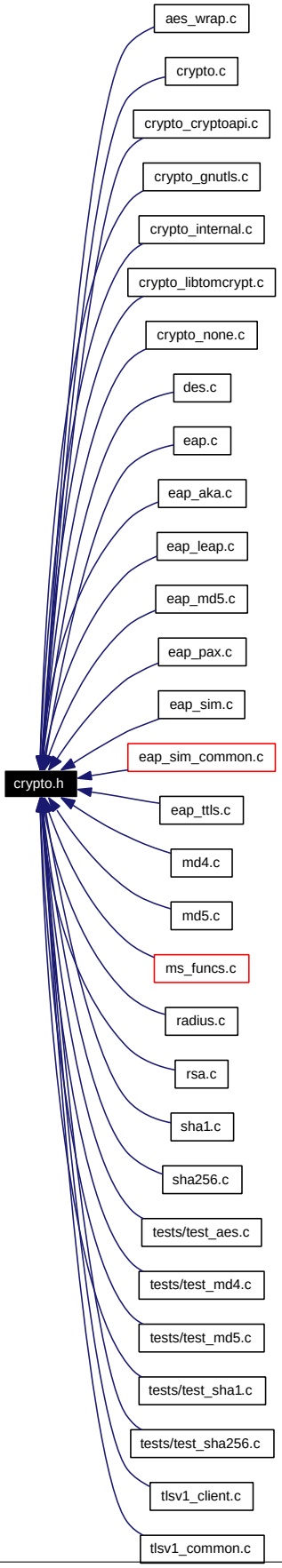
***mac*** Buffer for the hash

Definition at line 36 of file crypto.c.

## 6.23 crypto.h File Reference

WPA Supplicant / wrapper functions for crypto libraries.

This graph shows which files directly or indirectly include this file:


aes_wrap.c
crypto.c
crypto_cryptoapi.c
crypto_gnutls.c
crypto_internal.c
crypto_libtomcrypt.c
crypto_none.c
des.c
eap.c
eap_aka.c
eap_leap.c
eap_md5.c
eap_pax.c
eap_sim.c
eap_sim_common.c
crypto.h
eap_ttls.c
md4.c
md5.c
ms_funcs.c
radius.c
rsa.c
sha1.c
sha256.c
tests/test_aes.c
tests/test_md4.c
tests/test_md5.c
tests/test_sha1.c
tests/test_sha256.c
tlsv1_client.c
tlsv1_common.c

## Enumerations

- enum **crypto_hash_alg** { **CRYPTO_HASH_ALG_MD5**, **CRYPTO_HASH_ALG_SHA1**, **CRYPTO_HASH_ALG_HMAC_MD5**, **CRYPTO_HASH_ALG_HMAC_SHA1** }
- enum **crypto_cipher_alg** {

  **CRYPTO_CIPHER_NULL** = 0, **CRYPTO_CIPHER_ALG_AES**, **CRYPTO_CIPHER_ALG_-3DES**, **CRYPTO_CIPHER_ALG_DES**,

  **CRYPTO_CIPHER_ALG_RC2**, **CRYPTO_CIPHER_ALG_RC4** }

## Functions

- void md4_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *MD4 hash for data vector.*

- void md5_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *MD5 hash for data vector.*

- void sha1_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *SHA-1 hash for data vector.*

- int fips186_2_prf (const u8 *seed, size_t seed_len, u8 *x, size_t xlen)

  *NIST FIPS Publication 186-2 change notice 1 PRF.*

- void sha256_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *SHA256 hash for data vector.*

- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)

  *Encrypt one block with DES.*

- void * aes_encrypt_init (const u8 *key, size_t len)

  *Initialize AES for encryption.*

- void aes_encrypt (void *ctx, const u8 *plain, u8 *crypt)

  *Encrypt one AES block.*

- void aes_encrypt_deinit (void *ctx)

  *Deinitialize AES encryption.*

- void * aes_decrypt_init (const u8 *key, size_t len)

  *Initialize AES for decryption.*

- void aes_decrypt (void *ctx, const u8 *crypt, u8 *plain)

  *Decrypt one AES block.*

- void aes_decrypt_deinit (void *ctx)

  *Deinitialize AES decryption.*

- crypto_hash * crypto_hash_init (enum crypto_hash_alg alg, const u8 *key, size_t key_len)

  *Initialize hash/HMAC function.*

- void crypto_hash_update (struct crypto_hash *ctx, const u8 *data, size_t len)

  *Add data to hash calculation.*

- int crypto_hash_finish (struct crypto_hash *ctx, u8 *hash, size_t *len)

  *Complete hash calculation.*

- crypto_cipher * crypto_cipher_init (enum crypto_cipher_alg alg, const u8 *iv, const u8 *key, size_t key_len)

  *Initialize block/stream cipher function.*

- int crypto_cipher_encrypt (struct crypto_cipher *ctx, const u8 *plain, u8 *crypt, size_t len)

  *Cipher encrypt.*

- int crypto_cipher_decrypt (struct crypto_cipher *ctx, const u8 *crypt, u8 *plain, size_t len)

  *Cipher decrypt.*

- void crypto_cipher_deinit (struct crypto_cipher *ctx)

  *Free cipher context.*

- crypto_public_key * crypto_public_key_import (const u8 *key, size_t len)

  *Import an RSA public key.*

- crypto_private_key * crypto_private_key_import (const u8 *key, size_t len)

  *Import an RSA private key.*

- crypto_public_key * crypto_public_key_from_cert (const u8 *buf, size_t len)

  *Import an RSA public key from a certificate.*

- int crypto_public_key_encrypt_pkcs1_v15 (struct crypto_public_key *key, const u8 *in, size_t inlen, u8 *out, size_t *outlen)

  *Public key encryption (PKCS #1 v1.5).*

- int crypto_private_key_sign_pkcs1 (struct crypto_private_key *key, const u8 *in, size_t inlen, u8 *out, size_t *outlen)

  *Sign with private key (PKCS #1).*

- void crypto_public_key_free (struct crypto_public_key *key)

  *Free public key.*

- void crypto_private_key_free (struct crypto_private_key *key)

  *Free private key.*

- int crypto_public_key_decrypt_pkcs1 (struct crypto_public_key *key, const u8 *crypt, size_t crypt_-len, u8 *plain, size_t *plain_len)

  *Decrypt PKCS #1 signature.*

- int crypto_global_init (void)

  *Initialize crypto wrapper.*

- void crypto_global_deinit (void)
  *Deinitialize crypto wrapper.*
- int crypto_mod_exp (const u8 *base, size_t base_len, const u8 *power, size_t power_len, const u8 *modulus, size_t modulus_len, u8 *result, size_t *result_len)
  *Modular exponentiation of large integers.*

## 6.23.1 Detailed Description

WPA Supplicant / wrapper functions for crypto libraries.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file defines the cryptographic functions that need to be implemented for wpa_supplicant and hostapd. When TLS is not used, internal implementation of MD5, SHA1, and AES is used and no external libraries are required. When TLS is enabled (e.g., by enabling EAP-TLS or EAP-PEAP), the crypto library used by the TLS implementation is expected to be used for non-TLS needs, too, in order to save space by not implementing these functions twice.

Wrapper code for using each crypto library is in its own file (crypto*.c) and one of these files is build and linked in to provide the functions defined here.

Definition in file crypto.h.

## 6.23.2 Function Documentation

### 6.23.2.1 void aes_decrypt (void * *ctx*, const u8 * *crypt*, u8 * *plain*)

Decrypt one AES block.

**Parameters:**
***ctx*** Context pointer from aes_encrypt_init()
***crypt*** Encrypted data (16 bytes)
***plain*** Buffer for the decrypted data (16 bytes)

Definition at line 1099 of file aes.c.

### 6.23.2.2 void aes_decrypt_deinit (void * *ctx*)

Deinitialize AES decryption.

**Parameters:**
***ctx*** Context pointer from aes_encrypt_init()

Definition at line 1105 of file aes.c.

#### 6.23.2.3 void* aes_decrypt_init (const u8 * *key*, size_t *len*)

Initialize AES for decryption.

**Parameters:**

***key*** Decryption key

***len*** Key length in bytes (usually 16, i.e., 128 bits)

**Returns:**

Pointer to context data or NULL on failure

Definition at line 1086 of file aes.c.

Here is the call graph for this function:


#### 6.23.2.4 void aes_encrypt (void * *ctx*, const u8 * *plain*, u8 * *crypt*)

Encrypt one AES block.

**Parameters:**

***ctx*** Context pointer from aes_encrypt_init()

***plain*** Plaintext data to be encrypted (16 bytes)

***crypt*** Buffer for the encrypted data (16 bytes)

Definition at line 1074 of file aes.c.

#### 6.23.2.5 void aes_encrypt_deinit (void * *ctx*)

Deinitialize AES encryption.

**Parameters:**

***ctx*** Context pointer from aes_encrypt_init()

Definition at line 1080 of file aes.c.

#### 6.23.2.6 void* aes_encrypt_init (const u8 * *key*, size_t *len*)

Initialize AES for encryption.

**Parameters:**

***key*** Encryption key

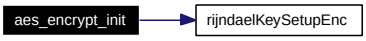
***len*** Key length in bytes (usually 16, i.e., 128 bits)

**Returns:**

Pointer to context data or NULL on failure

Definition at line 1061 of file aes.c.

Here is the call graph for this function:




#### 6.23.2.7 int crypto_cipher_decrypt (struct crypto_cipher * *ctx*, const u8 * *crypt*, u8 * *plain*, size_t *len*)

Cipher decrypt.

**Parameters:**

***ctx*** Context pointer from crypto_cipher_init()

***crypt*** Ciphertext to decrypt

***plain*** Resulting plaintext

***len*** Length of the cipher text

**Returns:**

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.8 void crypto_cipher_deinit (struct crypto_cipher * *ctx*)

Free cipher context.

**Parameters:**

***ctx*** Context pointer from crypto_cipher_init()

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.9 int crypto_cipher_encrypt (struct crypto_cipher * *ctx*, const u8 * *plain*, u8 * *crypt*, size_t *len*)

Cipher encrypt.

**Parameters:**

***ctx*** Context pointer from crypto_cipher_init()

***plain*** Plaintext to cipher

***crypt*** Resulting ciphertext

***len*** Length of the plaintext

**Returns:**

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.10 struct crypto_cipher* crypto_cipher_init (enum crypto_cipher_alg *alg*, const u8 * *iv*, const u8 * *key*, size_t *key_len*)

Initialize block/stream cipher function.

**Parameters:**

***alg*** Cipher algorithm

***iv*** Initialization vector for block ciphers or NULL for stream ciphers

***key*** Cipher key

***key_len*** Length of key in bytes

**Returns:**

Pointer to cipher context to use with other cipher functions or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.11 void crypto_global_deinit (void)

Deinitialize crypto wrapper.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.12 int crypto_global_init (void)

Initialize crypto wrapper.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.13 int crypto_hash_finish (struct crypto_hash * *ctx*, u8 * *hash*, size_t * *len*)

Complete hash calculation.

**Parameters:**

***ctx*** Context pointer from crypto_hash_init()

***hash*** Buffer for hash value or NULL if caller is just freeing the hash context

***len*** Pointer to length of the buffer or NULL if caller is just freeing the hash context; on return, this is set to the actual length of the hash value

**Returns:**

0 on success, -1 if buffer is too small (len set to needed length), or -2 on other failures (including failed crypto_hash_update() operations)

This function calculates the hash value and frees the context buffer that was used for hash calculation.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.14 struct crypto_hash* crypto_hash_init (enum crypto_hash_alg *alg*, const u8 * *key*, size_t *key_len*)

Initialize hash/HMAC function.

**Parameters:**

***alg*** Hash algorithm

***key*** Key for keyed hash (e.g., HMAC) or NULL if not needed

***key_len*** Length of the key in bytes

**Returns:**

Pointer to hash context to use with other hash functions or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.15 void crypto_hash_update (struct crypto_hash * *ctx*, const u8 * *data*, size_t *len*)

Add data to hash calculation.

**Parameters:**

***ctx*** Context pointer from crypto_hash_init()

***data*** Data buffer to add

***len*** Length of the buffer

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.16 int crypto_mod_exp (const u8 * *base*, size_t *base_len*, const u8 * *power*, size_t *power_len*, const u8 * *modulus*, size_t *modulus_len*, u8 * *result*, size_t * *result_len*)

Modular exponentiation of large integers.

**Parameters:**

***base*** Base integer (big endian byte array)

***base_len*** Length of base integer in bytes

***power*** Power integer (big endian byte array)

***power_len*** Length of power integer in bytes

***modulus*** Modulus integer (big endian byte array)

***modulus_len*** Length of modulus integer in bytes

***result*** Buffer for the result

***result_len*** Result length (max buffer size on input, real len on output)

**Returns:**

0 on success, -1 on failure

This function calculates result = base ^ power mod modulus. modules_len is used as the maximum size of modulus buffer. It is set to the used size on success.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.17 void crypto_private_key_free (struct crypto_private_key * *key*)

Free private key.

**Parameters:**
***key*** Private key from crypto_private_key_import()

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.18 struct crypto_private_key* crypto_private_key_import (const u8 * *key*, size_t *len*)

Import an RSA private key.

**Parameters:**
***key*** Key buffer (DER encoded RSA private key)

***len*** Key buffer length in bytes

**Returns:**
Pointer to the private key or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.19 int crypto_private_key_sign_pkcs1 (struct crypto_private_key * *key*, const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*)

Sign with private key (PKCS #1).

**Parameters:**
***key*** Private key from crypto_private_key_import()

***in*** Plaintext buffer

***inlen*** Length of plaintext buffer in bytes

***out*** Output buffer for encrypted (signed) data

***outlen*** Length of output buffer in bytes; set to used length on success

**Returns:**
0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.20 int crypto_public_key_decrypt_pkcs1 (struct crypto_public_key * *key*, const u8 * *crypt*, size_t *crypt_len*, u8 * *plain*, size_t * *plain_len*)

Decrypt PKCS #1 signature.

**Parameters:**
***key*** Public key

***crypt*** Encrypted signature data (using the private key)

***crypt_len*** Encrypted signature data length

***plain*** Buffer for plaintext (at least crypt_len bytes)

***plain_len*** Plaintext length (max buffer size on input, real len on output);

**Returns:**
0 on success, -1 on failure

#### 6.23.2.21 int crypto_public_key_encrypt_pkcs1_v15 (struct crypto_public_key * *key*, const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*)

Public key encryption (PKCS #1 v1.5).

**Parameters:**
***key*** Public key

***in*** Plaintext buffer

***inlen*** Length of plaintext buffer in bytes

***out*** Output buffer for encrypted data

***outlen*** Length of output buffer in bytes; set to used length on success

**Returns:**
0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.22 void crypto_public_key_free (struct crypto_public_key * *key*)

Free public key.

**Parameters:**
***key*** Public key

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.23 struct crypto_public_key* crypto_public_key_from_cert (const u8 * *buf*, size_t *len*)

Import an RSA public key from a certificate.

**Parameters:**
***buf*** DER encoded X.509 certificate

***len*** Certificate buffer length in bytes

**Returns:**
Pointer to public key or NULL on failure

This function can just return NULL if the crypto library does not support X.509 parsing. In that case, internal code will be used to parse the certificate and public key is imported using crypto_public_key_import().

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.24 struct crypto_public_key* crypto_public_key_import (const u8 * *key*, size_t *len*)

Import an RSA public key.

**Parameters:**

***key*** Key buffer (DER encoded RSA public key)

***len*** Key buffer length in bytes

**Returns:**

Pointer to the public key or NULL on failure

This function can just return NULL if the crypto library supports X.509 parsing. In that case, crypto_public_key_from_cert() is used to import the public key from a certificate.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

#### 6.23.2.25 void des_encrypt (const u8 * *clear*, const u8 * *key*, u8 * *cypher*)

Encrypt one block with DES.

**Parameters:**

***clear*** 8 octets (in)

***key*** 7 octets (in) (no parity bits included)

***cypher*** 8 octets (out)

Definition at line 48 of file crypto.c.

#### 6.23.2.26 int fips186_2_prf (const u8 * *seed*, size_t *seed_len*, u8 * *x*, size_t *xlen*)

NIST FIPS Publication 186-2 change notice 1 PRF.

**Parameters:**

***seed*** Seed/key for the PRF

***seed_len*** Seed length in bytes

***x*** Buffer for PRF output

***xlen*** Output length in bytes

**Returns:**

0 on success, -1 on failure

This function implements random number generation specified in NIST FIPS Publication 186-2 for EAP-SIM. This PRF uses a function that is similar to SHA-1, but has different message padding.

#### 6.23.2.27 void md4_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

MD4 hash for data vector.

**Parameters:**

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash

Definition at line 36 of file crypto.c.

#### 6.23.2.28 void md5_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

MD5 hash for data vector.

**Parameters:**

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash

#### 6.23.2.29 void sha1_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

SHA-1 hash for data vector.

**Parameters:**

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash

#### 6.23.2.30 void sha256_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

SHA256 hash for data vector.

**Parameters:**

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

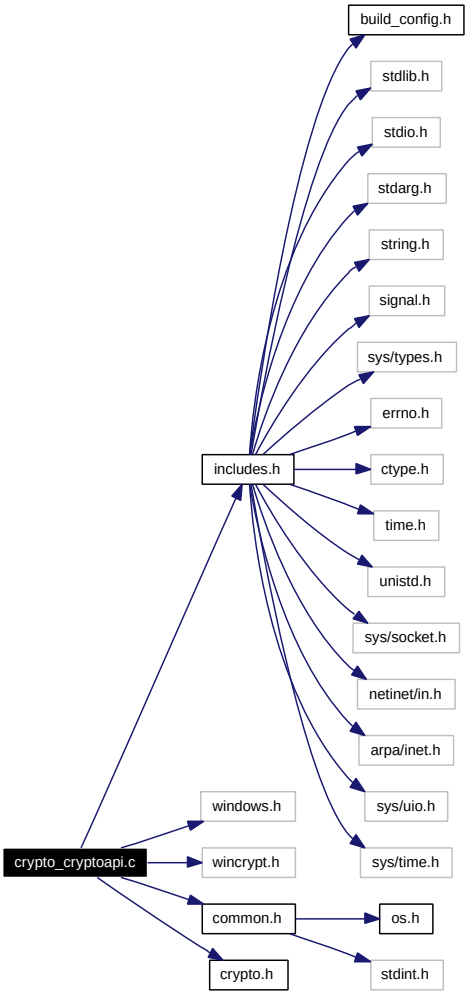
***mac*** Buffer for the hash

## 6.24 crypto_cryptoapi.c File Reference

WPA Supplicant / Crypto wrapper for Microsoft CryptoAPI.

```
#include "includes.h"
#include <windows.h>
#include <wincrypt.h>
#include "common.h"
#include "crypto.h"
```

Include dependency graph for crypto_cryptoapi.c:




### Defines

- #define **MS_ENH_RSA_AES_PROV** "Microsoft Enhanced RSA and AES Cryptographic Provider (Prototype)"
- #define **CALG_HMAC** (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_HMAC)

## Functions

- int **cryptoapi_hash_vector** (ALG_ID alg, size_t hash_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
- void md4_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *MD4 hash for data vector.*

- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)

  *Encrypt one block with DES.*

### 6.24.1 Detailed Description

WPA Supplicant / Crypto wrapper for Microsoft CryptoAPI.

**Copyright**

Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file crypto_cryptoapi.c.

### 6.24.2 Function Documentation

#### 6.24.2.1 void des_encrypt (const u8 * *clear*, const u8 * *key*, u8 * *cypher*)

Encrypt one block with DES.

**Parameters:**

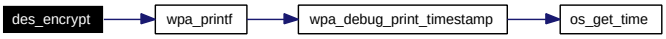
***clear*** 8 octets (in)

***key*** 7 octets (in) (no parity bits included)

***cypher*** 8 octets (out)

Definition at line 177 of file crypto_cryptoapi.c.

Here is the call graph for this function:




#### 6.24.2.2 void md4_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

MD4 hash for data vector.

**Parameters:**

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash

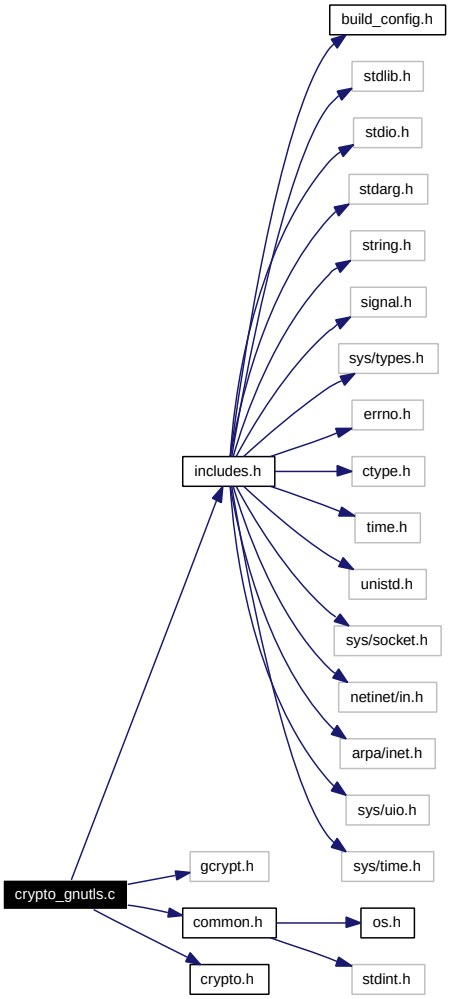
Definition at line 171 of file crypto_cryptoapi.c.

## 6.25 crypto_gnutls.c File Reference

WPA Supplicant / wrapper functions for libgcrypt.

```
#include "includes.h"
#include <gcrypt.h>
#include "common.h"
#include "crypto.h"
```

Include dependency graph for crypto_gnutls.c:




### Functions

- void md4_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  *MD4 hash for data vector.*
- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)
  *Encrypt one block with DES.*

### 6.25.1 Detailed Description

WPA Supplicant / wrapper functions for libgcrypt.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file crypto_gnutls.c.

### 6.25.2 Function Documentation

#### 6.25.2.1 void des_encrypt (const u8 * *clear*, const u8 * *key*, u8 * *cypher*)

Encrypt one block with DES.

**Parameters:**
***clear*** 8 octets (in)
***key*** 7 octets (in) (no parity bits included)
***cypher*** 8 octets (out)

Definition at line 39 of file crypto_gnutls.c.

#### 6.25.2.2 void md4_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

MD4 hash for data vector.

**Parameters:**
***num_elem*** Number of elements in the data vector
***addr*** Pointers to the data areas
***len*** Lengths of the data blocks
***mac*** Buffer for the hash

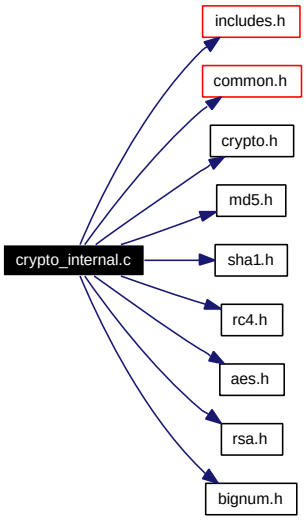
Definition at line 22 of file crypto_gnutls.c.

## 6.26 crypto_internal.c File Reference

WPA Supplicant / Crypto wrapper for internal crypto implementation.

```
#include "includes.h"
#include "common.h"
#include "crypto.h"
#include "md5.h"
#include "sha1.h"
#include "rc4.h"
#include "aes.h"
#include "rsa.h"
#include "bignum.h"
```

Include dependency graph for crypto_internal.c:




### 6.26.1 Detailed Description

WPA Supplicant / Crypto wrapper for internal crypto implementation.

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

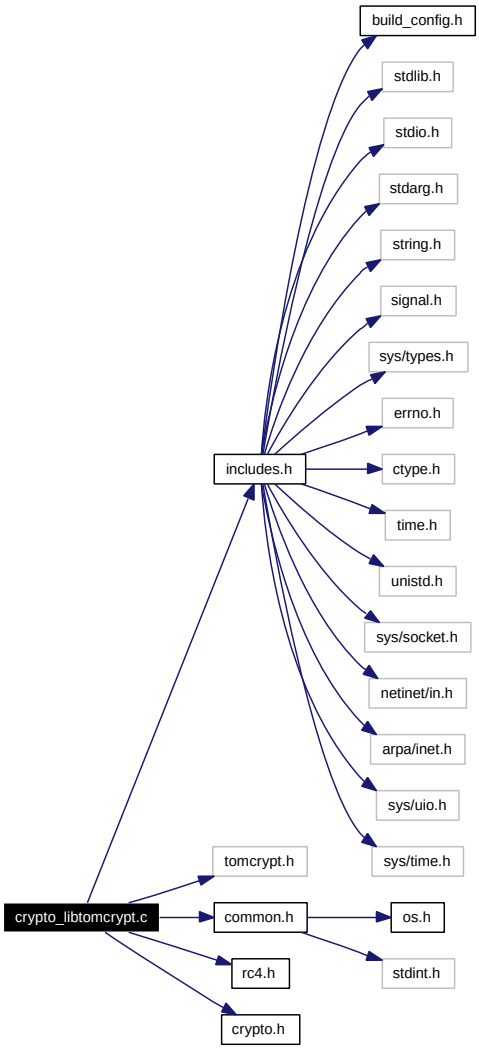
Definition in file crypto_internal.c.

## 6.27 crypto_libtomcrypt.c File Reference

WPA Supplicant / Crypto wrapper for LibTomCrypt (for internal TLSv1).

```
#include "includes.h"
#include <tomcrypt.h>
#include "common.h"
#include "rc4.h"
#include "crypto.h"
```

Include dependency graph for crypto_libtomcrypt.c:




### Defines

- #define **mp_init_multi** ltc_init_multi
- #define **mp_clear_multi** ltc_deinit_multi
- #define **mp_unsigned_bin_size**(a) ltc_mp.unsigned_size(a)

- #define **mp_to_unsigned_bin**(a, b) ltc_mp.unsigned_write(a, b)
- #define **mp_read_unsigned_bin**(a, b, c) ltc_mp.unsigned_read(a, b, c)
- #define **mp_exptmod**(a, b, c, d) ltc_mp.exptmod(a,b,c,d)

## Functions

- void md4_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  *MD4 hash for data vector.*
- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)
  *Encrypt one block with DES.*

### 6.27.1 Detailed Description

WPA Supplicant / Crypto wrapper for LibTomCrypt (for internal TLSv1).

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file crypto_libtomcrypt.c.

### 6.27.2 Function Documentation

#### 6.27.2.1 void des_encrypt (const u8 * *clear*, const u8 * *key*, u8 * *cypher*)

Encrypt one block with DES.

**Parameters:**
***clear*** 8 octets (in)
***key*** 7 octets (in) (no parity bits included)
***cypher*** 8 octets (out)

Definition at line 45 of file crypto_libtomcrypt.c.

#### 6.27.2.2 void md4_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

MD4 hash for data vector.

**Parameters:**
***num_elem*** Number of elements in the data vector
***addr*** Pointers to the data areas
***len*** Lengths of the data blocks

***mac*** Buffer for the hash

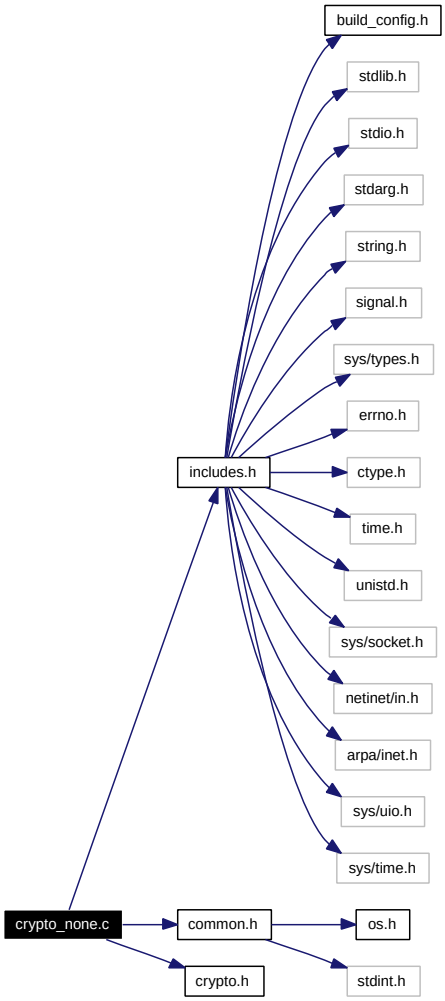
Definition at line 33 of file crypto_libtomcrypt.c.

# 6.28 crypto_none.c File Reference

WPA Supplicant / Empty template functions for crypto wrapper.

```
#include "includes.h"
#include "common.h"
#include "crypto.h"
```

Include dependency graph for crypto_none.c:




## Functions

- void md4_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *MD4 hash for data vector.*

- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)

  *Encrypt one block with DES.*

### 6.28.1 Detailed Description

WPA Supplicant / Empty template functions for crypto wrapper.

**Copyright**
Copyright (c) 2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file crypto_none.c.

### 6.28.2 Function Documentation

#### 6.28.2.1 void des_encrypt (const u8 * *clear*, const u8 * *key*, u8 * *cypher*)

Encrypt one block with DES.

**Parameters:**
***clear*** 8 octets (in)
***key*** 7 octets (in) (no parity bits included)
***cypher*** 8 octets (out)

Definition at line 27 of file crypto_none.c.

#### 6.28.2.2 void md4_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

MD4 hash for data vector.

**Parameters:**
***num_elem*** Number of elements in the data vector
***addr*** Pointers to the data areas
***len*** Lengths of the data blocks
***mac*** Buffer for the hash

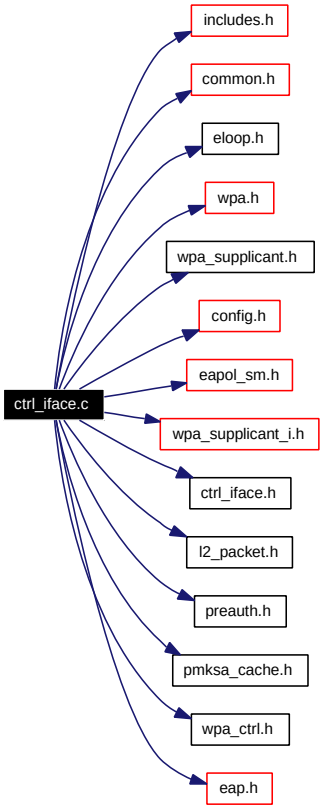
Definition at line 22 of file crypto_none.c.

## 6.29 ctrl_iface.c File Reference

WPA Supplicant / Control interface (shared code for all backends).

```
#include "includes.h"
#include "common.h"
#include "eloop.h"
#include "wpa.h"
#include "wpa_supplicant.h"
#include "config.h"
#include "eapol_sm.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface.h"
#include "l2_packet.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_ctrl.h"
#include "eap.h"
```

Include dependency graph for ctrl_iface.c:



## Functions

- char * wpa_supplicant_ctrl_iface_process (struct wpa_supplicant *wpa_s, char *buf, size_t *resp_-len)

  *Process ctrl_iface command.*

- char * wpa_supplicant_global_ctrl_iface_process (struct wpa_global *global, char *buf, size_-t *resp_len)

  *Process global ctrl_iface command.*

### 6.29.1 Detailed Description

WPA Supplicant / Control interface (shared code for all backends).

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ctrl_iface.c.

### 6.29.2 Function Documentation

#### 6.29.2.1 char* wpa_supplicant_ctrl_iface_process (struct wpa_supplicant * *wpa_s*, char * *buf*, size_t * *resp_len*)

Process ctrl_iface command.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

***buf*** Received command buffer (nul terminated string)

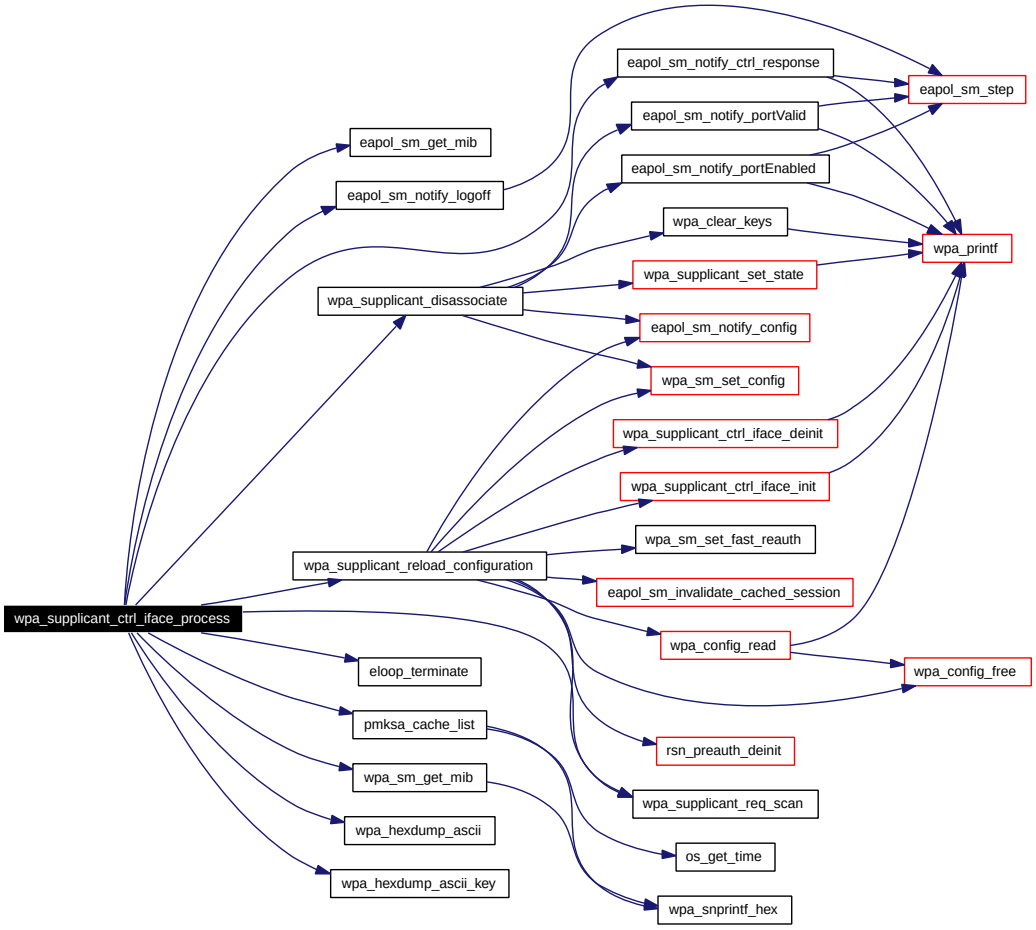
***resp_len*** Variable to be set to the response length

**Returns:**
Response (*resp_len bytes) or NULL on failure

Control interface backends call this function when receiving a message that they do not process internally, i.e., anything else than ATTACH, DETACH, and LEVEL. The return response value is then sent to the external program that sent the command. Caller is responsible for freeing the buffer after this. If NULL is returned, *resp_len can be set to two special values: 1 = send "FAIL\n" response, 2 = send "OK\n" response. If *resp_len has any other value, no response is sent.

Definition at line 1068 of file ctrl_iface.c.

Here is the call graph for this function:




### 6.29.2.2 char* wpa_supplicant_global_ctrl_iface_process (struct wpa_global * *global*, char * *buf*, size_t * *resp_len*)

Process global ctrl_iface command.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

***buf*** Received command buffer (nul terminated string)

***resp_len*** Variable to be set to the response length

**Returns:**

Response (*resp_len bytes) or NULL on failure

Control interface backends call this function when receiving a message from the global ctrl_iface connection. The return response value is then sent to the external program that sent the command. Caller is responsible for freeing the buffer after this. If NULL is returned, *resp_len can be set to two special values: 1 = send "FAIL\n" response, 2 = send "OK\n" response. If *resp_len has any other value, no response is sent.

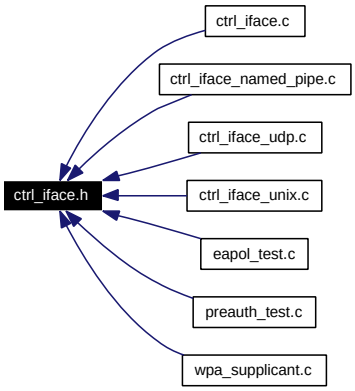
Definition at line 1325 of file ctrl_iface.c.

Here is the call graph for this function:

# 6.30 ctrl_iface.h File Reference

WPA Supplicant / UNIX domain socket -based control interface.

This graph shows which files directly or indirectly include this file:




## Functions

- char * wpa_supplicant_ctrl_iface_process (struct wpa_supplicant *wpa_s, char *buf, size_t *resp_-len)

  *Process ctrl_iface command.*

- char * wpa_supplicant_global_ctrl_iface_process (struct wpa_global *global, char *buf, size_-t *resp_len)

  *Process global ctrl_iface command.*

- ctrl_iface_priv * wpa_supplicant_ctrl_iface_init (struct wpa_supplicant *wpa_s)

  *Initialize control interface.*

- void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv *priv)

  *Deinitialize control interface.*

- void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv *priv)

  *Wait for ctrl_iface monitor.*

- ctrl_iface_global_priv * wpa_supplicant_global_ctrl_iface_init (struct wpa_global *global)

  *Initialize global control interface.*

- void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv *priv)

  *Deinitialize global ctrl interface.*

### 6.30.1 Detailed Description

WPA Supplicant / UNIX domain socket -based control interface.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ctrl_iface.h.

## 6.30.2 Function Documentation

### 6.30.2.1 void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv * *priv*)

Deinitialize control interface.

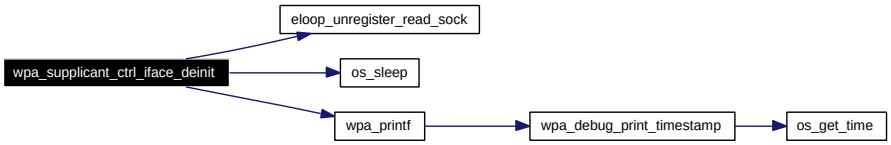
**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_ctrl_iface_init()

Deinitialize the control interface that was initialized with wpa_supplicant_ctrl_iface_init().

Required to be implemented in each control interface backend.

Definition at line 472 of file ctrl_iface_named_pipe.c.

Here is the call graph for this function:




### 6.30.2.2 struct ctrl_iface_priv* wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * *wpa_s*)

Initialize control interface.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

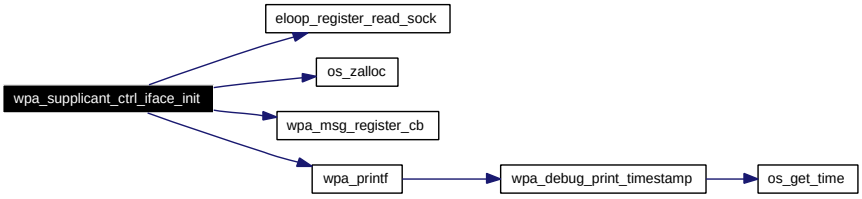
**Returns:**
Pointer to private data on success, NULL on failure

Initialize the control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

Definition at line 444 of file ctrl_iface_named_pipe.c.

Here is the call graph for this function:




#### 6.30.2.3 char* wpa_supplicant_ctrl_iface_process (struct wpa_supplicant * *wpa_s*, char * *buf*, size_t * *resp_len*)

Process ctrl_iface command.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***buf*** Received command buffer (nul terminated string)

***resp_len*** Variable to be set to the response length

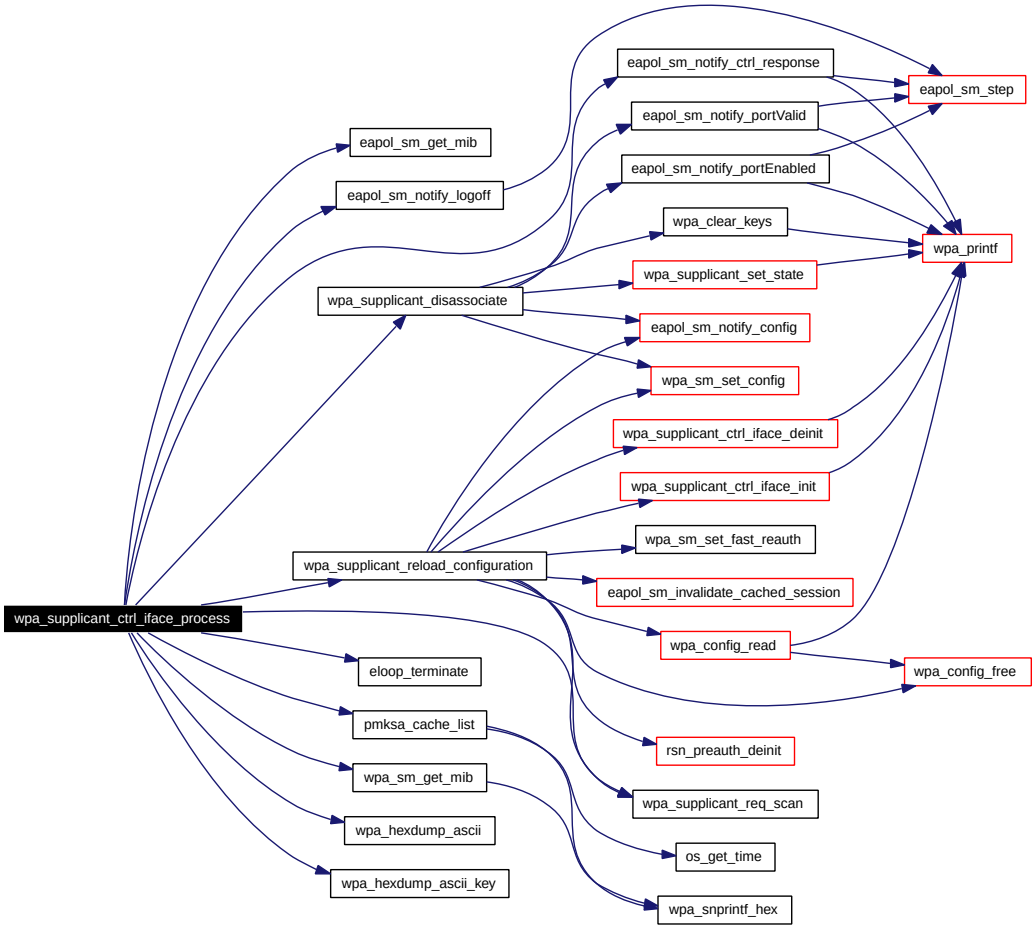
**Returns:**

Response (*resp_len bytes) or NULL on failure

Control interface backends call this function when receiving a message that they do not process internally, i.e., anything else than ATTACH, DETACH, and LEVEL. The return response value is then sent to the external program that sent the command. Caller is responsible for freeing the buffer after this. If NULL is returned, *resp_len can be set to two special values: 1 = send "FAIL\n" response, 2 = send "OK\n" response. If *resp_len has any other value, no response is sent.

Definition at line 1068 of file ctrl_iface.c.

Here is the call graph for this function:




### 6.30.2.4 void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv * *priv*)

Wait for ctrl_iface monitor.

**Parameters:**

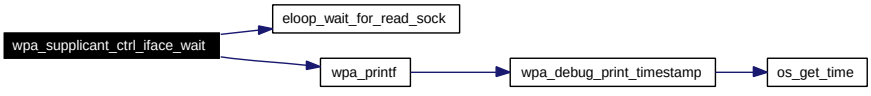
***priv*** Pointer to private data from wpa_supplicant_ctrl_iface_init()

Wait until the first message from an external program using the control interface is received. This function can be used to delay normal startup processing to allow control interface programs to attach with wpa_-supplicant before normal operations are started.

Required to be implemented in each control interface backend.

Definition at line 531 of file ctrl_iface_named_pipe.c.

Here is the call graph for this function:



#### 6.30.2.5 void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv * *priv*)

Deinitialize global ctrl interface.

**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_global_ctrl_iface_init()

Deinitialize the global control interface that was initialized with wpa_supplicant_global_ctrl_iface_init().

Required to be implemented in each control interface backend.

Definition at line 831 of file ctrl_iface_named_pipe.c.

Here is the call graph for this function:


#### 6.30.2.6 struct ctrl_iface_global_priv* wpa_supplicant_global_ctrl_iface_init (struct wpa_global * *global*)

Initialize global control interface.

**Parameters:**
***global*** Pointer to global data from wpa_supplicant_init()

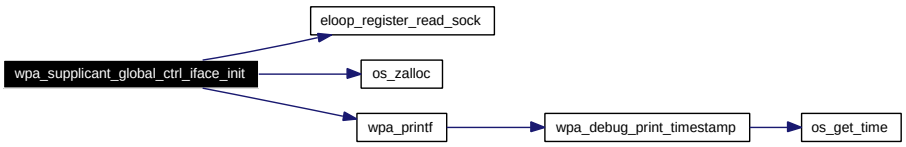
**Returns:**
Pointer to private data on success, NULL on failure

Initialize the global control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

Definition at line 812 of file ctrl_iface_named_pipe.c.

Here is the call graph for this function:




#### 6.30.2.7 char* wpa_supplicant_global_ctrl_iface_process (struct wpa_global * *global*, char * *buf*, size_t * *resp_len*)

Process global ctrl_iface command.

**Parameters:**
***global*** Pointer to global data from wpa_supplicant_init()
***buf*** Received command buffer (nul terminated string)

***resp_len*** Variable to be set to the response length

**Returns:**
Response (∗resp_len bytes) or NULL on failure

Control interface backends call this function when receiving a message from the global ctrl_iface connection. The return response value is then sent to the external program that sent the command. Caller is responsible for freeing the buffer after this. If NULL is returned, ∗resp_len can be set to two special values: 1 = send "FAIL\n" response, 2 = send "OK\n" response. If ∗resp_len has any other value, no response is sent.

Definition at line 1325 of file ctrl_iface.c.

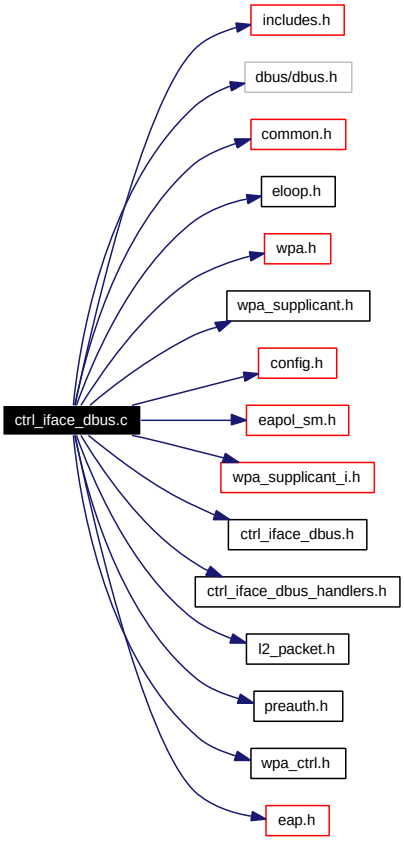
Here is the call graph for this function:

## 6.31 ctrl_iface_dbus.c File Reference

WPA Supplicant / dbus-based control interface.

```
#include "includes.h"
#include <dbus/dbus.h>
#include "common.h"
#include "eloop.h"
#include "wpa.h"
#include "wpa_supplicant.h"
#include "config.h"
#include "eapol_sm.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface_dbus.h"
#include "ctrl_iface_dbus_handlers.h"
#include "l2_packet.h"
#include "preauth.h"
#include "wpa_ctrl.h"
#include "eap.h"
```

Include dependency graph for ctrl_iface_dbus.c:




## Functions

- u32 wpa_supplicant_dbus_next_objid (struct ctrl_iface_dbus_priv *iface)

  *Return next available object id.*

- char * wpas_dbus_decompose_object_path (const char *path, char **network, char **bssid)

  *Decompose an interface object path into parts.*

- DBusMessage * wpas_dbus_new_invalid_iface_error (DBusMessage *message)

  *Return a new invalid interface error message.*

- DBusMessage * wpas_dbus_new_invalid_network_error (DBusMessage *message)

  *Return a new invalid network error message.*

- void wpa_supplicant_dbus_notify_scan_results (struct wpa_supplicant *wpa_s)

  *Send a scan results signal.*

- void wpa_supplicant_dbus_notify_state_change (struct wpa_supplicant *wpa_s, wpa_states new_state, wpa_states old_state)

  *Send a state change signal.*

- ctrl_iface_dbus_priv * wpa_supplicant_dbus_ctrl_iface_init (struct wpa_global *global)

  *Initialize dbus control interface.*

- void wpa_supplicant_dbus_ctrl_iface_deinit (struct ctrl_iface_dbus_priv *iface)
  *Deinitialize dbus ctrl interface.*
- int wpas_dbus_register_iface (struct wpa_supplicant *wpa_s)
  *Register a new interface with dbus.*
- int wpas_dbus_unregister_iface (struct wpa_supplicant *wpa_s)
  *Unregister an interface from dbus.*
- wpa_supplicant * wpa_supplicant_get_iface_by_dbus_path (struct wpa_global *global, const char *path)
  *Get a new network interface.*
- int wpa_supplicant_set_dbus_path (struct wpa_supplicant *wpa_s, const char *path)
  *Assign a dbus path to an interface.*
- const char * wpa_supplicant_get_dbus_path (struct wpa_supplicant *wpa_s)
  *Get an interface's dbus path.*

## 6.31.1 Detailed Description

WPA Supplicant / dbus-based control interface.

**Copyright**
Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ctrl_iface_dbus.c.

## 6.31.2 Function Documentation

### 6.31.2.1 void wpa_supplicant_dbus_ctrl_iface_deinit (struct ctrl_iface_dbus_priv * *iface*)

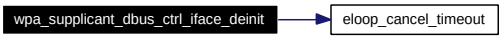
Deinitialize dbus ctrl interface.

**Parameters:**
***iface*** Pointer to dbus private data from wpa_supplicant_dbus_ctrl_iface_init()

Deinitialize the dbus control interface that was initialized with wpa_supplicant_dbus_ctrl_iface_init().

Definition at line 906 of file ctrl_iface_dbus.c.

Here is the call graph for this function:



#### 6.31.2.2 struct ctrl_iface_dbus_priv* wpa_supplicant_dbus_ctrl_iface_init (struct wpa_global * *global*)

Initialize dbus control interface.

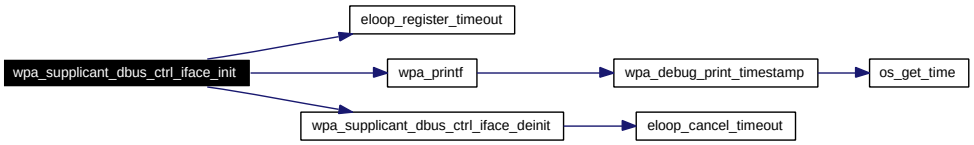
**Parameters:**
***global*** Pointer to global data from wpa_supplicant_init()

**Returns:**
Pointer to dbus_ctrl_iface date or NULL on failure

Initialize the dbus control interface and start receiving commands from external programs over the bus.

Definition at line 812 of file ctrl_iface_dbus.c.

Here is the call graph for this function:




#### 6.31.2.3 u32 wpa_supplicant_dbus_next_objid (struct ctrl_iface_dbus_priv * *iface*)

Return next available object id.

**Parameters:**
***iface*** dbus control interface private data

**Returns:**
Object id

Definition at line 273 of file ctrl_iface_dbus.c.

#### 6.31.2.4 void wpa_supplicant_dbus_notify_scan_results (struct wpa_supplicant * *wpa_s*)

Send a scan results signal.

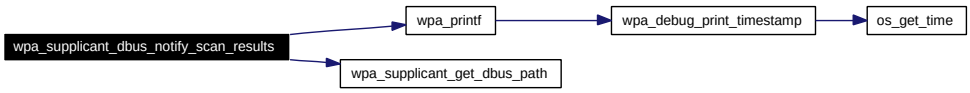
**Parameters:**
***wpa_s*** wpa_supplicant network interface data

**Returns:**
0 on success, -1 on failure

Notify listeners that this interface has updated scan results.

Definition at line 631 of file ctrl_iface_dbus.c.

Here is the call graph for this function:




#### 6.31.2.5 void wpa_supplicant_dbus_notify_state_change (struct wpa_supplicant * *wpa_s*, wpa_states *new_state*, wpa_states *old_state*)

Send a state change signal.

**Parameters:**

***wpa_s*** wpa_supplicant network interface data

***new_state*** new state wpa_supplicant is entering

***old_state*** old state wpa_supplicant is leaving

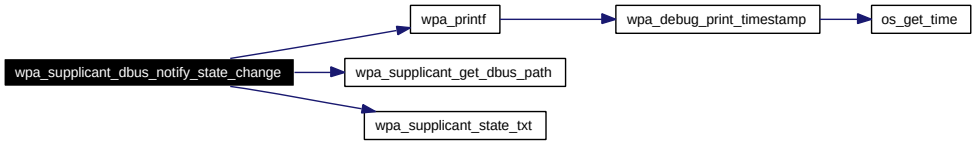
**Returns:**

0 on success, -1 on failure

Notify listeners that wpa_supplicant has changed state

Definition at line 674 of file ctrl_iface_dbus.c.

Here is the call graph for this function:




#### 6.31.2.6 const char* wpa_supplicant_get_dbus_path (struct wpa_supplicant * *wpa_s*)

Get an interface's dbus path.

**Parameters:**

***wpa_s*** wpa_supplicant interface structure

**Returns:**

Interface's dbus object path, or NULL on error

Definition at line 1061 of file ctrl_iface_dbus.c.

#### 6.31.2.7 struct wpa_supplicant* wpa_supplicant_get_iface_by_dbus_path (struct wpa_global * *global*, const char * *path*)

Get a new network interface.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

***path*** Pointer to a dbus object path representing an interface

**Returns:**
Pointer to the interface or NULL if not found

Definition at line 1022 of file ctrl_iface_dbus.c.

#### 6.31.2.8 int wpa_supplicant_set_dbus_path (struct wpa_supplicant * *wpa_s*, const char * *path*)

Assign a dbus path to an interface.

**Parameters:**
***wpa_s*** wpa_supplicant interface structure

***path*** dbus path to set on the interface

**Returns:**
0 on succes, -1 on error

Definition at line 1042 of file ctrl_iface_dbus.c.

#### 6.31.2.9 char* wpas_dbus_decompose_object_path (const char * *path*, char ** *network*, char ** *bssid*)

Decompose an interface object path into parts.

**Parameters:**
***path*** The dbus object path

***network*** (out) the configured network this object path refers to, if any

***bssid*** (out) the scanned bssid this object path refers to, if any

**Returns:**
The object path of the network interface this path refers to

For a given object path, decomposes the object path into object id, network, and BSSID parts, if those parts exist.

Definition at line 290 of file ctrl_iface_dbus.c.

#### 6.31.2.10 DBusMessage* wpas_dbus_new_invalid_iface_error (DBusMessage * *message*)

Return a new invalid interface error message.

**Parameters:**
***message*** Pointer to incoming dbus message this error refers to

**Returns:**
A dbus error message

Convenience function to create and return an invalid interface error

Definition at line 351 of file ctrl_iface_dbus.c.

#### 6.31.2.11 DBusMessage* wpas_dbus_new_invalid_network_error (DBusMessage * *message*)

Return a new invalid network error message.

**Parameters:**
***message*** Pointer to incoming dbus message this error refers to

**Returns:**
a dbus error message

Convenience function to create and return an invalid network error

Definition at line 367 of file ctrl_iface_dbus.c.

#### 6.31.2.12 int wpas_dbus_register_iface (struct wpa_supplicant * *wpa_s*)

Register a new interface with dbus.

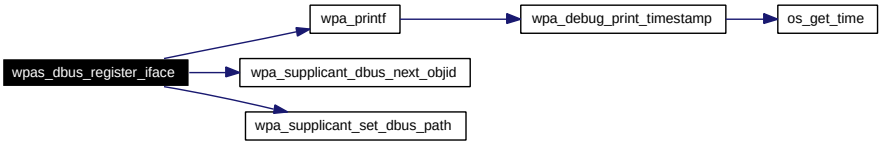
**Parameters:**
***global*** Global wpa_supplicant data
***wpa_s*** wpa_supplicant interface description structure to register

**Returns:**
0 on success, -1 on error

Registers a new interface with dbus and assigns it a dbus object path.

Definition at line 935 of file ctrl_iface_dbus.c.

Here is the call graph for this function:




#### 6.31.2.13 int wpas_dbus_unregister_iface (struct wpa_supplicant * *wpa_s*)

Unregister an interface from dbus.

**Parameters:**
***wpa_s*** wpa_supplicant interface structure

**Returns:**
0 on success, -1 on failure

Unregisters the interface with dbus

Definition at line 991 of file ctrl_iface_dbus.c.

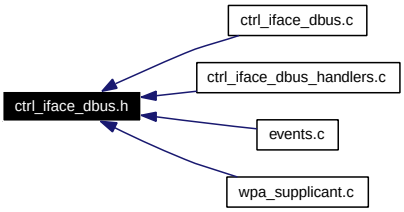
Here is the call graph for this function:


wpas_dbus_unregister_iface
wpa_supplicant_get_dbus_path

# 6.32 ctrl_iface_dbus.h File Reference

WPA Supplicant / dbus-based control interface.

This graph shows which files directly or indirectly include this file:




## 6.32.1 Detailed Description

WPA Supplicant / dbus-based control interface.

**Copyright**

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

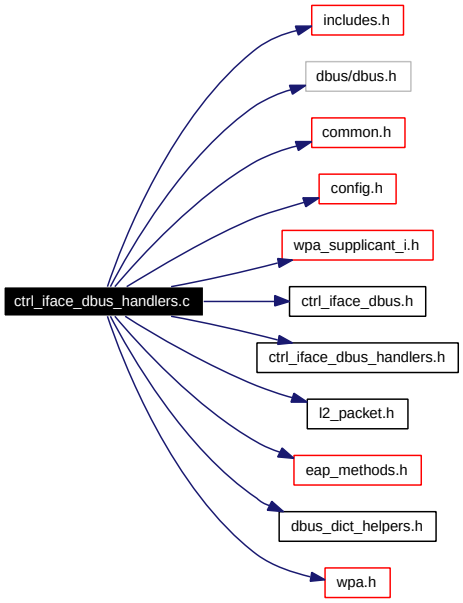
Definition in file ctrl_iface_dbus.h.

## 6.33 ctrl_iface_dbus_handlers.c File Reference

WPA Supplicant / dbus-based control interface.

```
#include "includes.h"
#include <dbus/dbus.h>
#include "common.h"
#include "config.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface_dbus.h"
#include "ctrl_iface_dbus_handlers.h"
#include "l2_packet.h"
#include "eap_methods.h"
#include "dbus_dict_helpers.h"
#include "wpa.h"
```

Include dependency graph for ctrl_iface_dbus_handlers.c:




### Functions

- DBusMessage * wpas_dbus_global_add_interface (DBusMessage *message, struct wpa_global *global)

  *Request registration of a network interface.*

- DBusMessage * wpas_dbus_global_remove_interface (DBusMessage *message, struct wpa_global *global)

  *Request deregistration of an interface.*

- DBusMessage * wpas_dbus_global_get_interface (DBusMessage *message, struct wpa_global *global)

  *Get the object path for an interface name.*

- DBusMessage * wpas_dbus_iface_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Request a wireless scan on an interface.*

- DBusMessage * wpas_dbus_iface_scan_results (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Get the results of a recent scan request.*

- DBusMessage * wpas_dbus_bssid_properties (DBusMessage *message, struct wpa_supplicant *wpa_s, struct wpa_scan_result *res)

  *Return the properties of a scanned network.*

- DBusMessage * wpas_dbus_iface_capabilities (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Return interface capabilities.*

- DBusMessage * wpas_dbus_iface_add_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Add a new configured network.*

- DBusMessage * wpas_dbus_iface_remove_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Remove a configured network.*

- DBusMessage * wpas_dbus_iface_set_network (DBusMessage *message, struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  *Set options for a configured network.*

- DBusMessage * wpas_dbus_iface_enable_network (DBusMessage *message, struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  *Mark a configured network as enabled.*

- DBusMessage * wpas_dbus_iface_disable_network (DBusMessage *message, struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  *Mark a configured network as disabled.*

- DBusMessage * wpas_dbus_iface_select_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Attempt association with a configured network.*

- DBusMessage * wpas_dbus_iface_disconnect (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Terminate the current connection.*

- DBusMessage * wpas_dbus_iface_set_ap_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Control roaming mode.*

- DBusMessage * wpas_dbus_iface_get_state (DBusMessage *message, struct wpa_supplicant *wpa_s)

  *Get interface state.*

### 6.33.1 Detailed Description

WPA Supplicant / dbus-based control interface.

**Copyright**

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ctrl_iface_dbus_handlers.c.

### 6.33.2 Function Documentation

#### 6.33.2.1 DBusMessage* wpas_dbus_bssid_properties (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*, struct wpa_scan_result * *res*)

Return the properties of a scanned network.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

***res*** wpa_supplicant scan result for which to get properties

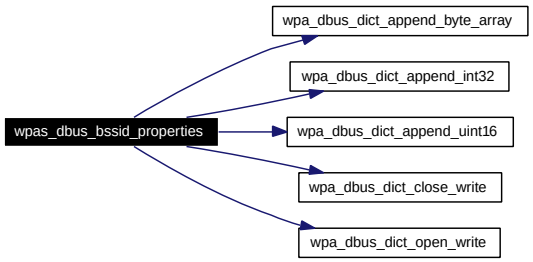
**Returns:**

a dbus message containing the properties for the requested network

Handler function for "properties" method call of a scanned network. Returns a dbus message containing the the properties.

Definition at line 389 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:



#### 6.33.2.2 DBusMessage* wpas_dbus_global_add_interface (DBusMessage * *message*, struct wpa_global * *global*)

Request registration of a network interface.

**Parameters:**

***message*** Pointer to incoming dbus message

***global*** wpa_supplicant global data structure

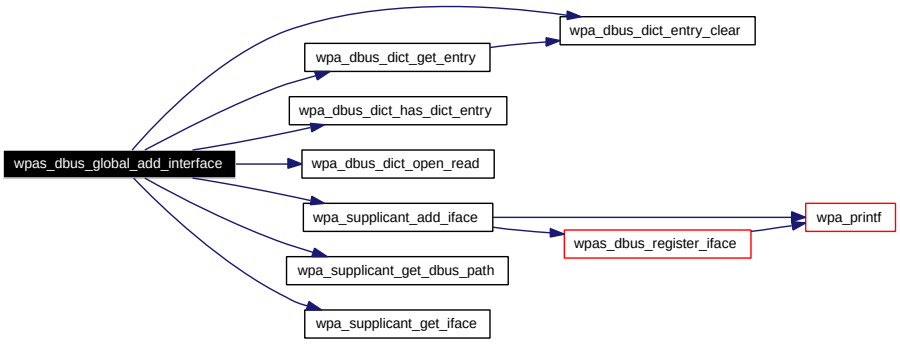
**Returns:**

The object path of the new interface object, or a dbus error message with more information

Handler function for "addInterface" method call. Handles requests by dbus clients to register a network interface that wpa_supplicant will manage.

Definition at line 97 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.3 DBusMessage* wpas_dbus_global_get_interface (DBusMessage * *message*, struct wpa_global * *global*)

Get the object path for an interface name.

**Parameters:**

***message*** Pointer to incoming dbus message

***global*** wpa_supplicant global data structure

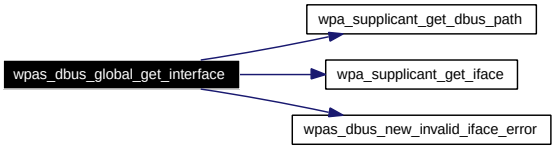
**Returns:**

The object path of the interface object, or a dbus error message with more information

Handler function for "getInterface" method call. Handles requests by dbus clients for the object path of an specific network interface.

Definition at line 248 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.4 DBusMessage* wpas_dbus_global_remove_interface (DBusMessage * *message*, struct wpa_global * *global*)

Request deregistration of an interface.

**Parameters:**

***message*** Pointer to incoming dbus message

***global*** wpa_supplicant global data structure

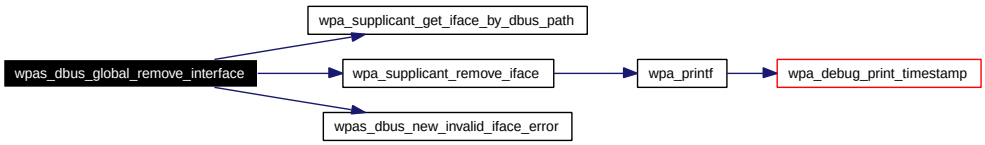
**Returns:**

a dbus message containing a UINT32 indicating success (1) or failure (0), or returns a dbus error message with more information

Handler function for "removeInterface" method call. Handles requests by dbus clients to deregister a network interface that wpa_supplicant currently manages.

Definition at line 203 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.5 DBusMessage* wpas_dbus_iface_add_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Add a new configured network.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

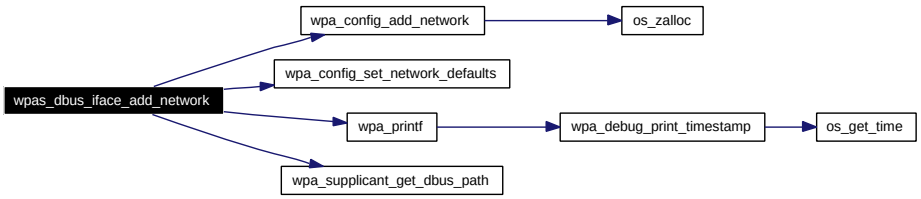
**Returns:**

A dbus message containing the object path of the new network

Handler function for "addNetwork" method call of a network interface.

Definition at line 761 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.6 DBusMessage* wpas_dbus_iface_capabilities (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Return interface capabilities.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

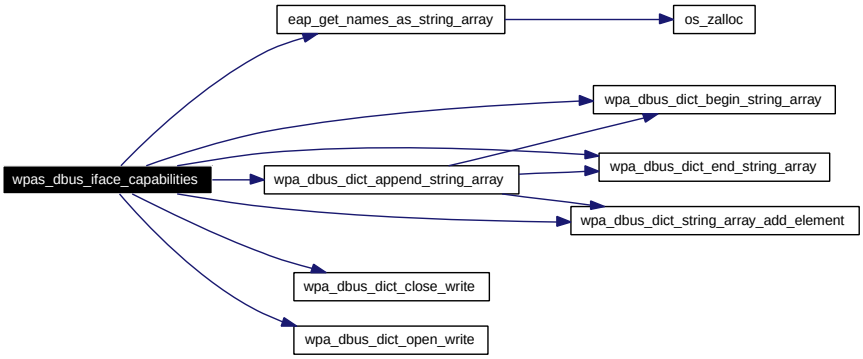
**Returns:**

A dbus message containing a dict of strings

Handler function for "capabilities" method call of an interface.

Definition at line 472 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.7 DBusMessage* wpas_dbus_iface_disable_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Mark a configured network as disabled.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

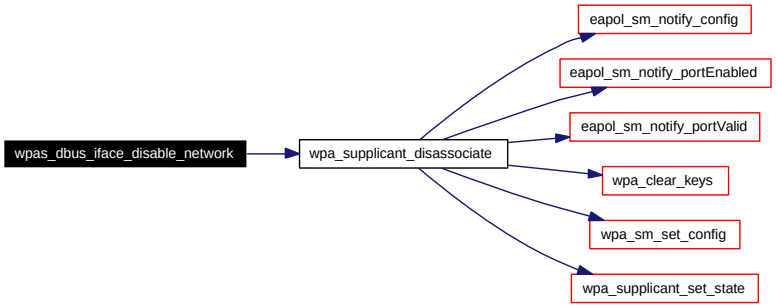
***ssid*** wpa_ssid structure for a configured network

**Returns:**
A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "disable" method call of a configured network.

Definition at line 1041 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.8 DBusMessage* wpas_dbus_iface_disconnect (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Terminate the current connection.

**Parameters:**
***message*** Pointer to incoming dbus message

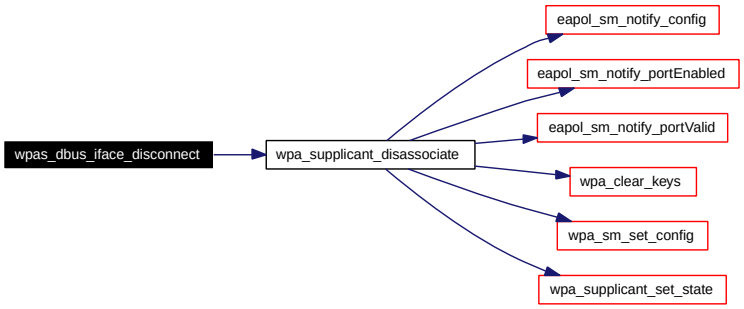
***wpa_s*** wpa_supplicant structure for a network interface

**Returns:**
A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "disconnect" method call of network interface.

Definition at line 1156 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:



#### 6.33.2.9 DBusMessage* wpas_dbus_iface_enable_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Mark a configured network as enabled.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

***ssid*** wpa_ssid structure for a configured network

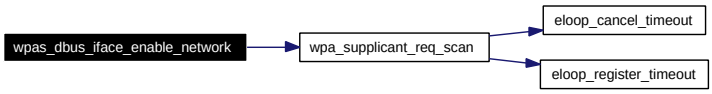
**Returns:**

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "enable" method call of a configured network.

Definition at line 1012 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.10 DBusMessage* wpas_dbus_iface_get_state (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface state.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

**Returns:**

A dbus message containing a STRING representing the current interface state

Handler function for "state" method call.

Definition at line 1210 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:


#### 6.33.2.11 DBusMessage* wpas_dbus_iface_remove_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Remove a configured network.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

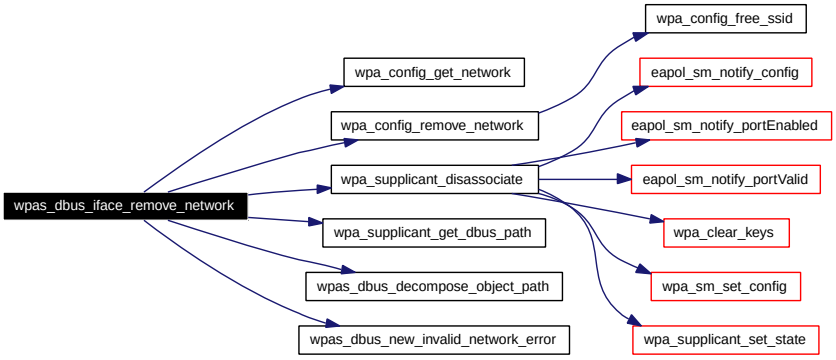
**Returns:**

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "removeNetwork" method call of a network interface.

Definition at line 815 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




### 6.33.2.12 DBusMessage* wpas_dbus_iface_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Request a wireless scan on an interface.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

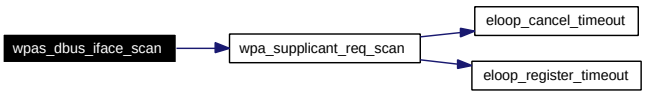
**Returns:**

a dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "scan" method call of a network device. Requests that wpa_supplicant perform a wireless scan as soon as possible on a particular wireless interface.

Definition at line 300 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:



#### 6.33.2.13 DBusMessage* wpas_dbus_iface_scan_results (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get the results of a recent scan request.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

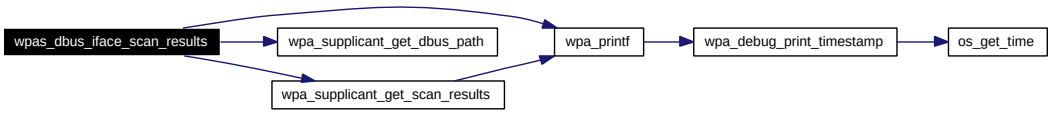
**Returns:**

a dbus message containing a dbus array of objects paths, or returns a dbus error message if not scan results could be found

Handler function for "scanResults" method call of a network device. Returns a dbus message containing the object paths of wireless networks found.

Definition at line 320 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.14 DBusMessage* wpas_dbus_iface_select_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Attempt association with a configured network.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

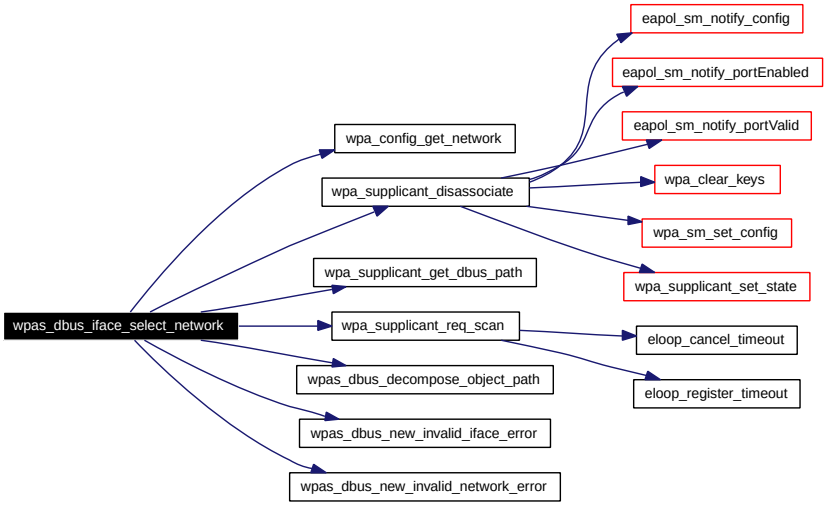
**Returns:**

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "selectNetwork" method call of network interface.

Definition at line 1063 of file ctrl_iface_dbus_handlers.c.

Here is the call graph for this function:




#### 6.33.2.15 DBusMessage* wpas_dbus_iface_set_ap_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Control roaming mode.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

**Returns:**

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "setAPScan" method call.

Definition at line 1176 of file ctrl_iface_dbus_handlers.c.

#### 6.33.2.16 DBusMessage* wpas_dbus_iface_set_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Set options for a configured network.

**Parameters:**

***message*** Pointer to incoming dbus message

***wpa_s*** wpa_supplicant structure for a network interface

***ssid*** wpa_ssid structure for a configured network

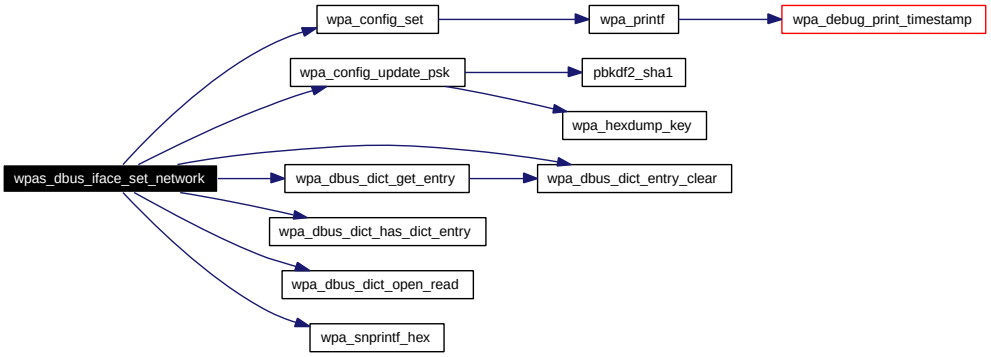
**Returns:**

a dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "set" method call of a configured network.

Definition at line 897 of file ctrl_iface_dbus_handlers.c.

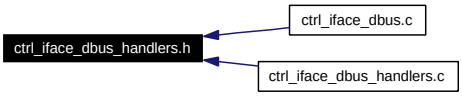
Here is the call graph for this function:



## 6.34 ctrl_iface_dbus_handlers.h File Reference

WPA Supplicant / dbus-based control interface.

This graph shows which files directly or indirectly include this file:




### 6.34.1 Detailed Description

WPA Supplicant / dbus-based control interface.

**Copyright**

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

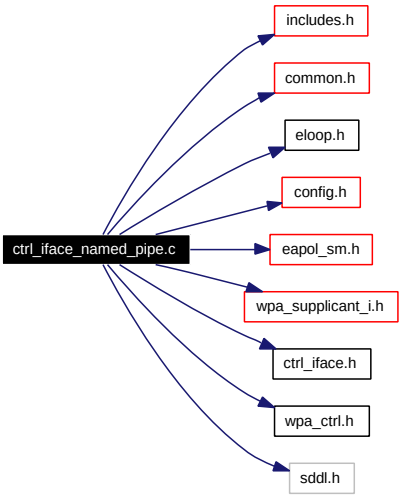
Definition in file ctrl_iface_dbus_handlers.h.

# 6.35 ctrl_iface_named_pipe.c File Reference

WPA Supplicant / Windows Named Pipe -based control interface.

```
#include "includes.h"
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "eapol_sm.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface.h"
#include "wpa_ctrl.h"
#include <sddl.h>
```

Include dependency graph for ctrl_iface_named_pipe.c:




## Defines

- #define **_WIN32_WINNT** 0x0500
- #define **WPA_SUPPLICANT_NAMED_PIPE** "WpaSupplicant"
- #define **NAMED_PIPE_PREFIX** TEXT("\\\\.\\pipe\\") TEXT(WPA_SUPPLICANT_-NAMED_PIPE)
- #define **REQUEST_BUFSIZE** 256
- #define **REPLY_BUFSIZE** 4096

## Functions

- ctrl_iface_priv * wpa_supplicant_ctrl_iface_init (struct wpa_supplicant *wpa_s)

  *Initialize control interface.*

- void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv *priv)

*Deinitialize control interface.*

- void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv *priv)

  *Wait for ctrl_iface monitor.*

- ctrl_iface_global_priv * wpa_supplicant_global_ctrl_iface_init (struct wpa_global *global)

  *Initialize global control interface.*

- void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv *priv)

  *Deinitialize global ctrl interface.*

## 6.35.1 Detailed Description

WPA Supplicant / Windows Named Pipe -based control interface.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ctrl_iface_named_pipe.c.

## 6.35.2 Function Documentation

### 6.35.2.1 void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv * *priv*)

Deinitialize control interface.

**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_ctrl_iface_init()

Deinitialize the control interface that was initialized with wpa_supplicant_ctrl_iface_init().

Required to be implemented in each control interface backend.

Definition at line 472 of file ctrl_iface_named_pipe.c.

### 6.35.2.2 struct ctrl_iface_priv* wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * *wpa_s*)

Initialize control interface.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

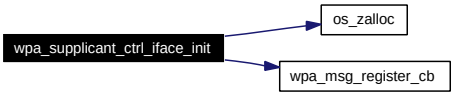
**Returns:**
Pointer to private data on success, NULL on failure

Initialize the control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

Definition at line 444 of file ctrl_iface_named_pipe.c.

Here is the call graph for this function:




#### 6.35.2.3 void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv * *priv*)

Wait for ctrl_iface monitor.

**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_ctrl_iface_init()

Wait until the first message from an external program using the control interface is received. This function can be used to delay normal startup processing to allow control interface programs to attach with wpa_supplicant before normal operations are started.

Required to be implemented in each control interface backend.

Definition at line 531 of file ctrl_iface_named_pipe.c.

Here is the call graph for this function:


#### 6.35.2.4 void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv * *priv*)

Deinitialize global ctrl interface.

**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_global_ctrl_iface_init()

Deinitialize the global control interface that was initialized with wpa_supplicant_global_ctrl_iface_init().

Required to be implemented in each control interface backend.

Definition at line 831 of file ctrl_iface_named_pipe.c.

#### 6.35.2.5 struct ctrl_iface_global_priv* wpa_supplicant_global_ctrl_iface_init (struct wpa_global * *global*)

Initialize global control interface.

**Parameters:**
***global*** Pointer to global data from wpa_supplicant_init()

**Returns:**
Pointer to private data on success, NULL on failure

Initialize the global control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

Definition at line 812 of file ctrl_iface_named_pipe.c.

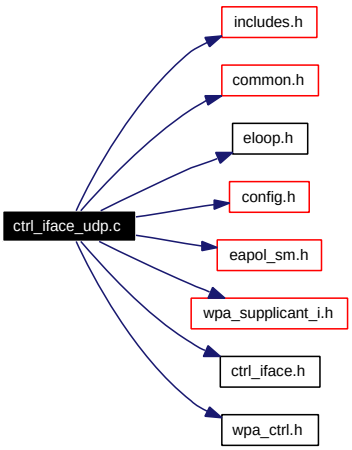
Here is the call graph for this function:

# 6.36 ctrl_iface_udp.c File Reference

WPA Supplicant / UDP socket -based control interface.

```
#include "includes.h"
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "eapol_sm.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface.h"
#include "wpa_ctrl.h"
```

Include dependency graph for ctrl_iface_udp.c:




## Data Structures

- struct wpa_ctrl_dst

  *Internal data structure of control interface clients.*

- struct **ctrl_iface_priv**
- struct **ctrl_iface_global_priv**

## Defines

- #define **COOKIE_LEN** 8

## Functions

- ctrl_iface_priv * wpa_supplicant_ctrl_iface_init (struct wpa_supplicant *wpa_s)

  *Initialize control interface.*

- void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv *priv)
  *Deinitialize control interface.*
- void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv *priv)
  *Wait for ctrl_iface monitor.*
- ctrl_iface_global_priv * wpa_supplicant_global_ctrl_iface_init (struct wpa_global *global)
  *Initialize global control interface.*
- void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv *priv)
  *Deinitialize global ctrl interface.*

### 6.36.1 Detailed Description

WPA Supplicant / UDP socket -based control interface.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ctrl_iface_udp.c.

### 6.36.2 Function Documentation

#### 6.36.2.1 void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv * *priv*)

Deinitialize control interface.

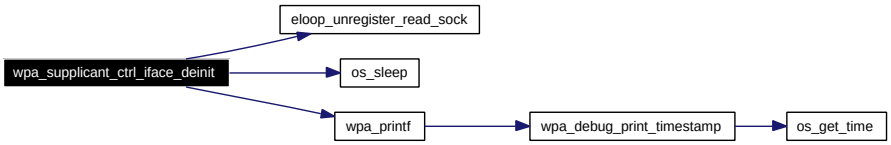
**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_ctrl_iface_init()

Deinitialize the control interface that was initialized with wpa_supplicant_ctrl_iface_init().

Required to be implemented in each control interface backend.

Definition at line 317 of file ctrl_iface_udp.c.

Here is the call graph for this function:



#### 6.36.2.2 struct ctrl_iface_priv* wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * *wpa_s*)

Initialize control interface.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

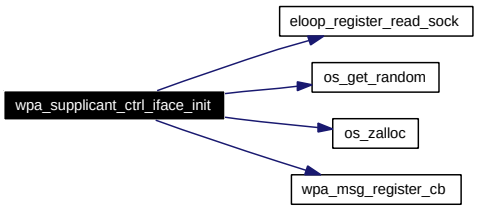
**Returns:**
Pointer to private data on success, NULL on failure

Initialize the control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

Definition at line 273 of file ctrl_iface_udp.c.

Here is the call graph for this function:




#### 6.36.2.3 void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv * *priv*)

Wait for ctrl_iface monitor.

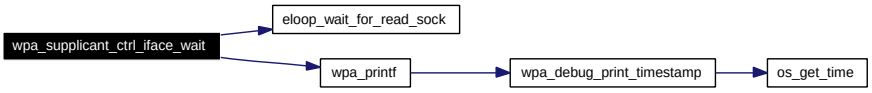
**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_ctrl_iface_init()

Wait until the first message from an external program using the control interface is received. This function can be used to delay normal startup processing to allow control interface programs to attach with wpa_-supplicant before normal operations are started.

Required to be implemented in each control interface backend.

Definition at line 398 of file ctrl_iface_udp.c.

Here is the call graph for this function:




#### 6.36.2.4 void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv * *priv*)

Deinitialize global ctrl interface.

**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_global_ctrl_iface_init()

Deinitialize the global control interface that was initialized with wpa_supplicant_global_ctrl_iface_init().

Required to be implemented in each control interface backend.

Definition at line 556 of file ctrl_iface_udp.c.

Here is the call graph for this function:


#### 6.36.2.5 struct ctrl_iface_global_priv* wpa_supplicant_global_ctrl_iface_init (struct wpa_global * *global*)

Initialize global control interface.

**Parameters:**
***global*** Pointer to global data from wpa_supplicant_init()

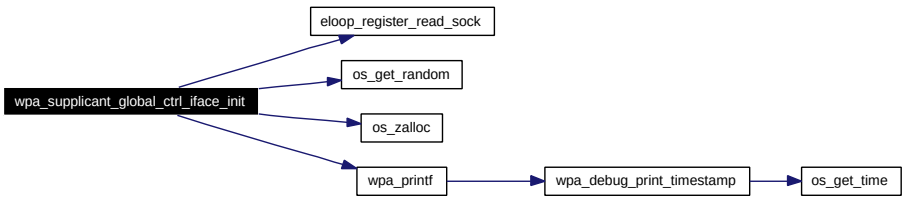
**Returns:**
Pointer to private data on success, NULL on failure

Initialize the global control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

Definition at line 509 of file ctrl_iface_udp.c.

Here is the call graph for this function:

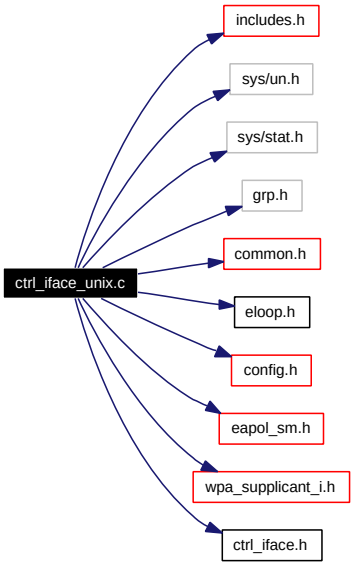


# 6.37 ctrl_iface_unix.c File Reference

WPA Supplicant / UNIX domain socket -based control interface.

```
#include "includes.h"
#include <sys/un.h>
#include <sys/stat.h>
#include <grp.h>
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "eapol_sm.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface.h"
```

Include dependency graph for ctrl_iface_unix.c:




## Data Structures

- struct wpa_ctrl_dst

  *Internal data structure of control interface clients.*

- struct **ctrl_iface_priv**
- struct **ctrl_iface_global_priv**

## Functions

- ctrl_iface_priv * wpa_supplicant_ctrl_iface_init (struct wpa_supplicant *wpa_s)

*Initialize control interface.*

- void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv *priv)
  *Deinitialize control interface.*
- void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv *priv)
  *Wait for ctrl_iface monitor.*
- ctrl_iface_global_priv * wpa_supplicant_global_ctrl_iface_init (struct wpa_global *global)
  *Initialize global control interface.*
- void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv *priv)
  *Deinitialize global ctrl interface.*

### 6.37.1 Detailed Description

WPA Supplicant / UNIX domain socket -based control interface.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ctrl_iface_unix.c.

### 6.37.2 Function Documentation

#### 6.37.2.1 void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv * *priv*)

Deinitialize control interface.

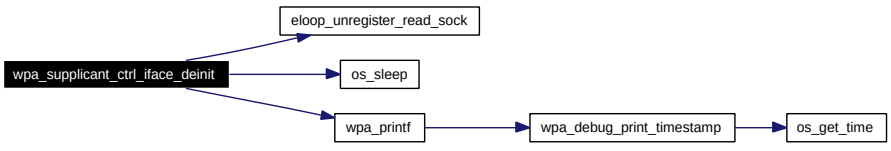
**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_ctrl_iface_init()

Deinitialize the control interface that was initialized with wpa_supplicant_ctrl_iface_init().

Required to be implemented in each control interface backend.

Definition at line 405 of file ctrl_iface_unix.c.

Here is the call graph for this function:



#### 6.37.2.2 struct ctrl_iface_priv* wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * *wpa_s*)

Initialize control interface.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

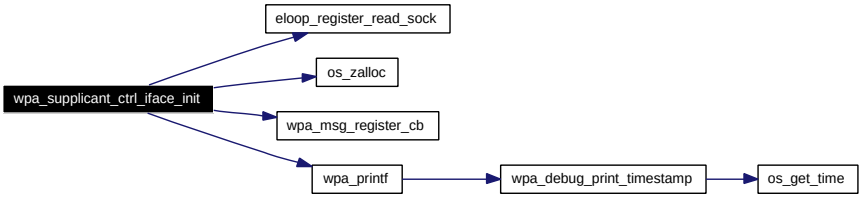
**Returns:**
Pointer to private data on success, NULL on failure

Initialize the control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

Definition at line 253 of file ctrl_iface_unix.c.

Here is the call graph for this function:




#### 6.37.2.3 void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv * *priv*)

Wait for ctrl_iface monitor.

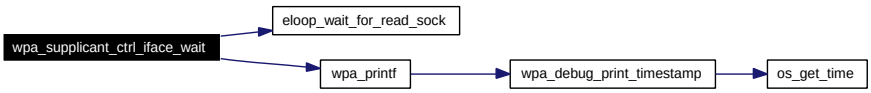
**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_ctrl_iface_init()

Wait until the first message from an external program using the control interface is received. This function can be used to delay normal startup processing to allow control interface programs to attach with wpa_-supplicant before normal operations are started.

Required to be implemented in each control interface backend.

Definition at line 526 of file ctrl_iface_unix.c.

Here is the call graph for this function:




#### 6.37.2.4 void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv * *priv*)

Deinitialize global ctrl interface.

**Parameters:**
***priv*** Pointer to private data from wpa_supplicant_global_ctrl_iface_init()

Deinitialize the global control interface that was initialized with wpa_supplicant_global_ctrl_iface_init().

Required to be implemented in each control interface backend.

Definition at line 650 of file ctrl_iface_unix.c.

Here is the call graph for this function:


#### 6.37.2.5 struct ctrl_iface_global_priv* wpa_supplicant_global_ctrl_iface_init (struct wpa_global * *global*)

Initialize global control interface.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

**Returns:**

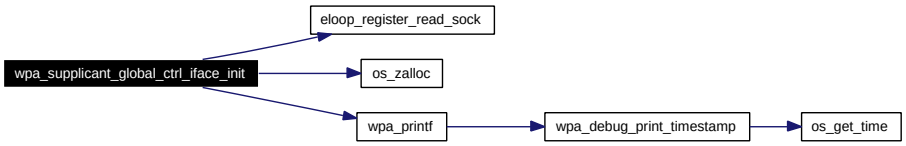
Pointer to private data on success, NULL on failure

Initialize the global control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

Definition at line 576 of file ctrl_iface_unix.c.

Here is the call graph for this function:

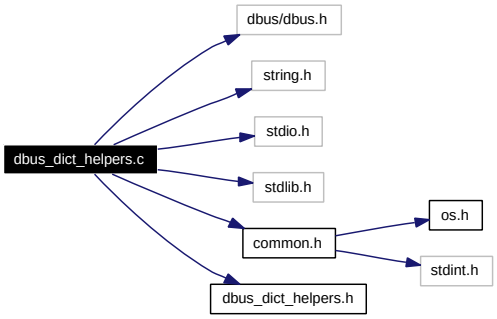


# 6.38 dbus_dict_helpers.c File Reference

WPA Supplicant / dbus-based control interface.

```
#include <dbus/dbus.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "common.h"
#include "dbus_dict_helpers.h"
```

Include dependency graph for dbus_dict_helpers.c:




## Functions

- dbus_bool_t wpa_dbus_dict_open_write (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_close_write (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_append_string (DBusMessageIter *iter_dict, const char *key, const char *value)
- dbus_bool_t wpa_dbus_dict_append_byte (DBusMessageIter *iter_dict, const char *key, const char value)
- dbus_bool_t wpa_dbus_dict_append_bool (DBusMessageIter *iter_dict, const char *key, const dbus_bool_t value)
- dbus_bool_t wpa_dbus_dict_append_int16 (DBusMessageIter *iter_dict, const char *key, const dbus_int16_t value)
- dbus_bool_t wpa_dbus_dict_append_uint16 (DBusMessageIter *iter_dict, const char *key, const dbus_uint16_t value)
- dbus_bool_t wpa_dbus_dict_append_int32 (DBusMessageIter *iter_dict, const char *key, const dbus_int32_t value)
- dbus_bool_t wpa_dbus_dict_append_uint32 (DBusMessageIter *iter_dict, const char *key, const dbus_uint32_t value)
- dbus_bool_t wpa_dbus_dict_append_int64 (DBusMessageIter *iter_dict, const char *key, const dbus_int64_t value)
- dbus_bool_t wpa_dbus_dict_append_uint64 (DBusMessageIter *iter_dict, const char *key, const dbus_uint64_t value)
- dbus_bool_t wpa_dbus_dict_append_double (DBusMessageIter *iter_dict, const char *key, const double value)
- dbus_bool_t wpa_dbus_dict_append_object_path (DBusMessageIter *iter_dict, const char *key, const char *value)

- dbus_bool_t wpa_dbus_dict_append_byte_array (DBusMessageIter *iter_dict, const char *key, const char *value, const dbus_uint32_t value_len)
- dbus_bool_t wpa_dbus_dict_begin_string_array (DBusMessageIter *iter_dict, const char *key, DBusMessageIter *iter_dict_entry, DBusMessageIter *iter_dict_val, DBusMessageIter *iter_array)
- dbus_bool_t wpa_dbus_dict_string_array_add_element (DBusMessageIter *iter_array, const char *elem)
- dbus_bool_t wpa_dbus_dict_end_string_array (DBusMessageIter *iter_dict, DBusMessageIter *iter_dict_entry, DBusMessageIter *iter_dict_val, DBusMessageIter *iter_array)
- dbus_bool_t wpa_dbus_dict_append_string_array (DBusMessageIter *iter_dict, const char *key, const char **items, const dbus_uint32_t num_items)
- dbus_bool_t wpa_dbus_dict_open_read (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_get_entry (DBusMessageIter *iter_dict, struct wpa_dbus_dict_entry *entry)
- dbus_bool_t wpa_dbus_dict_has_dict_entry (DBusMessageIter *iter_dict)
- void wpa_dbus_dict_entry_clear (struct wpa_dbus_dict_entry *entry)

## 6.38.1 Detailed Description

WPA Supplicant / dbus-based control interface.

**Copyright**

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file dbus_dict_helpers.c.

## 6.38.2 Function Documentation

### 6.38.2.1 dbus_bool_t wpa_dbus_dict_append_bool (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_bool_t *value*)

Add a boolean entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The boolean value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 261 of file dbus_dict_helpers.c.

#### 6.38.2.2 dbus_bool_t wpa_dbus_dict_append_byte (DBusMessageIter * *iter_dict*, const char * *key*, const char *value*)

Add a byte entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The byte value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 241 of file dbus_dict_helpers.c.

#### 6.38.2.3 dbus_bool_t wpa_dbus_dict_append_byte_array (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*, const dbus_uint32_t *value_len*)

Add a byte array entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The byte array

***value_len*** The length of the byte array, in bytes

**Returns:**

TRUE on success, FALSE on failure

Definition at line 450 of file dbus_dict_helpers.c.

#### 6.38.2.4 dbus_bool_t wpa_dbus_dict_append_double (DBusMessageIter * *iter_dict*, const char * *key*, const double *value*)

Add a double-precision floating point entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The double-precision floating point value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 407 of file dbus_dict_helpers.c.

#### 6.38.2.5 dbus_bool_t wpa_dbus_dict_append_int16 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int16_t *value*)

Add a 16-bit signed integer entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 16-bit signed integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 281 of file dbus_dict_helpers.c.

#### 6.38.2.6 dbus_bool_t wpa_dbus_dict_append_int32 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int32_t *value*)

Add a 32-bit signed integer to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 32-bit signed integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 323 of file dbus_dict_helpers.c.

#### 6.38.2.7 dbus_bool_t wpa_dbus_dict_append_int64 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int64_t *value*)

Add a 64-bit integer entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 64-bit integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 365 of file dbus_dict_helpers.c.

#### 6.38.2.8 dbus_bool_t wpa_dbus_dict_append_object_path (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*)

Add a DBus object path entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The DBus object path value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 428 of file dbus_dict_helpers.c.

#### 6.38.2.9 dbus_bool_t wpa_dbus_dict_append_string (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*)

Add a string entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The string value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 221 of file dbus_dict_helpers.c.

#### 6.38.2.10 dbus_bool_t wpa_dbus_dict_append_string_array (DBusMessageIter * *iter_dict*, const char * *key*, const char ** *items*, const dbus_uint32_t *num_items*)

Convenience function to add an entire string array to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from nmu_dbus_dict_open_write

***key*** The key of the dict item

***items*** The array of strings

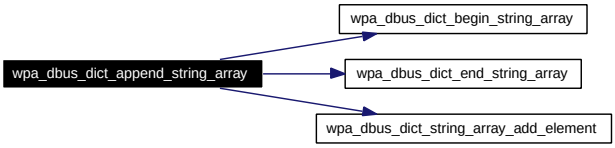
***num_items*** The number of strings in the array

**Returns:**

TRUE on success, FALSE on failure

Definition at line 573 of file dbus_dict_helpers.c.

Here is the call graph for this function:




#### 6.38.2.11 dbus_bool_t wpa_dbus_dict_append_uint16 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint16_t *value*)

Add a 16-bit unsigned integer entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 16-bit unsigned integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 302 of file dbus_dict_helpers.c.

#### 6.38.2.12 dbus_bool_t wpa_dbus_dict_append_uint32 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint32_t *value*)

Add a 32-bit unsigned integer entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 32-bit unsigned integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 344 of file dbus_dict_helpers.c.

#### 6.38.2.13 dbus_bool_t wpa_dbus_dict_append_uint64 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint64_t *value*)

Add a 64-bit unsigned integer entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 64-bit unsigned integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 386 of file dbus_dict_helpers.c.

#### 6.38.2.14 dbus_bool_t wpa_dbus_dict_begin_string_array (DBusMessageIter * *iter_dict*, const char * *key*, DBusMessageIter * *iter_dict_entry*, DBusMessageIter * *iter_dict_val*, DBusMessageIter * *iter_array*)

Begin a string array entry in the dict

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from nmu_dbus_dict_open_write

***key*** The key of the dict item

***iter_dict_entry*** A private DBusMessageIter provided by the caller to be passed to wpa_dbus_dict_-end_string_array

***iter_dict_val*** A private DBusMessageIter provided by the caller to be passed to wpa_dbus_dict_end_-string_array

***iter_array*** On return, the DBusMessageIter to be passed to wpa_dbus_dict_string_array_add_element

**Returns:**

TRUE on success, FALSE on failure

Definition at line 479 of file dbus_dict_helpers.c.

#### 6.38.2.15 dbus_bool_t wpa_dbus_dict_close_write (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

End a dict element in a dbus message. Should be paired with a call to wpa_dbus_dict_open_write.

**Parameters:**

***iter*** valid dbus message iterator, same as passed to wpa_dbus_dict_open_write()

***iter_dict*** a dbus dict iterator returned from wpa_dbus_dict_open_write

**Returns:**

TRUE on success, FALSE on failure

Definition at line 65 of file dbus_dict_helpers.c.

#### 6.38.2.16 dbus_bool_t wpa_dbus_dict_end_string_array (DBusMessageIter * *iter_dict*, DBusMessageIter * *iter_dict_entry*, DBusMessageIter * *iter_dict_val*, DBusMessageIter * *iter_array*)

End a string array dict entry

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from nmu_dbus_dict_open_write

***iter_dict_entry*** A private DBusMessageIter returned from wpa_dbus_dict_end_string_array

***iter_dict_val*** A private DBusMessageIter returned from wpa_dbus_dict_end_string_array

***iter_array*** A DBusMessageIter returned from wpa_dbus_dict_end_string_array

**Returns:**

TRUE on success, FALSE on failure

Definition at line 543 of file dbus_dict_helpers.c.

#### 6.38.2.17 void wpa_dbus_dict_entry_clear (struct wpa_dbus_dict_entry * *entry*)

Free any memory used by the entry object.

**Parameters:**
***entry*** The entry object

Definition at line 939 of file dbus_dict_helpers.c.

#### 6.38.2.18 dbus_bool_t wpa_dbus_dict_get_entry (DBusMessageIter * *iter_dict*, struct wpa_dbus_dict_entry * *entry*)

Read the current key/value entry from the dict. Entries are dynamically allocated when needed and must be freed after use with the wpa_dbus_dict_entry_clear function.

The returned entry object will be filled with the type and value of the next entry in the dict, or the type will be DBUS_TYPE_INVALID if an error occurred.

**Parameters:**
***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_read

***entry*** A valid dict entry object into which the dict key and value will be placed

**Returns:**
TRUE on success, FALSE on failure

Definition at line 873 of file dbus_dict_helpers.c.

Here is the call graph for this function:


#### 6.38.2.19 dbus_bool_t wpa_dbus_dict_has_dict_entry (DBusMessageIter * *iter_dict*)

Return whether or not there are additional dictionary entries.

**Parameters:**
***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_read

**Returns:**
TRUE if more dict entries exists, FALSE if no more dict entries exist

Definition at line 923 of file dbus_dict_helpers.c.

#### 6.38.2.20 dbus_bool_t wpa_dbus_dict_open_read (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

Start reading from a dbus dict.

**Parameters:**
***iter*** A valid DBusMessageIter pointing to the start of the dict

***iter_dict*** (out) A DBusMessageIter to be passed to wpa_dbus_dict_read_next_entry

**Returns:**
TRUE on success, FALSE on failure

Definition at line 618 of file dbus_dict_helpers.c.

#### 6.38.2.21 dbus_bool_t wpa_dbus_dict_open_write (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

Start a dict in a dbus message. Should be paired with a call to wpa_dbus_dict_close_write.

**Parameters:**
***iter*** A valid dbus message iterator

***iter_dict*** (out) A dict iterator to pass to further dict functions

**Returns:**
TRUE on success, FALSE on failure

Definition at line 34 of file dbus_dict_helpers.c.

#### 6.38.2.22 dbus_bool_t wpa_dbus_dict_string_array_add_element (DBusMessageIter * *iter_array*, const char * *elem*)

Add a single string element to a string array dict entry

**Parameters:**
***iter_array*** A valid DBusMessageIter returned from wpa_dbus_dict_begin_string_array's iter_array parameter

***elem*** The string element to be added to the dict entry's string array

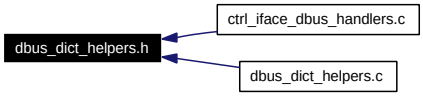
**Returns:**
TRUE on success, FALSE on failure

Definition at line 518 of file dbus_dict_helpers.c.

# 6.39 dbus_dict_helpers.h File Reference

WPA Supplicant / dbus-based control interface.

This graph shows which files directly or indirectly include this file:




## Functions

- dbus_bool_t wpa_dbus_dict_open_write (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_close_write (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_append_string (DBusMessageIter *iter_dict, const char *key, const char *value)
- dbus_bool_t wpa_dbus_dict_append_byte (DBusMessageIter *iter_dict, const char *key, const char value)
- dbus_bool_t wpa_dbus_dict_append_bool (DBusMessageIter *iter_dict, const char *key, const dbus_bool_t value)
- dbus_bool_t wpa_dbus_dict_append_int16 (DBusMessageIter *iter_dict, const char *key, const dbus_int16_t value)
- dbus_bool_t wpa_dbus_dict_append_uint16 (DBusMessageIter *iter_dict, const char *key, const dbus_uint16_t value)
- dbus_bool_t wpa_dbus_dict_append_int32 (DBusMessageIter *iter_dict, const char *key, const dbus_int32_t value)
- dbus_bool_t wpa_dbus_dict_append_uint32 (DBusMessageIter *iter_dict, const char *key, const dbus_uint32_t value)
- dbus_bool_t wpa_dbus_dict_append_int64 (DBusMessageIter *iter_dict, const char *key, const dbus_int64_t value)
- dbus_bool_t wpa_dbus_dict_append_uint64 (DBusMessageIter *iter_dict, const char *key, const dbus_uint64_t value)
- dbus_bool_t wpa_dbus_dict_append_double (DBusMessageIter *iter_dict, const char *key, const double value)
- dbus_bool_t wpa_dbus_dict_append_object_path (DBusMessageIter *iter_dict, const char *key, const char *value)
- dbus_bool_t wpa_dbus_dict_append_byte_array (DBusMessageIter *iter_dict, const char *key, const char *value, const dbus_uint32_t value_len)
- dbus_bool_t wpa_dbus_dict_begin_string_array (DBusMessageIter *iter_dict, const char *key, DBusMessageIter *iter_dict_entry, DBusMessageIter *iter_dict_val, DBusMessageIter *iter_array)
- dbus_bool_t wpa_dbus_dict_string_array_add_element (DBusMessageIter *iter_array, const char *elem)
- dbus_bool_t wpa_dbus_dict_end_string_array (DBusMessageIter *iter_dict, DBusMessageIter *iter_dict_entry, DBusMessageIter *iter_dict_val, DBusMessageIter *iter_array)
- dbus_bool_t wpa_dbus_dict_append_string_array (DBusMessageIter *iter_dict, const char *key, const char **items, const dbus_uint32_t num_items)
- dbus_bool_t wpa_dbus_dict_open_read (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_get_entry (DBusMessageIter *iter_dict, struct wpa_dbus_dict_entry *entry)
- dbus_bool_t wpa_dbus_dict_has_dict_entry (DBusMessageIter *iter_dict)
- void wpa_dbus_dict_entry_clear (struct wpa_dbus_dict_entry *entry)

### 6.39.1 Detailed Description

WPA Supplicant / dbus-based control interface.

**Copyright**
Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file dbus_dict_helpers.h.

### 6.39.2 Function Documentation

#### 6.39.2.1 dbus_bool_t wpa_dbus_dict_append_bool (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_bool_t *value*)

Add a boolean entry to the dict.

**Parameters:**
***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The boolean value

**Returns:**
TRUE on success, FALSE on failure

Definition at line 261 of file dbus_dict_helpers.c.

#### 6.39.2.2 dbus_bool_t wpa_dbus_dict_append_byte (DBusMessageIter * *iter_dict*, const char * *key*, const char *value*)

Add a byte entry to the dict.

**Parameters:**
***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The byte value

**Returns:**
TRUE on success, FALSE on failure

Definition at line 241 of file dbus_dict_helpers.c.

#### 6.39.2.3 dbus_bool_t wpa_dbus_dict_append_byte_array (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*, const dbus_uint32_t *value_len*)

Add a byte array entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The byte array

***value_len*** The length of the byte array, in bytes

**Returns:**

TRUE on success, FALSE on failure

Definition at line 450 of file dbus_dict_helpers.c.

#### 6.39.2.4 dbus_bool_t wpa_dbus_dict_append_double (DBusMessageIter * *iter_dict*, const char * *key*, const double *value*)

Add a double-precision floating point entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The double-precision floating point value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 407 of file dbus_dict_helpers.c.

#### 6.39.2.5 dbus_bool_t wpa_dbus_dict_append_int16 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int16_t *value*)

Add a 16-bit signed integer entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 16-bit signed integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 281 of file dbus_dict_helpers.c.

#### 6.39.2.6 dbus_bool_t wpa_dbus_dict_append_int32 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int32_t *value*)

Add a 32-bit signed integer to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 32-bit signed integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 323 of file dbus_dict_helpers.c.

#### 6.39.2.7 dbus_bool_t wpa_dbus_dict_append_int64 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int64_t *value*)

Add a 64-bit integer entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 64-bit integer value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 365 of file dbus_dict_helpers.c.

#### 6.39.2.8 dbus_bool_t wpa_dbus_dict_append_object_path (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*)

Add a DBus object path entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The DBus object path value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 428 of file dbus_dict_helpers.c.

#### 6.39.2.9 dbus_bool_t wpa_dbus_dict_append_string (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*)

Add a string entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The string value

**Returns:**

TRUE on success, FALSE on failure

Definition at line 221 of file dbus_dict_helpers.c.

#### 6.39.2.10 dbus_bool_t wpa_dbus_dict_append_string_array (DBusMessageIter * *iter_dict*, const char * *key*, const char ** *items*, const dbus_uint32_t *num_items*)

Convenience function to add an entire string array to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from nmu_dbus_dict_open_write

***key*** The key of the dict item

***items*** The array of strings

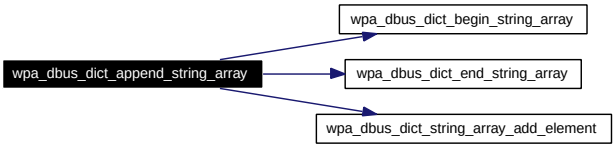
***num_items*** The number of strings in the array

**Returns:**

TRUE on success, FALSE on failure

Definition at line 573 of file dbus_dict_helpers.c.

Here is the call graph for this function:




#### 6.39.2.11 dbus_bool_t wpa_dbus_dict_append_uint16 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint16_t *value*)

Add a 16-bit unsigned integer entry to the dict.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 16-bit unsigned integer value

**Returns:**
TRUE on success, FALSE on failure

Definition at line 302 of file dbus_dict_helpers.c.

#### 6.39.2.12 dbus_bool_t wpa_dbus_dict_append_uint32 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint32_t *value*)

Add a 32-bit unsigned integer entry to the dict.

**Parameters:**
***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 32-bit unsigned integer value

**Returns:**
TRUE on success, FALSE on failure

Definition at line 344 of file dbus_dict_helpers.c.

#### 6.39.2.13 dbus_bool_t wpa_dbus_dict_append_uint64 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint64_t *value*)

Add a 64-bit unsigned integer entry to the dict.

**Parameters:**
***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_write

***key*** The key of the dict item

***value*** The 64-bit unsigned integer value

**Returns:**
TRUE on success, FALSE on failure

Definition at line 386 of file dbus_dict_helpers.c.

#### 6.39.2.14 dbus_bool_t wpa_dbus_dict_begin_string_array (DBusMessageIter * *iter_dict*, const char * *key*, DBusMessageIter * *iter_dict_entry*, DBusMessageIter * *iter_dict_val*, DBusMessageIter * *iter_array*)

Begin a string array entry in the dict

**Parameters:**
***iter_dict*** A valid DBusMessageIter returned from nmu_dbus_dict_open_write

***key*** The key of the dict item

***iter_dict_entry*** A private DBusMessageIter provided by the caller to be passed to wpa_dbus_dict_-end_string_array

***iter_dict_val*** A private DBusMessageIter provided by the caller to be passed to wpa_dbus_dict_end_-string_array

***iter_array*** On return, the DBusMessageIter to be passed to wpa_dbus_dict_string_array_add_element

**Returns:**
TRUE on success, FALSE on failure

Definition at line 479 of file dbus_dict_helpers.c.

#### 6.39.2.15 dbus_bool_t wpa_dbus_dict_close_write (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

End a dict element in a dbus message. Should be paired with a call to wpa_dbus_dict_open_write.

**Parameters:**
***iter*** valid dbus message iterator, same as passed to wpa_dbus_dict_open_write()

***iter_dict*** a dbus dict iterator returned from wpa_dbus_dict_open_write

**Returns:**
TRUE on success, FALSE on failure

Definition at line 65 of file dbus_dict_helpers.c.

#### 6.39.2.16 dbus_bool_t wpa_dbus_dict_end_string_array (DBusMessageIter * *iter_dict*, DBusMessageIter * *iter_dict_entry*, DBusMessageIter * *iter_dict_val*, DBusMessageIter * *iter_array*)

End a string array dict entry

**Parameters:**
***iter_dict*** A valid DBusMessageIter returned from nmu_dbus_dict_open_write

***iter_dict_entry*** A private DBusMessageIter returned from wpa_dbus_dict_end_string_array

***iter_dict_val*** A private DBusMessageIter returned from wpa_dbus_dict_end_string_array

***iter_array*** A DBusMessageIter returned from wpa_dbus_dict_end_string_array

**Returns:**
TRUE on success, FALSE on failure

Definition at line 543 of file dbus_dict_helpers.c.

#### 6.39.2.17 void wpa_dbus_dict_entry_clear (struct wpa_dbus_dict_entry * *entry*)

Free any memory used by the entry object.

**Parameters:**
***entry*** The entry object

Definition at line 939 of file dbus_dict_helpers.c.

#### 6.39.2.18 dbus_bool_t wpa_dbus_dict_get_entry (DBusMessageIter * *iter_dict*, struct wpa_dbus_dict_entry * *entry*)

Read the current key/value entry from the dict. Entries are dynamically allocated when needed and must be freed after use with the wpa_dbus_dict_entry_clear function.

The returned entry object will be filled with the type and value of the next entry in the dict, or the type will be DBUS_TYPE_INVALID if an error occurred.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_read

***entry*** A valid dict entry object into which the dict key and value will be placed

**Returns:**

TRUE on success, FALSE on failure

Definition at line 873 of file dbus_dict_helpers.c.

Here is the call graph for this function:


#### 6.39.2.19 dbus_bool_t wpa_dbus_dict_has_dict_entry (DBusMessageIter * *iter_dict*)

Return whether or not there are additional dictionary entries.

**Parameters:**

***iter_dict*** A valid DBusMessageIter returned from wpa_dbus_dict_open_read

**Returns:**

TRUE if more dict entries exists, FALSE if no more dict entries exist

Definition at line 923 of file dbus_dict_helpers.c.

#### 6.39.2.20 dbus_bool_t wpa_dbus_dict_open_read (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

Start reading from a dbus dict.

**Parameters:**

***iter*** A valid DBusMessageIter pointing to the start of the dict

***iter_dict*** (out) A DBusMessageIter to be passed to wpa_dbus_dict_read_next_entry

**Returns:**

TRUE on success, FALSE on failure

Definition at line 618 of file dbus_dict_helpers.c.

#### 6.39.2.21 dbus_bool_t wpa_dbus_dict_open_write (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

Start a dict in a dbus message. Should be paired with a call to wpa_dbus_dict_close_write.

**Parameters:**

***iter*** A valid dbus message iterator

***iter_dict*** (out) A dict iterator to pass to further dict functions

**Returns:**

TRUE on success, FALSE on failure

Definition at line 34 of file dbus_dict_helpers.c.

#### 6.39.2.22 dbus_bool_t wpa_dbus_dict_string_array_add_element (DBusMessageIter * *iter_array*, const char * *elem*)

Add a single string element to a string array dict entry

**Parameters:**

***iter_array*** A valid DBusMessageIter returned from wpa_dbus_dict_begin_string_array's iter_array parameter

***elem*** The string element to be added to the dict entry's string array

**Returns:**

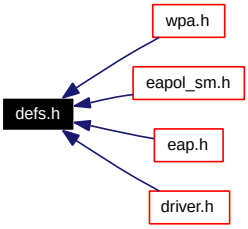
TRUE on success, FALSE on failure

Definition at line 518 of file dbus_dict_helpers.c.

## 6.40 defs.h File Reference

WPA Supplicant - Common definitions.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **MLME_SETPROTECTION_PROTECT_TYPE_NONE** 0
- #define **MLME_SETPROTECTION_PROTECT_TYPE_RX** 1
- #define **MLME_SETPROTECTION_PROTECT_TYPE_TX** 2
- #define **MLME_SETPROTECTION_PROTECT_TYPE_RX_TX** 3
- #define **MLME_SETPROTECTION_KEY_TYPE_GROUP** 0
- #define **MLME_SETPROTECTION_KEY_TYPE_PAIRWISE** 1

### Enumerations

- enum **Boolean** { **FALSE** = 0, **TRUE** = 1 }
- enum **wpa_alg** {
  **WPA_ALG_NONE**, **WPA_ALG_WEP**, **WPA_ALG_TKIP**, **WPA_ALG_CCMP**,
  **WPA_ALG_IGTK**, **WPA_ALG_DHV** }
- enum **wpa_cipher** {
  **CIPHER_NONE**, **CIPHER_WEP40**, **CIPHER_TKIP**, **CIPHER_CCMP**,
  **CIPHER_WEP104** }
- enum **wpa_key_mgmt** {
  **KEY_MGMT_802_1X**, **KEY_MGMT_PSK**, **KEY_MGMT_NONE**, **KEY_MGMT_802_1X_NO_WPA**,
  **KEY_MGMT_WPA_NONE** }
- enum wpa_states {
  WPA_DISCONNECTED, WPA_INACTIVE, WPA_SCANNING, WPA_ASSOCIATING,
  WPA_ASSOCIATED, WPA_4WAY_HANDSHAKE, WPA_GROUP_HANDSHAKE, WPA_COMPLETED }

### 6.40.1 Detailed Description

WPA Supplicant - Common definitions.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file defs.h.

## 6.40.2 Enumeration Type Documentation

### 6.40.2.1 enum wpa_states

enum wpa_states - wpa_supplicant state

These enumeration values are used to indicate the current wpa_supplicant state (wpa_s->wpa_state). The current state can be retrieved with wpa_supplicant_get_state() function and the state can be changed by calling wpa_supplicant_set_state(). In WPA state machine (wpa.c and preauth.c), the wrapper functions wpa_sm_get_state() and wpa_sm_set_state() should be used to access the state variable.

**Enumeration values:**

***WPA_DISCONNECTED*** Disconnected state.

This state indicates that client is not associated, but is likely to start looking for an access point. This state is entered when a connection is lost.

***WPA_INACTIVE*** Inactive state (wpa_supplicant disabled).

This state is entered if there are no enabled networks in the configuration. wpa_supplicant is not trying to associate with a new network and external interaction (e.g., ctrl_iface call to add or enable a network) is needed to start association.

***WPA_SCANNING*** Scanning for a network.

This state is entered when wpa_supplicant starts scanning for a network.

***WPA_ASSOCIATING*** Trying to associate with a BSS/SSID.

This state is entered when wpa_supplicant has found a suitable BSS to associate with and the driver is configured to try to associate with this BSS in ap_scan=1 mode. When using ap_scan=2 mode, this state is entered when the driver is configured to try to associate with a network using the configured SSID and security policy.

***WPA_ASSOCIATED*** Association completed.

This state is entered when the driver reports that association has been successfully completed with an AP. If IEEE 802.1X is used (with or without WPA/WPA2), wpa_supplicant remains in this state until the IEEE 802.1X/EAPOL authentication has been completed.

***WPA_4WAY_HANDSHAKE*** WPA 4-Way Key Handshake in progress.

This state is entered when WPA/WPA2 4-Way Handshake is started. In case of WPA-PSK, this happens when receiving the first EAPOL-Key frame after association. In case of WPA-EAP, this state is entered when the IEEE 802.1X/EAPOL authentication has been completed.

***WPA_GROUP_HANDSHAKE*** WPA Group Key Handshake in progress.

This state is entered when 4-Way Key Handshake has been completed (i.e., when the supplicant sends out message 4/4) and when Group Key rekeying is started by the AP (i.e., when supplicant receives message 1/2).

***WPA_COMPLETED*** All authentication completed.

This state is entered when the full authentication process is completed. In case of WPA2, this happens when the 4-Way Handshake is successfully completed. With WPA, this state is entered after the Group Key Handshake; with IEEE 802.1X (non-WPA) connection is completed after dynamic keys are received (or if not used, after the EAP authentication has been completed).

With static WEP keys and plaintext connections, this state is entered when an association has been completed.

This state indicates that the supplicant has completed its processing for the association phase and that data connection is fully configured.

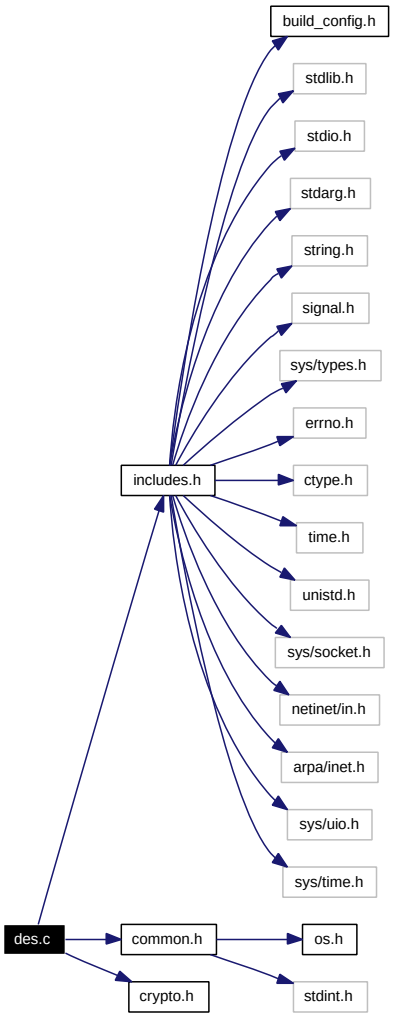
Definition at line 45 of file defs.h.

## 6.41 des.c File Reference

DES and 3DES-EDE ciphers.

```
#include "includes.h"
#include "common.h"
#include "crypto.h"
```

Include dependency graph for des.c:




### 6.41.1 Detailed Description

DES and 3DES-EDE ciphers.

Modifications to LibTomCrypt implementation:

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General

Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file des.c.

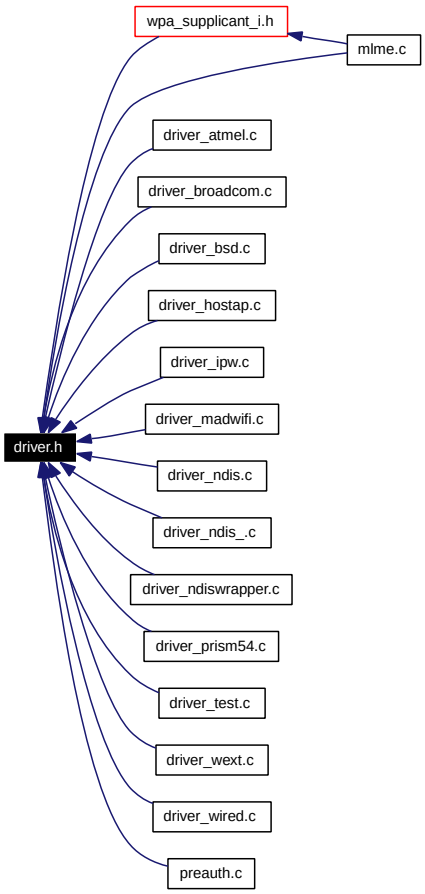
## 6.42 driver.h File Reference

WPA Supplicant - driver interface definition.

```
#include "defs.h"
```

Include dependency graph for driver.h:


This graph shows which files directly or indirectly include this file:




### Defines

- #define **WPA_SUPPLICANT_DRIVER_VERSION** 2
- #define **AUTH_ALG_OPEN_SYSTEM** 0x01
- #define **AUTH_ALG_SHARED_KEY** 0x02
- #define **AUTH_ALG_LEAP** 0x04
- #define **IEEE80211_MODE_INFRA** 0
- #define **IEEE80211_MODE_IBSS** 1
- #define **IEEE80211_CAP_ESS** 0x0001

- #define **IEEE80211_CAP_IBSS** 0x0002
- #define **IEEE80211_CAP_PRIVACY** 0x0010
- #define **SSID_MAX_WPA_IE_LEN** 40
- #define **WPA_DRIVER_CAPA_KEY_MGMT_WPA** 0x00000001
- #define **WPA_DRIVER_CAPA_KEY_MGMT_WPA2** 0x00000002
- #define **WPA_DRIVER_CAPA_KEY_MGMT_WPA_PSK** 0x00000004
- #define **WPA_DRIVER_CAPA_KEY_MGMT_WPA2_PSK** 0x00000008
- #define **WPA_DRIVER_CAPA_KEY_MGMT_WPA_NONE** 0x00000010
- #define **WPA_DRIVER_CAPA_ENC_WEP40** 0x00000001
- #define **WPA_DRIVER_CAPA_ENC_WEP104** 0x00000002
- #define **WPA_DRIVER_CAPA_ENC_TKIP** 0x00000004
- #define **WPA_DRIVER_CAPA_ENC_CCMP** 0x00000008
- #define **WPA_DRIVER_AUTH_OPEN** 0x00000001
- #define **WPA_DRIVER_AUTH_SHARED** 0x00000002
- #define **WPA_DRIVER_AUTH_LEAP** 0x00000004
- #define **WPA_DRIVER_FLAGS_DRIVER_IE** 0x00000001
- #define **WPA_DRIVER_FLAGS_SET_KEYS_AFTER_ASSOC** 0x00000002
- #define **WPA_DRIVER_FLAGS_USER_SPACE_MLME** 0x00000004
- #define **WPA_CHAN_W_SCAN** 0x00000001
- #define **WPA_CHAN_W_ACTIVE_SCAN** 0x00000002
- #define **WPA_CHAN_W_IBSS** 0x00000004
- #define **WPA_RATE_ERP** 0x00000001
- #define **WPA_RATE_BASIC** 0x00000002
- #define **WPA_RATE_PREAMBLE2** 0x00000004
- #define **WPA_RATE_SUPPORTED** 0x00000010
- #define **WPA_RATE_OFDM** 0x00000020
- #define **WPA_RATE_CCK** 0x00000040
- #define **WPA_RATE_MANDATORY** 0x00000100

## Enumerations

- enum **wpa_hw_mode** { **WPA_MODE_IEEE80211B**, **WPA_MODE_IEEE80211G**, **WPA_MODE_IEEE80211A**, **NUM_WPA_MODES** }

### 6.42.1 Detailed Description

WPA Supplicant - driver interface definition.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

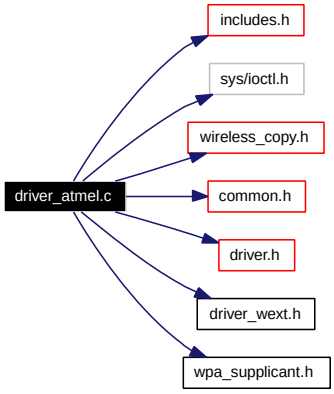
Definition in file driver.h.

## 6.43 driver_atmel.c File Reference

WPA Supplicant - Driver interaction with Atmel Wireless LAN drivers.

```
#include "includes.h"
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
#include "wpa_supplicant.h"
```

Include dependency graph for driver_atmel.c:




### Defines

- #define **ATMEL_WPA_IOCTL** (SIOCIWFIRSTPRIV + 2)
- #define **ATMEL_WPA_IOCTL_PARAM** (SIOCIWFIRSTPRIV + 3)
- #define **ATMEL_WPA_IOCTL_GET_PARAM** (SIOCIWFIRSTPRIV + 4)
- #define **MAX_KEY_LENGTH** 40

### Enumerations

- enum { **SET_WPA_ENCRYPTION** = 1, **SET_CIPHER_SUITES** = 2, **MLME_STA_DEAUTH** = 3, **MLME_STA_DISASSOC** = 4 }
- enum { **ATMEL_PARAM_WPA** = 1, **ATMEL_PARAM_PRIVACY_INVOKED** = 2, **ATMEL_-PARAM_WPA_TYPE** = 3 }

### Variables

- const struct wpa_driver_ops **wpa_driver_atmel_ops**

### 6.43.1 Detailed Description

WPA Supplicant - Driver interaction with Atmel Wireless LAN drivers.

**Copyright**

Copyright (c) 2000-2005, ATMEL Corporation Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_atmel.c.

### 6.43.2 Variable Documentation

#### 6.43.2.1 const struct wpa_driver_ops wpa_driver_atmel_ops

**Initial value:**

```
{
        .name = "atmel",
        .desc = "ATMEL AT76C5XXx (USB, PCMCIA)",
        .get_bssid = wpa_driver_atmel_get_bssid,
        .get_ssid = wpa_driver_atmel_get_ssid,
        .set_wpa = wpa_driver_atmel_set_wpa,
        .set_key = wpa_driver_atmel_set_key,
        .init = wpa_driver_atmel_init,
        .deinit = wpa_driver_atmel_deinit,
        .set_countermeasures = wpa_driver_atmel_set_countermeasures,
        .set_drop_unencrypted = wpa_driver_atmel_set_drop_unencrypted,
        .scan = wpa_driver_atmel_scan,
        .get_scan_results = wpa_driver_atmel_get_scan_results,
        .deauthenticate = wpa_driver_atmel_deauthenticate,
        .disassociate = wpa_driver_atmel_disassociate,
        .associate = wpa_driver_atmel_associate,
        .set_operstate = wpa_driver_atmel_set_operstate,
}
```

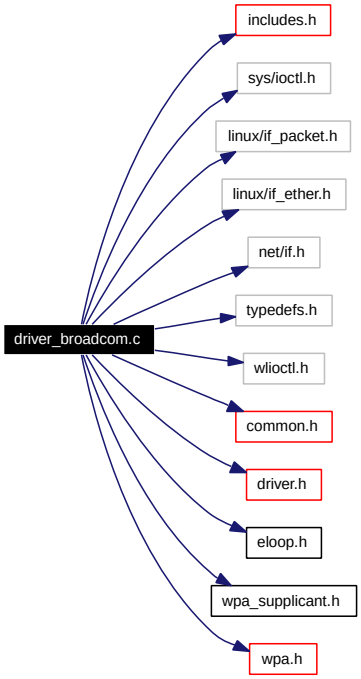
Definition at line 492 of file driver_atmel.c.

## 6.44 driver_broadcom.c File Reference

WPA Supplicant - driver interaction with Broadcom wl.o driver.

```
#include "includes.h"
#include <sys/ioctl.h>
#include <linux/if_packet.h>
#include <linux/if_ether.h>
#include <net/if.h>
#include <typedefs.h>
#include <wlioctl.h>
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "wpa.h"
```

Include dependency graph for driver_broadcom.c:




### Defines

- #define **WLC_DEAUTHENTICATE** 143
- #define **WLC_DEAUTHENTICATE_WITH_REASON** 201
- #define **WLC_SET_TKIP_COUNTERMEASURES** 202
- #define **WL_VERSION** 360130

- #define **WPA_ENABLED** 1
- #define **PSK_ENABLED** 2
- #define **WAUTH_WPA_ENABLED**(wauth) ((wauth) & WPA_ENABLED)
- #define **WAUTH_PSK_ENABLED**(wauth) ((wauth) & PSK_ENABLED)
- #define **WAUTH_ENABLED**(wauth) ((wauth) & (WPA_ENABLED | PSK_ENABLED))
- #define **WSEC_PRIMARY_KEY** WL_PRIMARY_KEY

## Typedefs

- typedef wl_wsec_key_t **wsec_key_t**

## Variables

- bss_ie_hdr **packed**
- const struct wpa_driver_ops **wpa_driver_broadcom_ops**

### 6.44.1 Detailed Description

WPA Supplicant - driver interaction with Broadcom wl.o driver.

**Copyright**

Copyright (c) 2004, Nikki Chumkov <nikki@gattaca.ru> Copyright (c) 2004, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_broadcom.c.

### 6.44.2 Variable Documentation

#### 6.44.2.1 const struct wpa_driver_ops wpa_driver_broadcom_ops

**Initial value:**

```
{
        .name = "broadcom",
        .desc = "Broadcom wl.o driver",
        .get_bssid = wpa_driver_broadcom_get_bssid,
        .get_ssid = wpa_driver_broadcom_get_ssid,
        .set_wpa = wpa_driver_broadcom_set_wpa,
        .set_key = wpa_driver_broadcom_set_key,
        .init = wpa_driver_broadcom_init,
        .deinit = wpa_driver_broadcom_deinit,
        .set_countermeasures = wpa_driver_broadcom_set_countermeasures,
        .set_drop_unencrypted = wpa_driver_broadcom_set_drop_unencrypted,
        .scan = wpa_driver_broadcom_scan,
        .get_scan_results = wpa_driver_broadcom_get_scan_results,
        .deauthenticate = wpa_driver_broadcom_deauthenticate,
        .disassociate = wpa_driver_broadcom_disassociate,
        .associate = wpa_driver_broadcom_associate,
}
```

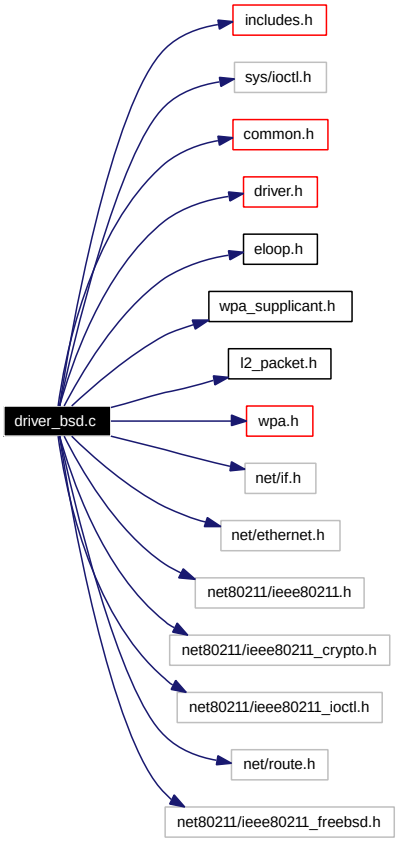
Definition at line 582 of file driver_broadcom.c.

## 6.45 driver_bsd.c File Reference

WPA Supplicant - driver interaction with BSD net80211 layer.

```
#include "includes.h"
#include <sys/ioctl.h>
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "l2_packet.h"
#include "wpa.h"
#include <net/if.h>
#include <net/ethernet.h>
#include <net80211/ieee80211.h>
#include <net80211/ieee80211_crypto.h>
#include <net80211/ieee80211_ioctl.h>
#include <net/route.h>
#include <net80211/ieee80211_freebsd.h>
```

Include dependency graph for driver_bsd.c:




## Defines

- #define **LE_READ_4**(p)
- #define **min**(a, b) ((a)>(b)?(b):(a))
- #define **GETPARAM**(drv, param, v) (((v) = get80211param(drv, param)) != -1)

## Variables

- const struct wpa_driver_ops **wpa_driver_bsd_ops**

### 6.45.1 Detailed Description

WPA Supplicant - driver interaction with BSD net80211 layer.

**Copyright**

Copyright (c) 2004, Sam Leffler <sam@errno.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_bsd.c.

## 6.45.2 Define Documentation

### 6.45.2.1 #define LE_READ_4(p)

**Value:**

```
((u_int32_t)                                    \
         ((((const u_int8_t *)(p))[0]      ) |          \
          (((const u_int8_t *)(p))[1] <<  8) |          \
          (((const u_int8_t *)(p))[2] << 16) |          \
          (((const u_int8_t *)(p))[3] << 24)))
```

Definition at line 599 of file driver_bsd.c.

## 6.45.3 Variable Documentation

### 6.45.3.1 const struct wpa_driver_ops wpa_driver_bsd_ops

**Initial value:**

```
 {
        .name                   = "bsd",
        .desc                   = "BSD 802.11 support (Atheros, etc.)",
        .init                   = wpa_driver_bsd_init,
        .deinit                 = wpa_driver_bsd_deinit,
        .get_bssid              = wpa_driver_bsd_get_bssid,
        .get_ssid               = wpa_driver_bsd_get_ssid,
        .set_wpa                = wpa_driver_bsd_set_wpa,
        .set_key                = wpa_driver_bsd_set_key,
        .set_countermeasures    = wpa_driver_bsd_set_countermeasures,
        .set_drop_unencrypted   = wpa_driver_bsd_set_drop_unencrypted,
        .scan                   = wpa_driver_bsd_scan,
        .get_scan_results       = wpa_driver_bsd_get_scan_results,
        .deauthenticate         = wpa_driver_bsd_deauthenticate,
        .disassociate           = wpa_driver_bsd_disassociate,
        .associate              = wpa_driver_bsd_associate,
        .set_auth_alg           = wpa_driver_bsd_set_auth_alg,
}
```

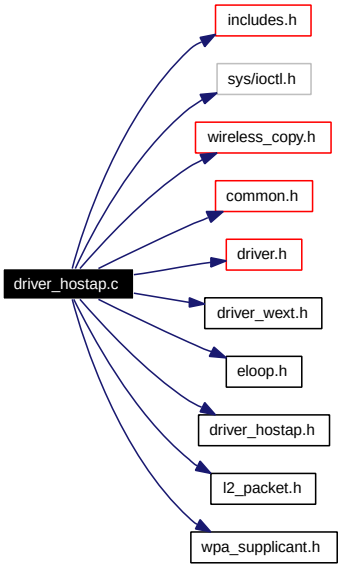
Definition at line 775 of file driver_bsd.c.

## 6.46 driver_hostap.c File Reference

WPA Supplicant - driver interaction with Linux Host AP driver.

```
#include "includes.h"
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
#include "eloop.h"
#include "driver_hostap.h"
#include "l2_packet.h"
#include "wpa_supplicant.h"
```

Include dependency graph for driver_hostap.c:




### Variables

- const struct wpa_driver_ops **wpa_driver_hostap_ops**

### 6.46.1 Detailed Description

WPA Supplicant - driver interaction with Linux Host AP driver.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_hostap.c.

## 6.46.2 Variable Documentation

### 6.46.2.1 const struct wpa_driver_ops wpa_driver_hostap_ops

**Initial value:**

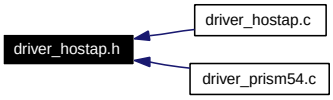
```
{
        .name = "hostap",
        .desc = "Host AP driver (Intersil Prism2/2.5/3)",
        .get_bssid = wpa_driver_hostap_get_bssid,
        .get_ssid = wpa_driver_hostap_get_ssid,
        .set_wpa = wpa_driver_hostap_set_wpa,
        .set_key = wpa_driver_hostap_set_key,
        .set_countermeasures = wpa_driver_hostap_set_countermeasures,
        .set_drop_unencrypted = wpa_driver_hostap_set_drop_unencrypted,
        .scan = wpa_driver_hostap_scan,
        .get_scan_results = wpa_driver_hostap_get_scan_results,
        .deauthenticate = wpa_driver_hostap_deauthenticate,
        .disassociate = wpa_driver_hostap_disassociate,
        .associate = wpa_driver_hostap_associate,
        .set_auth_alg = wpa_driver_hostap_set_auth_alg,
        .init = wpa_driver_hostap_init,
        .deinit = wpa_driver_hostap_deinit,
        .set_operstate = wpa_driver_hostap_set_operstate,
}
```

Definition at line 498 of file driver_hostap.c.

# 6.47 driver_hostap.h File Reference

WPA Supplicant - driver interaction with Linux Host AP driver.

This graph shows which files directly or indirectly include this file:




## Defines

- #define **PRISM2_IOCTL_PRISM2_PARAM** (SIOCIWFIRSTPRIV + 0)
- #define **PRISM2_IOCTL_RESET** (SIOCIWFIRSTPRIV + 6)
- #define **PRISM2_IOCTL_HOSTAPD** (SIOCDEVPRIVATE + 14)
- #define **PRISM2_HOSTAPD_MAX_BUF_SIZE** 1024
- #define **PRISM2_HOSTAPD_RID_HDR_LEN** ((int) (&((struct prism2_hostapd_param ∗) 0) → u.rid.data))
- #define **PRISM2_HOSTAPD_GENERIC_ELEMENT_HDR_LEN** ((int) (&((struct prism2_hostapd_param ∗) 0) → u.generic_elem.data))
- #define **HOSTAP_CRYPT_ALG_NAME_LEN** 16
- #define **MLME_STA_DEAUTH** 0
- #define **MLME_STA_DISASSOC** 1
- #define **HOSTAP_CRYPT_FLAG_SET_TX_KEY** 0x01
- #define **HOSTAP_CRYPT_FLAG_PERMANENT** 0x02
- #define **HOSTAP_CRYPT_ERR_UNKNOWN_ALG** 2
- #define **HOSTAP_CRYPT_ERR_UNKNOWN_ADDR** 3
- #define **HOSTAP_CRYPT_ERR_CRYPT_INIT_FAILED** 4
- #define **HOSTAP_CRYPT_ERR_KEY_SET_FAILED** 5
- #define **HOSTAP_CRYPT_ERR_TX_KEY_SET_FAILED** 6
- #define **HOSTAP_CRYPT_ERR_CARD_CONF_FAILED** 7

## Enumerations

- enum {

  **PRISM2_PARAM_TXRATECTRL** = 2, **PRISM2_PARAM_BEACON_INT** = 3, **PRISM2_PARAM_PSEUDO_IBSS** = 4, **PRISM2_PARAM_ALC** = 5,

  **PRISM2_PARAM_DUMP** = 7, **PRISM2_PARAM_OTHER_AP_POLICY** = 8, **PRISM2_PARAM_AP_MAX_INACTIVITY** = 9, **PRISM2_PARAM_AP_BRIDGE_PACKETS** = 10,

  **PRISM2_PARAM_DTIM_PERIOD** = 11, **PRISM2_PARAM_AP_NULLFUNC_ACK** = 12, **PRISM2_PARAM_MAX_WDS** = 13, **PRISM2_PARAM_AP_AUTOM_AP_WDS** = 14,

  **PRISM2_PARAM_AP_AUTH_ALGS** = 15, **PRISM2_PARAM_MONITOR_ALLOW_FCSERR** = 16, **PRISM2_PARAM_HOST_ENCRYPT** = 17, **PRISM2_PARAM_HOST_DECRYPT** = 18,

  **PRISM2_PARAM_BUS_MASTER_THRESHOLD_RX** = 19, **PRISM2_PARAM_BUS_MASTER_THRESHOLD_TX** = 20, **PRISM2_PARAM_HOST_ROAMING** = 21, **PRISM2_PARAM_BCRX_STA_KEY** = 22,

**PRISM2_PARAM_IEEE_802_1X** = 23, **PRISM2_PARAM_ANTSEL_TX** = 24, **PRISM2_PARAM_ANTSEL_RX** = 25, **PRISM2_PARAM_MONITOR_TYPE** = 26,

**PRISM2_PARAM_WDS_TYPE** = 27, **PRISM2_PARAM_HOSTSCAN** = 28, **PRISM2_PARAM_AP_SCAN** = 29, **PRISM2_PARAM_ENH_SEC** = 30,

**PRISM2_PARAM_IO_DEBUG** = 31, **PRISM2_PARAM_BASIC_RATES** = 32, **PRISM2_PARAM_OPER_RATES** = 33, **PRISM2_PARAM_HOSTAPD** = 34,

**PRISM2_PARAM_HOSTAPD_STA** = 35, **PRISM2_PARAM_WPA** = 36, **PRISM2_PARAM_PRIVACY_INVOKED** = 37, **PRISM2_PARAM_TKIP_COUNTERMEASURES** = 38,

**PRISM2_PARAM_DROP_UNENCRYPTED** = 39, **PRISM2_PARAM_SCAN_CHANNEL_MASK** = 40 }

- enum {

**PRISM2_HOSTAPD_FLUSH** = 1, **PRISM2_HOSTAPD_ADD_STA** = 2, **PRISM2_HOSTAPD_REMOVE_STA** = 3, **PRISM2_HOSTAPD_GET_INFO_STA** = 4,

**PRISM2_SET_ENCRYPTION** = 6, **PRISM2_GET_ENCRYPTION** = 7, **PRISM2_HOSTAPD_SET_FLAGS_STA** = 8, **PRISM2_HOSTAPD_GET_RID** = 9,

**PRISM2_HOSTAPD_SET_RID** = 10, **PRISM2_HOSTAPD_SET_ASSOC_AP_ADDR** = 11, **PRISM2_HOSTAPD_SET_GENERIC_ELEMENT** = 12, **PRISM2_HOSTAPD_MLME** = 13,

**PRISM2_HOSTAPD_SCAN_REQ** = 14, **PRISM2_HOSTAPD_STA_CLEAR_STATS** = 15 }

### 6.47.1 Detailed Description

WPA Supplicant - driver interaction with Linux Host AP driver.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

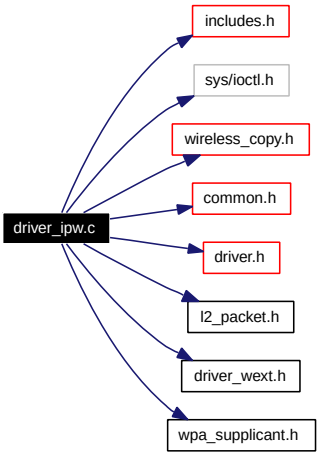
Definition in file driver_hostap.h.

## 6.48 driver_ipw.c File Reference

WPA Supplicant - driver interaction with Linux ipw2100/2200 drivers.

```
#include "includes.h"
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "l2_packet.h"
#include "driver_wext.h"
#include "wpa_supplicant.h"
```

Include dependency graph for driver_ipw.c:




### Defines

- #define **IPW_IOCTL_WPA_SUPPLICANT** SIOCIWFIRSTPRIV+30
- #define **IPW_CMD_SET_WPA_PARAM** 1
- #define **IPW_CMD_SET_WPA_IE** 2
- #define **IPW_CMD_SET_ENCRYPTION** 3
- #define **IPW_CMD_MLME** 4
- #define **IPW_PARAM_WPA_ENABLED** 1
- #define **IPW_PARAM_TKIP_COUNTERMEASURES** 2
- #define **IPW_PARAM_DROP_UNENCRYPTED** 3
- #define **IPW_PARAM_PRIVACY_INVOKED** 4
- #define **IPW_PARAM_AUTH_ALGS** 5
- #define **IPW_PARAM_IEEE_802_1X** 6
- #define **IPW_MLME_STA_DEAUTH** 1
- #define **IPW_MLME_STA_DISASSOC** 2
- #define **IPW_CRYPT_ERR_UNKNOWN_ALG** 2
- #define **IPW_CRYPT_ERR_UNKNOWN_ADDR** 3
- #define **IPW_CRYPT_ERR_CRYPT_INIT_FAILED** 4

- #define **IPW_CRYPT_ERR_KEY_SET_FAILED** 5
- #define **IPW_CRYPT_ERR_TX_KEY_SET_FAILED** 6
- #define **IPW_CRYPT_ERR_CARD_CONF_FAILED** 7
- #define **IPW_CRYPT_ALG_NAME_LEN** 16

## Variables

- const struct wpa_driver_ops **wpa_driver_ipw_ops**

### 6.48.1 Detailed Description

WPA Supplicant - driver interaction with Linux ipw2100/2200 drivers.

**Copyright**

Copyright (c) 2005 Zhu Yi <yi.zhu@intel.com> Copyright (c) 2004 Lubomir Gelo <lgelo@cnc.sk> Copyright (c) 2003-2004, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Please note that ipw2100/2200 drivers change to use generic Linux wireless extensions if the kernel includes support for WE-18 or newer (Linux 2.6.13 or newer). driver_wext.c should be used in those cases.

Definition in file driver_ipw.c.

### 6.48.2 Variable Documentation

#### 6.48.2.1 const struct wpa_driver_ops wpa_driver_ipw_ops

**Initial value:**

```
{
        .name = "ipw",
        .desc = "Intel ipw2100/2200 driver (old; use wext with Linux 2.6.13 "
        "or newer)",
        .get_bssid = wpa_driver_ipw_get_bssid,
        .get_ssid = wpa_driver_ipw_get_ssid,
        .set_wpa = wpa_driver_ipw_set_wpa,
        .set_key = wpa_driver_ipw_set_key,
        .set_countermeasures = wpa_driver_ipw_set_countermeasures,
        .set_drop_unencrypted = wpa_driver_ipw_set_drop_unencrypted,
        .scan = wpa_driver_ipw_scan,
        .get_scan_results = wpa_driver_ipw_get_scan_results,
        .deauthenticate = wpa_driver_ipw_deauthenticate,
        .disassociate = wpa_driver_ipw_disassociate,
        .associate = wpa_driver_ipw_associate,
        .set_auth_alg = wpa_driver_ipw_set_auth_alg,
        .init = wpa_driver_ipw_init,
        .deinit = wpa_driver_ipw_deinit,
        .set_operstate = wpa_driver_ipw_set_operstate,
}
```

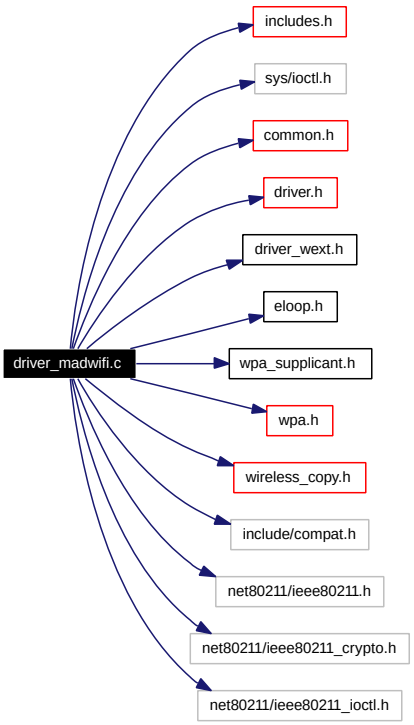
Definition at line 449 of file driver_ipw.c.

## 6.49 driver_madwifi.c File Reference

WPA Supplicant - driver interaction with MADWIFI 802.11 driver.

```
#include "includes.h"
#include <sys/ioctl.h>
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "wpa.h"
#include "wireless_copy.h"
#include <include/compat.h>
#include <net80211/ieee80211.h>
#include <net80211/ieee80211_crypto.h>
#include <net80211/ieee80211_ioctl.h>
```

Include dependency graph for driver_madwifi.c:




### Variables

- const struct wpa_driver_ops **wpa_driver_madwifi_ops**

### 6.49.1 Detailed Description

WPA Supplicant - driver interaction with MADWIFI 802.11 driver.

**Copyright**

Copyright (c) 2004, Sam Leffler <sam@errno.com> Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_madwifi.c.

### 6.49.2 Variable Documentation

#### 6.49.2.1 const struct wpa_driver_ops wpa_driver_madwifi_ops

**Initial value:**

```
{
        .name                   = "madwifi",
        .desc                   = "MADWIFI 802.11 support (Atheros, etc.)",
        .get_bssid              = wpa_driver_madwifi_get_bssid,
        .get_ssid               = wpa_driver_madwifi_get_ssid,
        .set_key                = wpa_driver_madwifi_set_key,
        .init                   = wpa_driver_madwifi_init,
        .deinit                 = wpa_driver_madwifi_deinit,
        .set_countermeasures    = wpa_driver_madwifi_set_countermeasures,
        .set_drop_unencrypted   = wpa_driver_madwifi_set_drop_unencrypted,
        .scan                   = wpa_driver_madwifi_scan,
        .get_scan_results       = wpa_driver_madwifi_get_scan_results,
        .deauthenticate         = wpa_driver_madwifi_deauthenticate,
        .disassociate           = wpa_driver_madwifi_disassociate,
        .associate              = wpa_driver_madwifi_associate,
        .set_auth_alg           = wpa_driver_madwifi_set_auth_alg,
        .set_operstate          = wpa_driver_madwifi_set_operstate,
}
```

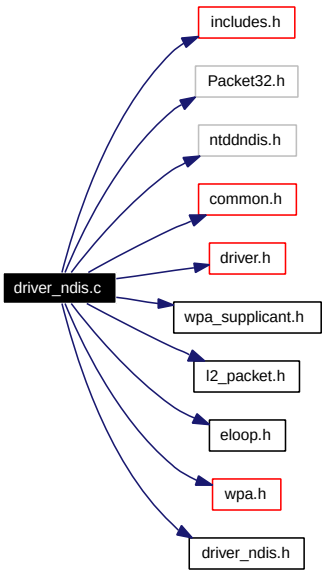
Definition at line 549 of file driver_madwifi.c.

## 6.50 driver_ndis.c File Reference

WPA Supplicant - Windows/NDIS driver interface.

```
#include "includes.h"
#include <Packet32.h>
#include <ntddndis.h>
#include "common.h"
#include "driver.h"
#include "wpa_supplicant.h"
#include "l2_packet.h"
#include "eloop.h"
#include "wpa.h"
#include "driver_ndis.h"
```

Include dependency graph for driver_ndis.c:




### Defines

- #define **OID_802_11_BSSID** 0x0d010101
- #define **OID_802_11_SSID** 0x0d010102
- #define **OID_802_11_INFRASTRUCTURE_MODE** 0x0d010108
- #define **OID_802_11_ADD_WEP** 0x0D010113
- #define **OID_802_11_REMOVE_WEP** 0x0D010114
- #define **OID_802_11_DISASSOCIATE** 0x0D010115
- #define **OID_802_11_BSSID_LIST** 0x0d010217
- #define **OID_802_11_AUTHENTICATION_MODE** 0x0d010118
- #define **OID_802_11_PRIVACY_FILTER** 0x0d010119
- #define **OID_802_11_BSSID_LIST_SCAN** 0x0d01011A

- #define **OID_802_11_WEP_STATUS** 0x0d01011B
- #define **OID_802_11_ENCRYPTION_STATUS** OID_802_11_WEP_STATUS
- #define **OID_802_11_ADD_KEY** 0x0d01011D
- #define **OID_802_11_REMOVE_KEY** 0x0d01011E
- #define **OID_802_11_ASSOCIATION_INFORMATION** 0x0d01011F
- #define **OID_802_11_TEST** 0x0d010120
- #define **OID_802_11_CAPABILITY** 0x0d010122
- #define **OID_802_11_PMKID** 0x0d010123
- #define **NDIS_802_11_LENGTH_SSID** 32
- #define **NDIS_802_11_LENGTH_RATES** 8
- #define **NDIS_802_11_LENGTH_RATES_EX** 16
- #define **NDIS_802_11_PMKID_CANDIDATE_PREAUTH_ENABLED** 0x01
- #define **NDIS_802_11_AUTH_REQUEST_REAUTH** 0x01
- #define **NDIS_802_11_AUTH_REQUEST_KEYUPDATE** 0x02
- #define **NDIS_802_11_AUTH_REQUEST_PAIRWISE_ERROR** 0x06
- #define **NDIS_802_11_AUTH_REQUEST_GROUP_ERROR** 0x0E
- #define **MAX_ADAPTERS** 32
- #define **INTF_ALL** 0xffffffff
- #define **INTF_ALL_FLAGS** 0x0000ffff
- #define **INTF_CTLFLAGS** 0x00000010
- #define **INTFCTL_ENABLED** 0x8000

## Typedefs

- typedef UCHAR **NDIS_802_11_MAC_ADDRESS** [6]
- typedef NDIS_802_11_SSID **NDIS_802_11_SSID**
- typedef LONG **NDIS_802_11_RSSI**
- typedef enum NDIS_802_11_NETWORK_TYPE **NDIS_802_11_NETWORK_TYPE**
- typedef NDIS_802_11_CONFIGURATION_FH **NDIS_802_11_CONFIGURATION_FH**
- typedef NDIS_802_11_CONFIGURATION **NDIS_802_11_CONFIGURATION**
- typedef enum NDIS_802_11_NETWORK_INFRASTRUCTURE **NDIS_802_11_NETWORK_-INFRASTRUCTURE**
- typedef enum NDIS_802_11_AUTHENTICATION_MODE **NDIS_802_11_-AUTHENTICATION_MODE**
- typedef enum NDIS_802_11_WEP_STATUS **NDIS_802_11_WEP_STATUS**
- typedef enum NDIS_802_11_WEP_STATUS **NDIS_802_11_ENCRYPTION_STATUS**
- typedef enum NDIS_802_11_PRIVACY_FILTER **NDIS_802_11_PRIVACY_FILTER**
- typedef UCHAR **NDIS_802_11_RATES** [NDIS_802_11_LENGTH_RATES]
- typedef UCHAR **NDIS_802_11_RATES_EX** [NDIS_802_11_LENGTH_RATES_EX]
- typedef NDIS_WLAN_BSSID_EX **NDIS_WLAN_BSSID_EX**
- typedef NDIS_802_11_BSSID_LIST_EX **NDIS_802_11_BSSID_LIST_EX**
- typedef NDIS_802_11_FIXED_IEs **NDIS_802_11_FIXED_IEs**
- typedef NDIS_802_11_WEP **NDIS_802_11_WEP**
- typedef ULONG **NDIS_802_11_KEY_INDEX**
- typedef ULONGLONG **NDIS_802_11_KEY_RSC**
- typedef NDIS_802_11_KEY **NDIS_802_11_KEY**
- typedef NDIS_802_11_REMOVE_KEY **NDIS_802_11_REMOVE_KEY**
- typedef NDIS_802_11_AI_REQFI **NDIS_802_11_AI_REQFI**
- typedef NDIS_802_11_AI_RESFI **NDIS_802_11_AI_RESFI**

- typedef NDIS_802_11_ASSOCIATION_INFORMATION **NDIS_802_11_ASSOCIATION_-INFORMATION**
- typedef NDIS_802_11_AUTHENTICATION_ENCRYPTION **NDIS_802_11_-AUTHENTICATION_ENCRYPTION**
- typedef NDIS_802_11_CAPABILITY **NDIS_802_11_CAPABILITY**
- typedef UCHAR **NDIS_802_11_PMKID_VALUE** [16]
- typedef BSSID_INFO **BSSID_INFO**
- typedef NDIS_802_11_PMKID **NDIS_802_11_PMKID**
- typedef enum NDIS_802_11_STATUS_TYPE **NDIS_802_11_STATUS_TYPE**
- typedef NDIS_802_11_STATUS_INDICATION **NDIS_802_11_STATUS_INDICATION**
- typedef PMKID_CANDIDATE **PMKID_CANDIDATE**
- typedef NDIS_802_11_PMKID_CANDIDATE_LIST **NDIS_802_11_PMKID_CANDIDATE_-LIST**
- typedef NDIS_802_11_AUTHENTICATION_REQUEST **NDIS_802_11_AUTHENTICATION_-REQUEST**
- typedef * **PINTF_KEY_ENTRY**
- typedef * **PINTFS_KEY_TABLE**
- typedef * **PRAW_DATA**
- typedef * **PINTF_ENTRY**

## Enumerations

- enum **NDIS_802_11_NETWORK_TYPE** {

  **Ndis802_11FH**, **Ndis802_11DS**, **Ndis802_11OFDM5**, **Ndis802_11OFDM24**,

  **Ndis802_11NetworkTypeMax** }
- enum **NDIS_802_11_NETWORK_INFRASTRUCTURE** { **Ndis802_11IBSS**, **Ndis802_-11Infrastructure**, **Ndis802_11AutoUnknown**, **Ndis802_11InfrastructureMax** }
- enum **NDIS_802_11_AUTHENTICATION_MODE** {

  **Ndis802_11AuthModeOpen**, **Ndis802_11AuthModeShared**, **Ndis802_11AuthModeAuto-Switch**, **Ndis802_11AuthModeWPA**,

  **Ndis802_11AuthModeWPAPSK**, **Ndis802_11AuthModeWPANone**, **Ndis802_11AuthMode-WPA2**, **Ndis802_11AuthModeWPA2PSK**,

  **Ndis802_11AuthModeMax** }
- enum **NDIS_802_11_WEP_STATUS** {

  **Ndis802_11WEPEnabled**, **Ndis802_11Encryption1Enabled** = Ndis802_11WEPEnabled, **Ndis802_11WEPDisabled**, **Ndis802_11EncryptionDisabled** = Ndis802_11WEPDisabled,

  **Ndis802_11WEPKeyAbsent**, **Ndis802_11Encryption1KeyAbsent** = Ndis802_11WEPKey-Absent, **Ndis802_11WEPNotSupported**, **Ndis802_11EncryptionNotSupported** = Ndis802_-11WEPNotSupported,

  **Ndis802_11Encryption2Enabled**, **Ndis802_11Encryption2KeyAbsent**, **Ndis802_-11Encryption3Enabled**, **Ndis802_11Encryption3KeyAbsent** }
- enum **NDIS_802_11_PRIVACY_FILTER** { **Ndis802_11PrivFilterAcceptAll**, **Ndis802_11Priv-Filter8021xWEP** }
- enum **NDIS_802_11_STATUS_TYPE** { **Ndis802_11StatusType_Authentication**, **Ndis802_-11StatusType_PMKID_CandidateList** = 2, **Ndis802_11StatusTypeMax** }

## Functions

- int **wpa_driver_register_event_cb** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_pipe_cb** (void *eloop_data, void *user_data)
- void **wpa_driver_ndis_event_connect** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_disconnect** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_media_specific** (struct wpa_driver_ndis_data *drv, const u8 *data, size_t data_len)
- void **wpa_driver_ndis_event_adapter_arrival** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_adapter_removal** (struct wpa_driver_ndis_data *drv)

## Variables

- const struct wpa_driver_ops **wpa_driver_ndis_ops**

### 6.50.1 Detailed Description

WPA Supplicant - Windows/NDIS driver interface.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

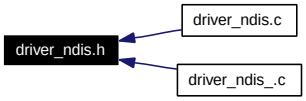
See README and COPYING for more details.

Definition in file driver_ndis.c.

## 6.51 driver_ndis.h File Reference

WPA Supplicant - Windows/NDIS driver interface.

This graph shows which files directly or indirectly include this file:




### Functions

- ndis_events_data * **ndis_events_init** (HANDLE *read_pipe, HANDLE *event, const char *ifname, const char *desc)
- void **ndis_events_deinit** (struct ndis_events_data *events)

### 6.51.1 Detailed Description

WPA Supplicant - Windows/NDIS driver interface.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

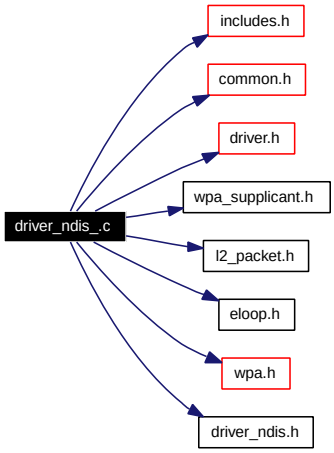
Definition in file driver_ndis.h.

## 6.52 driver_ndis_.c File Reference

WPA Supplicant - Windows/NDIS driver interface - event processing.

```
#include "includes.h"
#include "common.h"
#include "driver.h"
#include "wpa_supplicant.h"
#include "l2_packet.h"
#include "eloop.h"
#include "wpa.h"
#include "driver_ndis.h"
```

Include dependency graph for driver_ndis_.c:




### Typedefs

- typedef _ADAPTER * **LPADAPTER**

### Enumerations

- enum **event_types** {

  **EVENT_CONNECT**, **EVENT_DISCONNECT**, **EVENT_MEDIA_SPECIFIC**, **EVENT_ADAPTER_ARRIVAL**,

  **EVENT_ADAPTER_REMOVAL** }

### Functions

- void **wpa_driver_ndis_event_connect** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_disconnect** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_media_specific** (struct wpa_driver_ndis_data *drv, const u8 *data, size_t data_len)

- void **wpa_driver_ndis_event_adapter_arrival** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_adapter_removal** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_pipe_cb** (void *eloop_data, void *user_data)

### 6.52.1 Detailed Description

WPA Supplicant - Windows/NDIS driver interface - event processing.

**Copyright**

Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

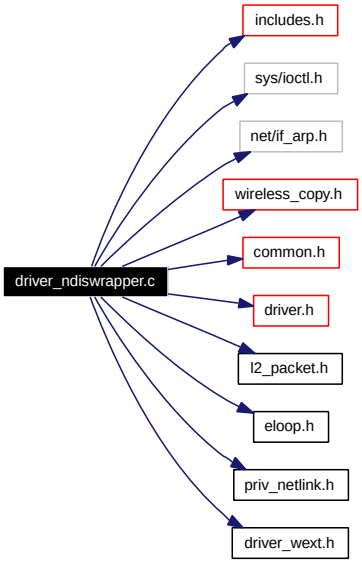
Definition in file driver_ndis_.c.

# 6.53 driver_ndiswrapper.c File Reference

WPA Supplicant - driver interaction with Linux ndiswrapper.

```
#include "includes.h"
#include <sys/ioctl.h>
#include <net/if_arp.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "l2_packet.h"
#include "eloop.h"
#include "priv_netlink.h"
#include "driver_wext.h"
```

Include dependency graph for driver_ndiswrapper.c:




## Defines

- #define **PRIV_RESET** SIOCIWFIRSTPRIV+0
- #define **WPA_SET_WPA** SIOCIWFIRSTPRIV+1
- #define **WPA_SET_KEY** SIOCIWFIRSTPRIV+2
- #define **WPA_ASSOCIATE** SIOCIWFIRSTPRIV+3
- #define **WPA_DISASSOCIATE** SIOCIWFIRSTPRIV+4
- #define **WPA_DROP_UNENCRYPTED** SIOCIWFIRSTPRIV+5
- #define **WPA_SET_COUNTERMEASURES** SIOCIWFIRSTPRIV+6
- #define **WPA_DEAUTHENTICATE** SIOCIWFIRSTPRIV+7
- #define **WPA_SET_AUTH_ALG** SIOCIWFIRSTPRIV+8
- #define **WPA_INIT** SIOCIWFIRSTPRIV+9

- #define **WPA_DEINIT** SIOCIWFIRSTPRIV+10
- #define **WPA_GET_CAPA** SIOCIWFIRSTPRIV+11

## Variables

- const struct wpa_driver_ops **wpa_driver_ndiswrapper_ops**

### 6.53.1 Detailed Description

WPA Supplicant - driver interaction with Linux ndiswrapper.

**Copyright**

Copyright (c) 2004-2006, Giridhar Pemmasani <giri@lmc.cs.sunysb.edu> Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_ndiswrapper.c.

### 6.53.2 Variable Documentation

#### 6.53.2.1 const struct wpa_driver_ops wpa_driver_ndiswrapper_ops

**Initial value:**

```
{
        .name = "ndiswrapper",
        .desc = "Linux ndiswrapper",
        .set_wpa = wpa_ndiswrapper_set_wpa,
        .set_key = wpa_ndiswrapper_set_key,
        .set_countermeasures = wpa_ndiswrapper_set_countermeasures,
        .set_drop_unencrypted = wpa_ndiswrapper_set_drop_unencrypted,
        .deauthenticate = wpa_ndiswrapper_deauthenticate,
        .disassociate = wpa_ndiswrapper_disassociate,
        .associate = wpa_ndiswrapper_associate,
        .set_auth_alg = wpa_ndiswrapper_set_auth_alg,

        .get_bssid = wpa_ndiswrapper_get_bssid,
        .get_ssid = wpa_ndiswrapper_get_ssid,
        .scan = wpa_ndiswrapper_scan,
        .get_scan_results = wpa_ndiswrapper_get_scan_results,
        .init = wpa_ndiswrapper_init,
        .deinit = wpa_ndiswrapper_deinit,
        .get_capa = wpa_ndiswrapper_get_capa,
        .set_operstate = wpa_ndiswrapper_set_operstate,
}
```

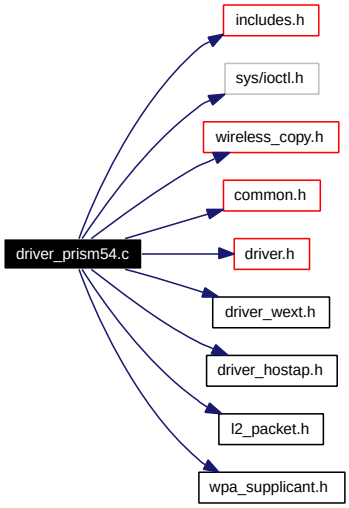
Definition at line 353 of file driver_ndiswrapper.c.

## 6.54 driver_prism54.c File Reference

WPA Supplicant - driver interaction with Linux Prism54.org driver.

```
#include "includes.h"
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
#include "driver_hostap.h"
#include "l2_packet.h"
#include "wpa_supplicant.h"
```

Include dependency graph for driver_prism54.c:




### Defines

- #define **PRISM54_SET_WPA** SIOCIWFIRSTPRIV+12
- #define **PRISM54_HOSTAPD** SIOCIWFIRSTPRIV+25
- #define **PRISM54_DROP_UNENCRYPTED** SIOCIWFIRSTPRIV+26

### Variables

- const struct wpa_driver_ops **wpa_driver_prism54_ops**

### 6.54.1 Detailed Description

WPA Supplicant - driver interaction with Linux Prism54.org driver.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi> Copyright (c) 2004, Luis R. Rodriguez <mcgrof@ruslug.rutgers.edu>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_prism54.c.

## 6.54.2 Variable Documentation

### 6.54.2.1 const struct wpa_driver_ops wpa_driver_prism54_ops

**Initial value:**

```
{
        .name = "prism54",
        .desc = "Prism54.org driver (Intersil Prism GT/Duette/Indigo)",
        .get_bssid = wpa_driver_prism54_get_bssid,
        .get_ssid = wpa_driver_prism54_get_ssid,
        .set_wpa = wpa_driver_prism54_set_wpa,
        .set_key = wpa_driver_prism54_set_key,
        .set_countermeasures = wpa_driver_prism54_set_countermeasures,
        .set_drop_unencrypted = wpa_driver_prism54_set_drop_unencrypted,
        .scan = wpa_driver_prism54_scan,
        .get_scan_results = wpa_driver_prism54_get_scan_results,
        .deauthenticate = wpa_driver_prism54_deauthenticate,
        .disassociate = wpa_driver_prism54_disassociate,
        .associate = wpa_driver_prism54_associate,
        .init = wpa_driver_prism54_init,
        .deinit = wpa_driver_prism54_deinit,
        .set_operstate = wpa_driver_prism54_set_operstate,
}
```

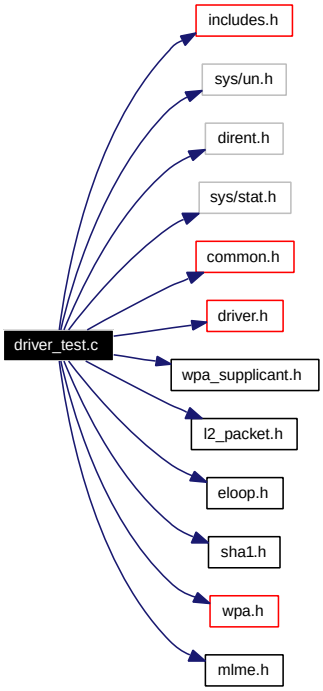
Definition at line 368 of file driver_prism54.c.

## 6.55 driver_test.c File Reference

WPA Supplicant - testing driver interface.

```
#include "includes.h"
#include <sys/un.h>
#include <dirent.h>
#include <sys/stat.h>
#include "common.h"
#include "driver.h"
#include "wpa_supplicant.h"
#include "l2_packet.h"
#include "eloop.h"
#include "sha1.h"
#include "wpa.h"
#include "mlme.h"
```

Include dependency graph for driver_test.c:




### Defines

- #define **MAX_SCAN_RESULTS** 30

## Variables

- const struct wpa_driver_ops **wpa_driver_test_ops**

### 6.55.1 Detailed Description

WPA Supplicant - testing driver interface.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

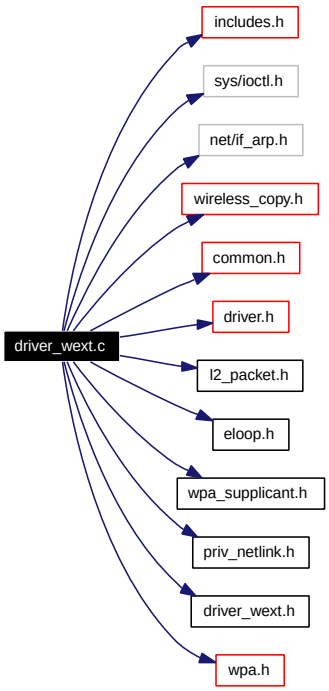
Definition in file driver_test.c.

## 6.56 driver_wext.c File Reference

WPA Supplicant - driver interaction with generic Linux Wireless Extensions.

```
#include "includes.h"
#include <sys/ioctl.h>
#include <net/if_arp.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "l2_packet.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "priv_netlink.h"
#include "driver_wext.h"
#include "wpa.h"
```

Include dependency graph for driver_wext.c:




### Functions

- int wpa_driver_wext_get_bssid (void *priv, u8 *bssid)
  *Get BSSID, SIOCGIWAP.*
- int wpa_driver_wext_set_bssid (void *priv, const u8 *bssid)

*Set BSSID, SIOCSIWAP.*

- int wpa_driver_wext_get_ssid (void *priv, u8 *ssid)

  *Get SSID, SIOCGIWESSID.*

- int wpa_driver_wext_set_ssid (void *priv, const u8 *ssid, size_t ssid_len)

  *Set SSID, SIOCSIWESSID.*

- int wpa_driver_wext_set_freq (void *priv, int freq)

  *Set frequency/channel, SIOCSIWFREQ.*

- int wpa_driver_wext_get_ifflags (struct wpa_driver_wext_data *drv, int *flags)

  *Get interface flags (SIOCGIFFLAGS).*

- int wpa_driver_wext_set_ifflags (struct wpa_driver_wext_data *drv, int flags)

  *Set interface flags (SIOCSIFFLAGS).*

- void * wpa_driver_wext_init (void *ctx, const char *ifname)

  *Initialize WE driver interface.*

- void wpa_driver_wext_deinit (void *priv)

  *Deinitialize WE driver interface.*

- void wpa_driver_wext_scan_timeout (void *eloop_ctx, void *timeout_ctx)

  *Scan timeout to report scan completion.*

- int wpa_driver_wext_scan (void *priv, const u8 *ssid, size_t ssid_len)

  *Request the driver to initiate scan.*

- int wpa_driver_wext_get_scan_results (void *priv, struct wpa_scan_result *results, size_t max_-size)

  *Fetch the latest scan results.*

- int wpa_driver_wext_set_key (void *priv, wpa_alg alg, const u8 *addr, int key_idx, int set_tx, const u8 *seq, size_t seq_len, const u8 *key, size_t key_len)

  *Configure encryption key.*

- int wpa_driver_wext_set_mode (void *priv, int mode)

  *Set wireless mode (infra/adhoc), SIOCSIWMODE.*

- int **wpa_driver_wext_alternative_ifindex** (struct wpa_driver_wext_data *drv, const char *ifname)
- int **wpa_driver_wext_set_operstate** (void *priv, int state)
- int **wpa_driver_wext_get_version** (struct wpa_driver_wext_data *drv)

## Variables

- const struct wpa_driver_ops **wpa_driver_wext_ops**

### 6.56.1 Detailed Description

WPA Supplicant - driver interaction with generic Linux Wireless Extensions.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements a driver interface for the Linux Wireless Extensions. When used with WE-18 or newer, this interface can be used as-is with number of drivers. In addition to this, some of the common functions in this file can be used by other driver interface implementations that use generic WE ioctls, but require private ioctls for some of the functionality.

Definition in file driver_wext.c.

### 6.56.2 Function Documentation

#### 6.56.2.1 void wpa_driver_wext_deinit (void * *priv*)

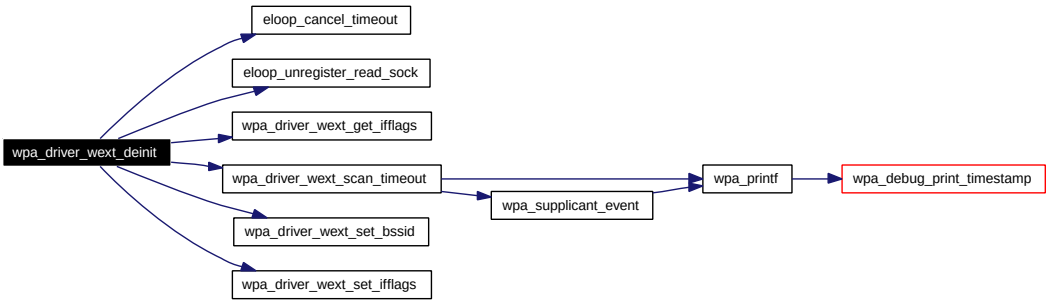
Deinitialize WE driver interface.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

Shut down driver interface and processing of driver events. Free private data buffer if one was allocated in wpa_driver_wext_init().

Definition at line 952 of file driver_wext.c.

Here is the call graph for this function:




#### 6.56.2.2 int wpa_driver_wext_get_bssid (void * *priv*, u8 * *bssid*)

Get BSSID, SIOCGIWAP.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

***bssid*** Buffer for BSSID

**Returns:**
0 on success, -1 on failure

Definition at line 164 of file driver_wext.c.

#### 6.56.2.3 int wpa_driver_wext_get_ifflags (struct wpa_driver_wext_data * *drv*, int * *flags*)

Get interface flags (SIOCGIFFLAGS).

**Parameters:**
***drv*** driver_wext private data

***flags*** Pointer to returned flags value

**Returns:**
0 on success, -1 on failure

Definition at line 819 of file driver_wext.c.

#### 6.56.2.4 int wpa_driver_wext_get_scan_results (void * *priv*, struct wpa_scan_result * *results*, size_t *max_size*)

Fetch the latest scan results.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

***results*** Pointer to buffer for scan results

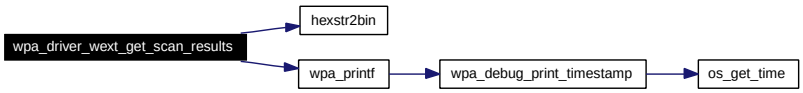
***max_size*** Maximum number of entries (buffer size)

**Returns:**
Number of scan result entries used on success, -1 on failure

If scan results include more than max_size BSSes, max_size will be returned and the remaining entries will not be included in the buffer.

Definition at line 1109 of file driver_wext.c.

Here is the call graph for this function:




#### 6.56.2.5 int wpa_driver_wext_get_ssid (void * *priv*, u8 * *ssid*)

Get SSID, SIOCGIWESSID.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

***ssid*** Buffer for the SSID; must be at least 32 bytes long

**Returns:**
SSID length on success, -1 on failure

Definition at line 220 of file driver_wext.c.

#### 6.56.2.6 void* wpa_driver_wext_init (void * *ctx*, const char * *ifname*)

Initialize WE driver interface.

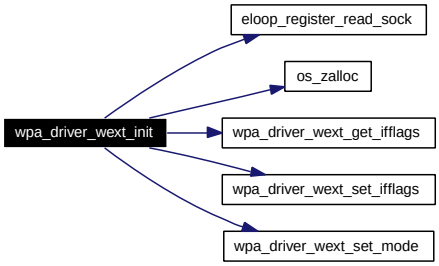
**Parameters:**
***ctx*** context to be used when calling wpa_supplicant functions, e.g., wpa_supplicant_event()

***ifname*** interface name, e.g., wlan0

**Returns:**
Pointer to private data, NULL on failure

Definition at line 862 of file driver_wext.c.

Here is the call graph for this function:




#### 6.56.2.7 int wpa_driver_wext_scan (void * *priv*, const u8 * *ssid*, size_t *ssid_len*)

Request the driver to initiate scan.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

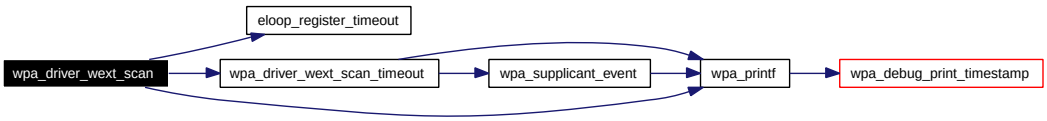
***ssid*** Specific SSID to scan for (ProbeReq) or NULL to scan for all SSIDs (either active scan with broadcast SSID or passive scan

***ssid_len*** Length of the SSID

**Returns:**
0 on success, -1 on failure

Definition at line 1017 of file driver_wext.c.

Here is the call graph for this function:




#### 6.56.2.8 void wpa_driver_wext_scan_timeout (void * *eloop_ctx*, void * *timeout_ctx*)

Scan timeout to report scan completion.

**Parameters:**

***eloop_ctx*** Unused

***timeout_ctx*** ctx argument given to wpa_driver_wext_init()

This function can be used as registered timeout when starting a scan to generate a scan completed event if the driver does not report this.

Definition at line 1000 of file driver_wext.c.

Here is the call graph for this function:


#### 6.56.2.9 int wpa_driver_wext_set_bssid (void * *priv*, const u8 * *bssid*)

Set BSSID, SIOCSIWAP.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***bssid*** BSSID

**Returns:**

0 on success, -1 on failure

Definition at line 190 of file driver_wext.c.

#### 6.56.2.10 int wpa_driver_wext_set_freq (void * *priv*, int *freq*)

Set frequency/channel, SIOCSIWFREQ.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***freq*** Frequency in MHz

**Returns:**

0 on success, -1 on failure

Definition at line 306 of file driver_wext.c.

#### 6.56.2.11 int wpa_driver_wext_set_ifflags (struct wpa_driver_wext_data * *drv*, int *flags*)

Set interface flags (SIOCSIFFLAGS).

**Parameters:**

***drv*** driver_wext private data

***flags*** New value for flags

**Returns:**

0 on success, -1 on failure

Definition at line 848 of file driver_wext.c.

#### 6.56.2.12 int wpa_driver_wext_set_key (void * *priv*, wpa_alg *alg*, const u8 * *addr*, int *key_idx*, int *set_tx*, const u8 * *seq*, size_t *seq_len*, const u8 * *key*, size_t *key_len*)

Configure encryption key.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***priv*** Private driver interface data

***alg*** Encryption algorithm (WPA_ALG_NONE, WPA_ALG_WEP, WPA_ALG_TKIP, WPA_ALG_-CCMP); WPA_ALG_NONE clears the key.

***addr*** Address of the peer STA or ff:ff:ff:ff:ff:ff for broadcast/default keys

***key_idx*** key index (0..3), usually 0 for unicast keys

***set_tx*** Configure this key as the default Tx key (only used when driver does not support separate unicast/individual key

***seq*** Sequence number/packet number, seq_len octets, the next packet number to be used for in replay protection; configured for Rx keys (in most cases, this is only used with broadcast keys and set to zero for unicast keys)

***seq_len*** Length of the seq, depends on the algorithm: TKIP: 6 octets, CCMP: 6 octets

***key*** Key buffer; TKIP: 16-byte temporal key, 8-byte Tx Mic key, 8-byte Rx Mic Key

***key_len*** Length of the key buffer in octets (WEP: 5 or 13, TKIP: 32, CCMP: 16)

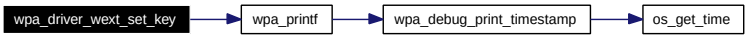
**Returns:**

0 on success, -1 on failure

This function uses SIOCSIWENCODEEXT by default, but tries to use SIOCSIWENCODE if the extended ioctl fails when configuring a WEP key.

Definition at line 1571 of file driver_wext.c.

Here is the call graph for this function:



#### 6.56.2.13 int wpa_driver_wext_set_mode (void * *priv*, int *mode*)

Set wireless mode (infra/adhoc), SIOCSIWMODE.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***mode*** 0 = infra/BSS (associate with an AP), 1 = adhoc/IBSS

**Returns:**

0 on success, -1 on failure

Definition at line 1909 of file driver_wext.c.

#### 6.56.2.14 int wpa_driver_wext_set_ssid (void * *priv*, const u8 * *ssid*, size_t *ssid_len*)

Set SSID, SIOCSIWESSID.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***ssid*** SSID

***ssid_len*** Length of SSID (0..32)

**Returns:**

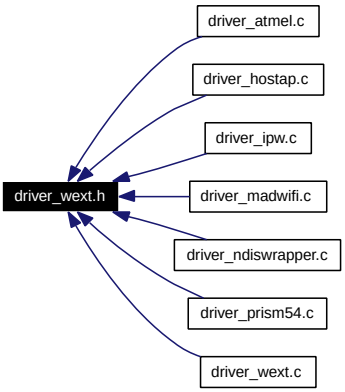
0 on success, -1 on failure

Definition at line 259 of file driver_wext.c.

# 6.57 driver_wext.h File Reference

WPA Supplicant - driver_wext exported functions.

This graph shows which files directly or indirectly include this file:




## Functions

- int wpa_driver_wext_get_ifflags (struct wpa_driver_wext_data *drv, int *flags)
  *Get interface flags (SIOCGIFFLAGS).*
- int wpa_driver_wext_set_ifflags (struct wpa_driver_wext_data *drv, int flags)
  *Set interface flags (SIOCSIFFLAGS).*
- int wpa_driver_wext_get_bssid (void *priv, u8 *bssid)
  *Get BSSID, SIOCGIWAP.*
- int wpa_driver_wext_set_bssid (void *priv, const u8 *bssid)
  *Set BSSID, SIOCSIWAP.*
- int wpa_driver_wext_get_ssid (void *priv, u8 *ssid)
  *Get SSID, SIOCGIWESSID.*
- int wpa_driver_wext_set_ssid (void *priv, const u8 *ssid, size_t ssid_len)
  *Set SSID, SIOCSIWESSID.*
- int wpa_driver_wext_set_freq (void *priv, int freq)
  *Set frequency/channel, SIOCSIWFREQ.*
- int wpa_driver_wext_set_mode (void *priv, int mode)
  *Set wireless mode (infra/adhoc), SIOCSIWMODE.*
- int wpa_driver_wext_set_key (void *priv, wpa_alg alg, const u8 *addr, int key_idx, int set_tx, const u8 *seq, size_t seq_len, const u8 *key, size_t key_len)
  *Configure encryption key.*
- int wpa_driver_wext_scan (void *priv, const u8 *ssid, size_t ssid_len)

*Request the driver to initiate scan.*

- int wpa_driver_wext_get_scan_results (void *priv, struct wpa_scan_result *results, size_t max_size)

  *Fetch the latest scan results.*

- void wpa_driver_wext_scan_timeout (void *eloop_ctx, void *timeout_ctx)

  *Scan timeout to report scan completion.*

- int **wpa_driver_wext_alternative_ifindex** (struct wpa_driver_wext_data *drv, const char *ifname)
- void * wpa_driver_wext_init (void *ctx, const char *ifname)

  *Initialize WE driver interface.*

- void wpa_driver_wext_deinit (void *priv)

  *Deinitialize WE driver interface.*

- int **wpa_driver_wext_set_operstate** (void *priv, int state)
- int **wpa_driver_wext_get_version** (struct wpa_driver_wext_data *drv)

### 6.57.1 Detailed Description

WPA Supplicant - driver_wext exported functions.

**Copyright**
Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_wext.h.

### 6.57.2 Function Documentation

#### 6.57.2.1 void wpa_driver_wext_deinit (void * *priv*)

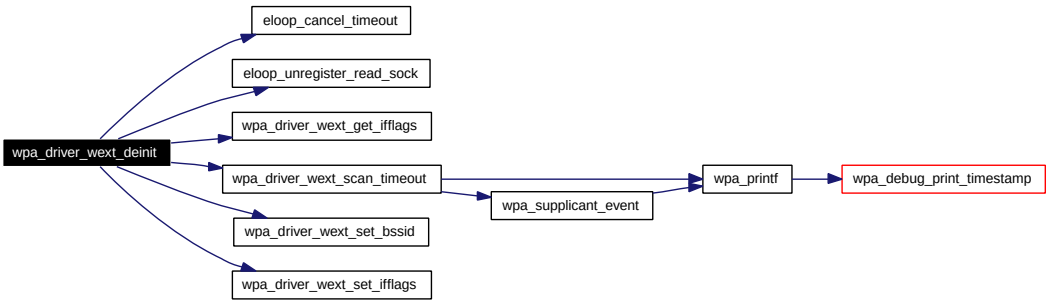
Deinitialize WE driver interface.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

Shut down driver interface and processing of driver events. Free private data buffer if one was allocated in wpa_driver_wext_init().

Definition at line 952 of file driver_wext.c.

Here is the call graph for this function:




#### 6.57.2.2 int wpa_driver_wext_get_bssid (void * *priv*, u8 * *bssid*)

Get BSSID, SIOCGIWAP.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***bssid*** Buffer for BSSID

**Returns:**

0 on success, -1 on failure

Definition at line 164 of file driver_wext.c.

#### 6.57.2.3 int wpa_driver_wext_get_ifflags (struct wpa_driver_wext_data * *drv*, int * *flags*)

Get interface flags (SIOCGIFFLAGS).

**Parameters:**

***drv*** driver_wext private data

***flags*** Pointer to returned flags value

**Returns:**

0 on success, -1 on failure

Definition at line 819 of file driver_wext.c.

#### 6.57.2.4 int wpa_driver_wext_get_scan_results (void * *priv*, struct wpa_scan_result * *results*, size_t *max_size*)

Fetch the latest scan results.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***results*** Pointer to buffer for scan results

***max_size*** Maximum number of entries (buffer size)

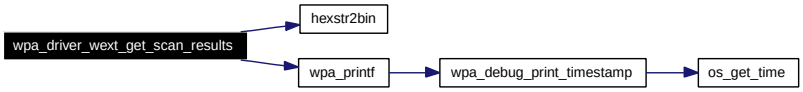
**Returns:**

Number of scan result entries used on success, -1 on failure

If scan results include more than max_size BSSes, max_size will be returned and the remaining entries will not be included in the buffer.

Definition at line 1109 of file driver_wext.c.

Here is the call graph for this function:




#### 6.57.2.5 int wpa_driver_wext_get_ssid (void * *priv*, u8 * *ssid*)

Get SSID, SIOCGIWESSID.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***ssid*** Buffer for the SSID; must be at least 32 bytes long

**Returns:**

SSID length on success, -1 on failure

Definition at line 220 of file driver_wext.c.

#### 6.57.2.6 void* wpa_driver_wext_init (void * *ctx*, const char * *ifname*)

Initialize WE driver interface.

**Parameters:**

***ctx*** context to be used when calling wpa_supplicant functions, e.g., wpa_supplicant_event()

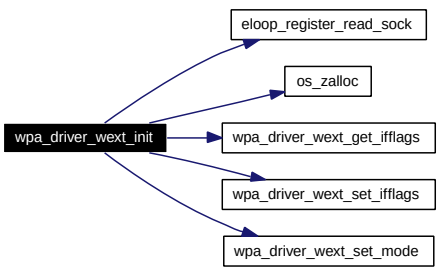
***ifname*** interface name, e.g., wlan0

**Returns:**

Pointer to private data, NULL on failure

Definition at line 862 of file driver_wext.c.

Here is the call graph for this function:



#### 6.57.2.7 int wpa_driver_wext_scan (void * *priv*, const u8 * *ssid*, size_t *ssid_len*)

Request the driver to initiate scan.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***ssid*** Specific SSID to scan for (ProbeReq) or NULL to scan for all SSIDs (either active scan with broadcast SSID or passive scan

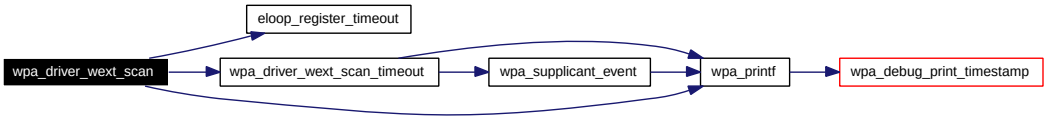
***ssid_len*** Length of the SSID

**Returns:**

0 on success, -1 on failure

Definition at line 1017 of file driver_wext.c.

Here is the call graph for this function:




#### 6.57.2.8 void wpa_driver_wext_scan_timeout (void * *eloop_ctx*, void * *timeout_ctx*)

Scan timeout to report scan completion.

**Parameters:**

***eloop_ctx*** Unused

***timeout_ctx*** ctx argument given to wpa_driver_wext_init()

This function can be used as registered timeout when starting a scan to generate a scan completed event if the driver does not report this.

Definition at line 1000 of file driver_wext.c.

Here is the call graph for this function:


#### 6.57.2.9 int wpa_driver_wext_set_bssid (void * *priv*, const u8 * *bssid*)

Set BSSID, SIOCSIWAP.

**Parameters:**

***priv*** Pointer to private wext data from wpa_driver_wext_init()

***bssid*** BSSID

**Returns:**
0 on success, -1 on failure

Definition at line 190 of file driver_wext.c.

#### 6.57.2.10 int wpa_driver_wext_set_freq (void * *priv*, int *freq*)

Set frequency/channel, SIOCSIWFREQ.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

***freq*** Frequency in MHz

**Returns:**
0 on success, -1 on failure

Definition at line 306 of file driver_wext.c.

#### 6.57.2.11 int wpa_driver_wext_set_ifflags (struct wpa_driver_wext_data * *drv*, int *flags*)

Set interface flags (SIOCSIFFLAGS).

**Parameters:**
***drv*** driver_wext private data

***flags*** New value for flags

**Returns:**
0 on success, -1 on failure

Definition at line 848 of file driver_wext.c.

#### 6.57.2.12 int wpa_driver_wext_set_key (void * *priv*, wpa_alg *alg*, const u8 * *addr*, int *key_idx*, int *set_tx*, const u8 * *seq*, size_t *seq_len*, const u8 * *key*, size_t *key_len*)

Configure encryption key.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

***priv*** Private driver interface data

***alg*** Encryption algorithm (WPA_ALG_NONE, WPA_ALG_WEP, WPA_ALG_TKIP, WPA_ALG_-CCMP); WPA_ALG_NONE clears the key.

***addr*** Address of the peer STA or ff:ff:ff:ff:ff:ff for broadcast/default keys

***key_idx*** key index (0..3), usually 0 for unicast keys

***set_tx*** Configure this key as the default Tx key (only used when driver does not support separate unicast/individual key

***seq*** Sequence number/packet number, seq_len octets, the next packet number to be used for in replay protection; configured for Rx keys (in most cases, this is only used with broadcast keys and set to zero for unicast keys)

***seq_len*** Length of the seq, depends on the algorithm: TKIP: 6 octets, CCMP: 6 octets

***key*** Key buffer; TKIP: 16-byte temporal key, 8-byte Tx Mic key, 8-byte Rx Mic Key

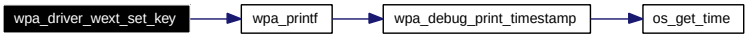
***key_len*** Length of the key buffer in octets (WEP: 5 or 13, TKIP: 32, CCMP: 16)

**Returns:**
0 on success, -1 on failure

This function uses SIOCSIWENCODEEXT by default, but tries to use SIOCSIWENCODE if the extended ioctl fails when configuring a WEP key.

Definition at line 1571 of file driver_wext.c.

Here is the call graph for this function:




#### 6.57.2.13 int wpa_driver_wext_set_mode (void * *priv*, int *mode*)

Set wireless mode (infra/adhoc), SIOCSIWMODE.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

***mode*** 0 = infra/BSS (associate with an AP), 1 = adhoc/IBSS

**Returns:**
0 on success, -1 on failure

Definition at line 1909 of file driver_wext.c.

#### 6.57.2.14 int wpa_driver_wext_set_ssid (void * *priv*, const u8 * *ssid*, size_t *ssid_len*)

Set SSID, SIOCSIWESSID.

**Parameters:**
***priv*** Pointer to private wext data from wpa_driver_wext_init()

***ssid*** SSID

***ssid_len*** Length of SSID (0..32)

**Returns:**
0 on success, -1 on failure

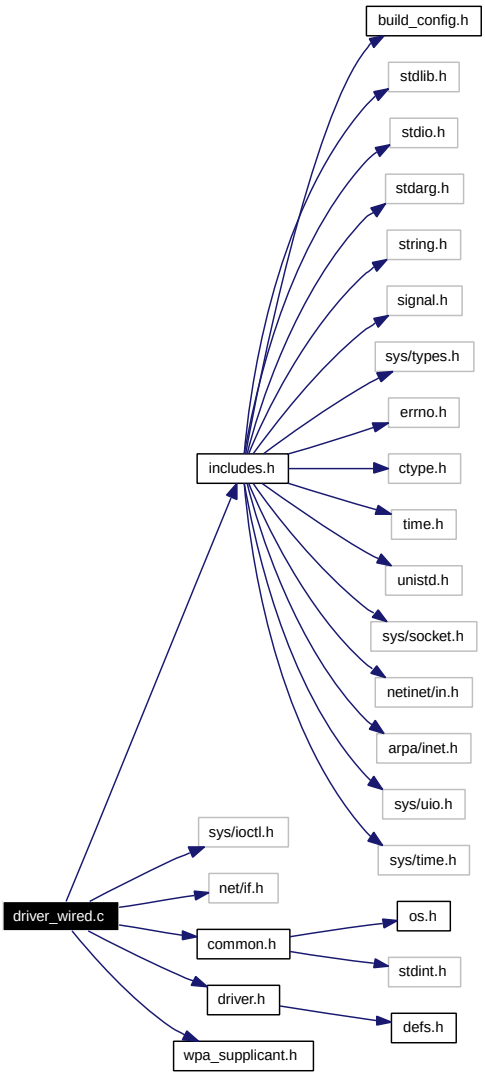
Definition at line 259 of file driver_wext.c.

## 6.58 driver_wired.c File Reference

WPA Supplicant - wired Ethernet driver interface.

```
#include "includes.h"
#include <sys/ioctl.h>
#include <net/if.h>
#include "common.h"
#include "driver.h"
#include "wpa_supplicant.h"
```

Include dependency graph for driver_wired.c:




### Variables

- const struct wpa_driver_ops **wpa_driver_wired_ops**

### 6.58.1 Detailed Description

WPA Supplicant - wired Ethernet driver interface.

**Copyright**
Copyright (c) 2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file driver_wired.c.

### 6.58.2 Variable Documentation

#### 6.58.2.1 const struct wpa_driver_ops wpa_driver_wired_ops

**Initial value:**

```
{
        .name = "wired",
        .desc = "wpa_supplicant wired Ethernet driver",
        .set_wpa = wpa_driver_wired_set_wpa,
        .get_ssid = wpa_driver_wired_get_ssid,
        .get_bssid = wpa_driver_wired_get_bssid,
        .init = wpa_driver_wired_init,
        .deinit = wpa_driver_wired_deinit,
}
```

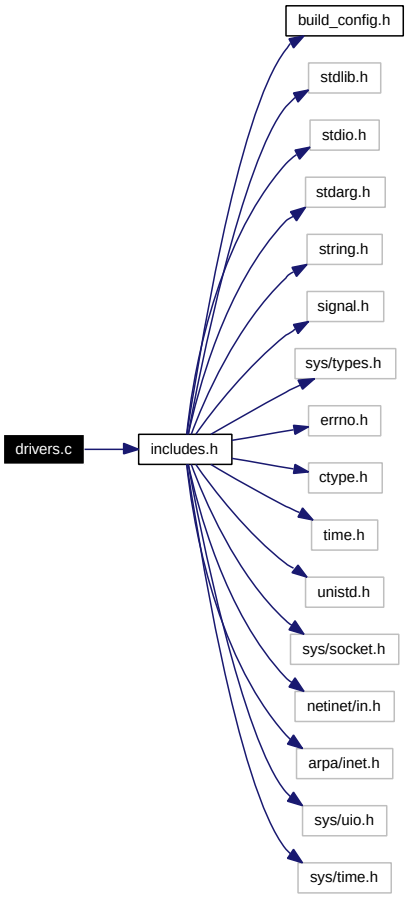
Definition at line 278 of file driver_wired.c.

## 6.59 drivers.c File Reference

WPA Supplicant / driver interface list.

```
#include "includes.h"
```

Include dependency graph for drivers.c:




### Variables

- wpa_driver_ops * **wpa_supplicant_drivers** [ ]

### 6.59.1 Detailed Description

WPA Supplicant / driver interface list.

**Copyright**

Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

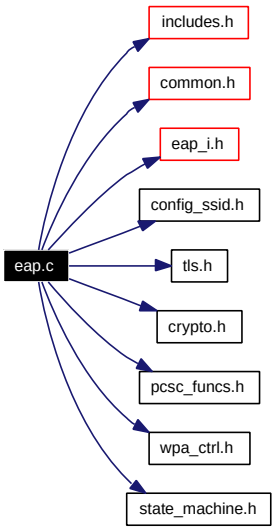
Definition in file drivers.c.

## 6.60 eap.c File Reference

EAP peer state machines (RFC 4137).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "config_ssid.h"
#include "tls.h"
#include "crypto.h"
#include "pcsc_funcs.h"
#include "wpa_ctrl.h"
#include "state_machine.h"
```

Include dependency graph for eap.c:




### Defines

- #define **STATE_MACHINE_DATA** struct eap_sm
- #define **STATE_MACHINE_DEBUG_PREFIX** "EAP"
- #define **EAP_MAX_AUTH_ROUNDS** 50

### Enumerations

- enum **eap_ctrl_req_type** {

  **TYPE_IDENTITY**, **TYPE_PASSWORD**, **TYPE_OTP**, **TYPE_PIN**,

  **TYPE_NEW_PASSWORD**, **TYPE_PASSPHRASE** }

## Functions

- **SM_STATE** (EAP, INITIALIZE)
- **SM_STATE** (EAP, DISABLED)
- **SM_STATE** (EAP, IDLE)
- **SM_STATE** (EAP, RECEIVED)
- **SM_STATE** (EAP, GET_METHOD)
- **SM_STATE** (EAP, METHOD)
- **SM_STATE** (EAP, SEND_RESPONSE)
- **SM_STATE** (EAP, DISCARD)
- **SM_STATE** (EAP, IDENTITY)
- **SM_STATE** (EAP, NOTIFICATION)
- **SM_STATE** (EAP, RETRANSMIT)
- **SM_STATE** (EAP, SUCCESS)
- **SM_STATE** (EAP, FAILURE)
- **SM_STEP** (EAP)
- u8 * eap_sm_buildIdentity (struct eap_sm *sm, int id, size_t *len, int encrypted)

  *Build EAP-Identity/Response for the current network.*

- eap_sm * eap_sm_init (void *eapol_ctx, struct eapol_callbacks *eapol_cb, void *msg_ctx, struct eap_config *conf)

  *Allocate and initialize EAP state machine.*

- void eap_sm_deinit (struct eap_sm *sm)

  *Deinitialize and free an EAP state machine.*

- int eap_sm_step (struct eap_sm *sm)

  *Step EAP state machine.*

- void eap_sm_abort (struct eap_sm *sm)

  *Abort EAP authentication.*

- int eap_sm_get_status (struct eap_sm *sm, char *buf, size_t buflen, int verbose)

  *Get EAP state machine status.*

- void eap_sm_request_identity (struct eap_sm *sm)

  *Request identity from user (ctrl_iface).*

- void eap_sm_request_password (struct eap_sm *sm)

  *Request password from user (ctrl_iface).*

- void eap_sm_request_new_password (struct eap_sm *sm)

  *Request new password from user (ctrl_iface).*

- void eap_sm_request_pin (struct eap_sm *sm)

  *Request SIM or smart card PIN from user (ctrl_iface).*

- void eap_sm_request_otp (struct eap_sm *sm, const char *msg, size_t msg_len)

  *Request one time password from user (ctrl_iface).*

- void eap_sm_request_passphrase (struct eap_sm *sm)
  *Request passphrase from user (ctrl_iface).*
- void eap_sm_notify_ctrl_attached (struct eap_sm *sm)
  *Notification of attached monitor.*
- u32 eap_get_phase2_type (const char *name, int *vendor)
  *Get EAP type for the given EAP phase 2 method name.*
- eap_method_type * eap_get_phase2_types (struct wpa_ssid *config, size_t *count)
  *Get list of allowed EAP phase 2 types.*
- void eap_set_fast_reauth (struct eap_sm *sm, int enabled)
  *Update fast_reauth setting.*
- void eap_set_workaround (struct eap_sm *sm, unsigned int workaround)
  *Update EAP workarounds setting.*
- wpa_ssid * eap_get_config (struct eap_sm *sm)
  *Get current network configuration.*
- const u8 * eap_get_config_identity (struct eap_sm *sm, size_t *len)
  *Get identity from the network configuration.*
- const u8 * eap_get_config_password (struct eap_sm *sm, size_t *len)
  *Get password from the network configuration.*
- const u8 * eap_get_config_new_password (struct eap_sm *sm, size_t *len)
  *Get new password from network configuration.*
- const u8 * eap_get_config_otp (struct eap_sm *sm, size_t *len)
  *Get one-time password from the network configuration.*
- void eap_clear_config_otp (struct eap_sm *sm)
  *Clear used one-time password.*
- int eap_key_available (struct eap_sm *sm)
  *Get key availability (eapKeyAvailable variable).*
- void eap_notify_success (struct eap_sm *sm)
  *Notify EAP state machine about external success trigger.*
- void eap_notify_lower_layer_success (struct eap_sm *sm)
  *Notification of lower layer success.*
- const u8 * eap_get_eapKeyData (struct eap_sm *sm, size_t *len)
  *Get master session key (MSK) from EAP state machine.*
- u8 * eap_get_eapRespData (struct eap_sm *sm, size_t *len)
  *Get EAP response data.*

- void eap_register_scard_ctx (struct eap_sm *sm, void *ctx)

  *Notification of smart card context.*

- const u8 * eap_hdr_validate (int vendor, EapType eap_type, const u8 *msg, size_t msglen, size_t *plen)

  *Validate EAP header.*

- void eap_set_config_blob (struct eap_sm *sm, struct wpa_config_blob *blob)

  *Set or add a named configuration blob.*

- const struct wpa_config_blob * eap_get_config_blob (struct eap_sm *sm, const char *name)

  *Get a named configuration blob.*

- void eap_set_force_disabled (struct eap_sm *sm, int disabled)

  *Set force_disabled flag.*

- eap_hdr * eap_msg_alloc (int vendor, EapType type, size_t *len, size_t payload_len, u8 code, u8 identifier, u8 **payload)

  *Allocate a buffer for an EAP message.*

- void eap_notify_pending (struct eap_sm *sm)

  *Notify that EAP method is ready to re-process a request.*

- void eap_invalidate_cached_session (struct eap_sm *sm)

  *Mark cached session data invalid.*

### 6.60.1 Detailed Description

EAP peer state machines (RFC 4137).

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements the Peer State Machine as defined in RFC 4137. The used states and state transitions match mostly with the RFC. However, there are couple of additional transitions for working around small issues noticed during testing. These exceptions are explained in comments within the functions in this file. The method functions, m.func(), are similar to the ones used in RFC 4137, but some small changes have used here to optimize operations and to add functionality needed for fast re-authentication (session resumption).

Definition in file eap.c.

## 6.60.2 Function Documentation

### 6.60.2.1 void eap_clear_config_otp (struct eap_sm * *sm*)

Clear used one-time password.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

This function clears a used one-time password (OTP) from the current network configuration. This should be called when the OTP has been used and is not needed anymore.

Definition at line 1824 of file eap.c.

Here is the call graph for this function:


### 6.60.2.2 struct wpa_ssid* eap_get_config (struct eap_sm * *sm*)

Get current network configuration.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

**Returns:**
Pointer to the current network configuration or NULL if not found

EAP peer methods should avoid using this function if they can use other access functions, like eap_get_config_identity() and eap_get_config_password(), that do not require direct access to struct wpa_ssid.

Definition at line 1741 of file eap.c.

### 6.60.2.3 const struct wpa_config_blob* eap_get_config_blob (struct eap_sm * *sm*, const char * *name*)

Get a named configuration blob.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***name*** Name of the blob

**Returns:**
Pointer to blob data or NULL if not found

Definition at line 2059 of file eap.c.

#### 6.60.2.4 const u8* eap_get_config_identity (struct eap_sm * *sm*, size_t * *len*)

Get identity from the network configuration.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Buffer for the length of the identity

**Returns:**

Pointer to the identity or NULL if not found

Definition at line 1754 of file eap.c.

Here is the call graph for this function:


#### 6.60.2.5 const u8* eap_get_config_new_password (struct eap_sm * *sm*, size_t * *len*)

Get new password from network configuration.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Buffer for the length of the new password

**Returns:**

Pointer to the new password or NULL if not found

Definition at line 1788 of file eap.c.

Here is the call graph for this function:


#### 6.60.2.6 const u8* eap_get_config_otp (struct eap_sm * *sm*, size_t * *len*)

Get one-time password from the network configuration.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Buffer for the length of the one-time password

**Returns:**

Pointer to the one-time password or NULL if not found

Definition at line 1805 of file eap.c.

Here is the call graph for this function:


#### 6.60.2.7 const u8* eap_get_config_password (struct eap_sm * *sm*, size_t * *len*)

Get password from the network configuration.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Buffer for the length of the password

**Returns:**

Pointer to the password or NULL if not found

Definition at line 1771 of file eap.c.

Here is the call graph for this function:


#### 6.60.2.8 const u8* eap_get_eapKeyData (struct eap_sm * *sm*, size_t * *len*)

Get master session key (MSK) from EAP state machine.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Pointer to variable that will be set to number of bytes in the key

**Returns:**

Pointer to the EAP keying data or NULL on failure

Fetch EAP keying material (MSK, eapKeyData) from the EAP state machine. The key is available only after a successful authentication. EAP state machine continues to manage the key data and the caller must not change or free the returned data.

Definition at line 1907 of file eap.c.

#### 6.60.2.9 u8* eap_get_eapRespData (struct eap_sm * *sm*, size_t * *len*)

Get EAP response data.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Pointer to variable that will be set to the length of the response

**Returns:**
Pointer to the EAP response (eapRespData) or NULL on failure

Fetch EAP response (eapRespData) from the EAP state machine. This data is available when EAP state machine has processed an incoming EAP request. The EAP state machine does not maintain a reference to the response after this function is called and the caller is responsible for freeing the data.

Definition at line 1931 of file eap.c.

#### 6.60.2.10 u32 eap_get_phase2_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP phase 2 method name.

**Parameters:**
***name*** EAP method name, e.g., MD5
***vendor*** Buffer for returning EAP Vendor-Id

**Returns:**
EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers that are allowed for Phase 2, i.e., for tunneled authentication. Phase 2 is used, e.g., with EAP-PEAP, EAP-TTLS, and EAP-FAST.

Definition at line 1648 of file eap.c.

Here is the call graph for this function:


#### 6.60.2.11 struct eap_method_type* eap_get_phase2_types (struct wpa_ssid * *config*, size_t * *count*)

Get list of allowed EAP phase 2 types.

**Parameters:**
***config*** Pointer to a network configuration
***count*** Pointer to a variable to be filled with number of returned EAP types

**Returns:**
Pointer to allocated type list or NULL on failure

This function generates an array of allowed EAP phase 2 (tunneled) types for the given network configuration.

Definition at line 1671 of file eap.c.

Here is the call graph for this function:

#### 6.60.2.12 const u8* eap_hdr_validate (int *vendor*, EapType *eap_type*, const u8 * *msg*, size_t *msglen*, size_t * *plen*)

Validate EAP header.

**Parameters:**

***vendor*** Expected EAP Vendor-Id (0 = IETF)

***eap_type*** Expected EAP type number

***msg*** EAP frame (starting with EAP header)

***msglen*** Length of msg

***plen*** Pointer to variable to contain the returned payload length

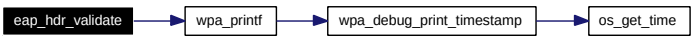
**Returns:**

Pointer to EAP payload (after type field), or NULL on failure

This is a helper function for EAP method implementations. This is usually called in the beginning of struct eap_method::process() function to verify that the received EAP request packet has a valid header. This function is able to process both legacy and expanded EAP headers and in most cases, the caller can just use the returned payload pointer (into *plen) for processing the payload regardless of whether the packet used the expanded EAP header or not.

Definition at line 1983 of file eap.c.

Here is the call graph for this function:




#### 6.60.2.13 void eap_invalidate_cached_session (struct eap_sm * *sm*)

Mark cached session data invalid.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

Definition at line 2153 of file eap.c.

#### 6.60.2.14 int eap_key_available (struct eap_sm * *sm*)

Get key availability (eapKeyAvailable variable).

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

**Returns:**

1 if EAP keying material is available, 0 if not

Definition at line 1842 of file eap.c.

#### 6.60.2.15 struct eap_hdr* eap_msg_alloc (int *vendor*, EapType *type*, size_t * *len*, size_t *payload_len*, u8 *code*, u8 *identifier*, u8 ** *payload*)

Allocate a buffer for an EAP message.

**Parameters:**

***vendor*** Vendor-Id (0 = IETF)

***type*** EAP type

***len*** Buffer for returning message length

***payload_len*** Payload length in bytes (data after Type)

***code*** Message Code (EAP_CODE_*)

***identifier*** Identifier

***payload*** Pointer to payload pointer that will be set to point to the beginning of the payload or NULL if payload pointer is not needed

**Returns:**

Pointer to the allocated message buffer or NULL on error

This function can be used to allocate a buffer for an EAP message and fill in the EAP header. This function is automatically using expanded EAP header if the selected Vendor-Id is not IETF. In other words, most EAP methods do not need to separately select which header type to use when using this function to allocate the message buffers.

Definition at line 2100 of file eap.c.

#### 6.60.2.16 void eap_notify_lower_layer_success (struct eap_sm * *sm*)

Notification of lower layer success.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

Notify EAP state machines that a lower layer has detected a successful authentication. This is used to recover from dropped EAP-Success messages.

Definition at line 1875 of file eap.c.

#### 6.60.2.17 void eap_notify_pending (struct eap_sm * *sm*)

Notify that EAP method is ready to re-process a request.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

An EAP method can perform a pending operation (e.g., to get a response from an external process). Once the response is available, this function can be used to request EAPOL state machine to retry delivering the previously received (and still unanswered) EAP request to EAP state machine.

Definition at line 2142 of file eap.c.

#### 6.60.2.18 void eap_notify_success (struct eap_sm * *sm*)

Notify EAP state machine about external success trigger.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

This function is called when external event, e.g., successful completion of WPA-PSK key handshake, is indicating that EAP state machine should move to success state. This is mainly used with security modes that do not use EAP state machine (e.g., WPA-PSK).

Definition at line 1858 of file eap.c.

#### 6.60.2.19 void eap_register_scard_ctx (struct eap_sm * *sm*, void * *ctx*)

Notification of smart card context.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***ctx*** Context data for smart card operations

Notify EAP state machines of context data for smart card operations. This context data will be used as a parameter for scard_*() functions.

Definition at line 1958 of file eap.c.

#### 6.60.2.20 void eap_set_config_blob (struct eap_sm * *sm*, struct wpa_config_blob * *blob*)

Set or add a named configuration blob.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***blob*** New value for the blob

Adds a new configuration blob or replaces the current value of an existing blob.

Definition at line 2046 of file eap.c.

#### 6.60.2.21 void eap_set_fast_reauth (struct eap_sm * *sm*, int *enabled*)

Update fast_reauth setting.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***enabled*** 1 = Fast reauthentication is enabled, 0 = Disabled

Definition at line 1712 of file eap.c.

#### 6.60.2.22 void eap_set_force_disabled (struct eap_sm * *sm*, int *disabled*)

Set force_disabled flag.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***disabled*** 1 = EAP disabled, 0 = EAP enabled

This function is used to force EAP state machine to be disabled when it is not in use (e.g., with WPA-PSK or plaintext connections).

Definition at line 2075 of file eap.c.

#### 6.60.2.23 void eap_set_workaround (struct eap_sm * *sm*, unsigned int *workaround*)

Update EAP workarounds setting.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***workaround*** 1 = Enable EAP workarounds, 0 = Disable EAP workarounds

Definition at line 1724 of file eap.c.

#### 6.60.2.24 void eap_sm_abort (struct eap_sm * *sm*)

Abort EAP authentication.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

Release system resources that have been allocated for the authentication session without fully deinitializing the EAP state machine.

Definition at line 1235 of file eap.c.

#### 6.60.2.25 u8* eap_sm_buildIdentity (struct eap_sm * *sm*, int *id*, size_t * *len*, int *encrypted*)

Build EAP-Identity/Response for the current network.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***id*** EAP identifier for the packet

***len*** Pointer to a variable that will be set to the length of the response

***encrypted*** Whether the packet is for encrypted tunnel (EAP phase 2)

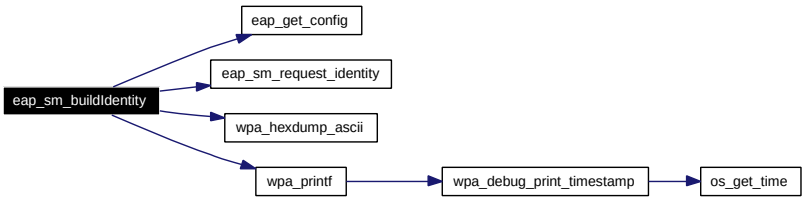
**Returns:**

Pointer to the allocated EAP-Identity/Response packet or NULL on failure

This function allocates and builds an EAP-Identity/Response packet for the current network. The caller is responsible for freeing the returned data.

Definition at line 932 of file eap.c.

Here is the call graph for this function:




#### 6.60.2.26 void eap_sm_deinit (struct eap_sm * *sm*)

Deinitialize and free an EAP state machine.

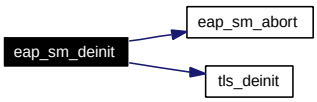
**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

This function deinitializes EAP state machine and frees all allocated resources.

Definition at line 1193 of file eap.c.

Here is the call graph for this function:




#### 6.60.2.27 int eap_sm_get_status (struct eap_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get EAP state machine status.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

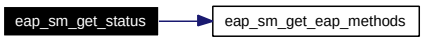
**Returns:**

Number of bytes written to buf.

Query EAP state machine for status information. This function fills in a text area with current status information from the EAPOL state machine. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

Definition at line 1340 of file eap.c.

Here is the call graph for this function:




#### 6.60.2.28 struct eap_sm* eap_sm_init (void * *eapol_ctx*, struct eapol_callbacks * *eapol_cb*, void * *msg_ctx*, struct eap_config * *conf*)

Allocate and initialize EAP state machine.

**Parameters:**

***eapol_ctx*** Context data to be used with eapol_cb calls

***eapol_cb*** Pointer to EAPOL callback functions

***msg_ctx*** Context data for wpa_msg() calls

***conf*** EAP configuration

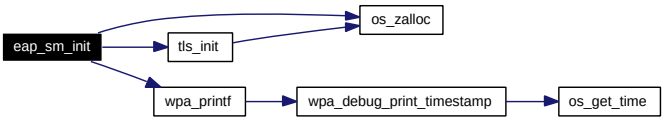
**Returns:**

Pointer to the allocated EAP state machine or NULL on failure

This function allocates and initializes an EAP state machine. In addition, this initializes TLS library for the new EAP state machine. eapol_cb pointer will be in use until eap_sm_deinit() is used to deinitialize this EAP state machine. Consequently, the caller must make sure that this data structure remains alive while the EAP state machine is active.

Definition at line 1155 of file eap.c.

Here is the call graph for this function:




#### 6.60.2.29 void eap_sm_notify_ctrl_attached (struct eap_sm * *sm*)

Notification of attached monitor.

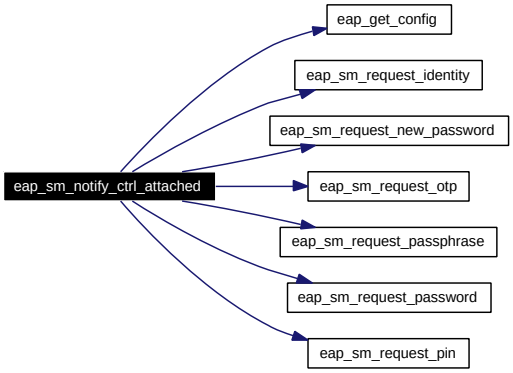
**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

Notify EAP state machines that a monitor was attached to the control interface to trigger re-sending of pending requests for user input.

Definition at line 1602 of file eap.c.

Here is the call graph for this function:




#### 6.60.2.30 void eap_sm_request_identity (struct eap_sm * *sm*)

Request identity from user (ctrl_iface).

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request identity information for the current network. This is normally called when the identity is not included in the network configuration. The request will be sent to monitor programs through the control interface.

Definition at line 1507 of file eap.c.

#### 6.60.2.31 void eap_sm_request_new_password (struct eap_sm * *sm*)

Request new password from user (ctrl_iface).

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request new password information for the current network. This is normally called when the EAP method indicates that the current password has expired and password change is required. The request will be sent to monitor programs through the control interface.

Definition at line 1539 of file eap.c.

#### 6.60.2.32 void eap_sm_request_otp (struct eap_sm * *sm*, const char * *msg*, size_t *msg_len*)

Request one time password from user (ctrl_iface).

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***msg*** Message to be displayed to the user when asking for OTP

***msg_len*** Length of the user displayable message

EAP methods can call this function to request open time password (OTP) for the current network. The request will be sent to monitor programs through the control interface.

Definition at line 1572 of file eap.c.

#### 6.60.2.33 void eap_sm_request_passphrase (struct eap_sm * *sm*)

Request passphrase from user (ctrl_iface).

**Parameters:**
*sm* Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request passphrase for a private key for the current network. This is normally called when the passphrase is not included in the network configuration. The request will be sent to monitor programs through the control interface.

Definition at line 1588 of file eap.c.

#### 6.60.2.34 void eap_sm_request_password (struct eap_sm * *sm*)

Request password from user (ctrl_iface).

**Parameters:**
*sm* Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request password information for the current network. This is normally called when the password is not included in the network configuration. The request will be sent to monitor programs through the control interface.

Definition at line 1523 of file eap.c.

#### 6.60.2.35 void eap_sm_request_pin (struct eap_sm * *sm*)

Request SIM or smart card PIN from user (ctrl_iface).

**Parameters:**
*sm* Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request SIM or smart card PIN information for the current network. This is normally called when the PIN is not included in the network configuration. The request will be sent to monitor programs through the control interface.

Definition at line 1555 of file eap.c.

#### 6.60.2.36 int eap_sm_step (struct eap_sm * *sm*)

Step EAP state machine.

**Parameters:**
*sm* Pointer to EAP state machine allocated with eap_sm_init()

**Returns:**
1 if EAP state was changed or 0 if not

This function advances EAP state machine to a new state to match with the current variables. This should be called whenever variables used by the EAP state machine have changed.

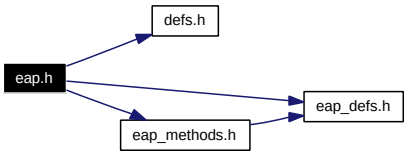
Definition at line 1214 of file eap.c.

## 6.61 eap.h File Reference

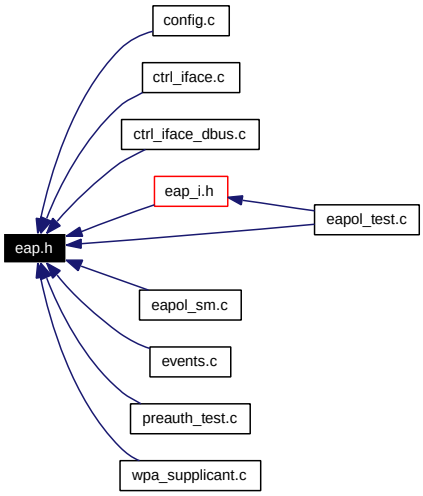
EAP peer state machine functions (RFC 4137).

```
#include "defs.h"
#include "eap_defs.h"
#include "eap_methods.h"
```

Include dependency graph for eap.h:




This graph shows which files directly or indirectly include this file:




### Enumerations

- enum eapol_bool_var { EAPOL_eapSuccess, EAPOL_eapRestart, EAPOL_eapFail, EAPOL_eapResp, EAPOL_eapNoResp, EAPOL_eapReq, EAPOL_portEnabled, EAPOL_altAccept, EAPOL_altReject }
- enum eapol_int_var { EAPOL_idleWhile }

### Functions

- eap_sm * eap_sm_init (void *eapol_ctx, struct eapol_callbacks *eapol_cb, void *msg_ctx, struct eap_config *conf)

  *Allocate and initialize EAP state machine.*

- void eap_sm_deinit (struct eap_sm *sm)
  *Deinitialize and free an EAP state machine.*
- int eap_sm_step (struct eap_sm *sm)
  *Step EAP state machine.*
- void eap_sm_abort (struct eap_sm *sm)
  *Abort EAP authentication.*
- int eap_sm_get_status (struct eap_sm *sm, char *buf, size_t buflen, int verbose)
  *Get EAP state machine status.*
- u8 * eap_sm_buildIdentity (struct eap_sm *sm, int id, size_t *len, int encrypted)
  *Build EAP-Identity/Response for the current network.*
- void eap_sm_request_identity (struct eap_sm *sm)
  *Request identity from user (ctrl_iface).*
- void eap_sm_request_password (struct eap_sm *sm)
  *Request password from user (ctrl_iface).*
- void eap_sm_request_new_password (struct eap_sm *sm)
  *Request new password from user (ctrl_iface).*
- void eap_sm_request_pin (struct eap_sm *sm)
  *Request SIM or smart card PIN from user (ctrl_iface).*
- void eap_sm_request_otp (struct eap_sm *sm, const char *msg, size_t msg_len)
  *Request one time password from user (ctrl_iface).*
- void eap_sm_request_passphrase (struct eap_sm *sm)
  *Request passphrase from user (ctrl_iface).*
- void eap_sm_notify_ctrl_attached (struct eap_sm *sm)
  *Notification of attached monitor.*
- u32 eap_get_phase2_type (const char *name, int *vendor)
  *Get EAP type for the given EAP phase 2 method name.*
- eap_method_type * eap_get_phase2_types (struct wpa_ssid *config, size_t *count)
  *Get list of allowed EAP phase 2 types.*
- void eap_set_fast_reauth (struct eap_sm *sm, int enabled)
  *Update fast_reauth setting.*
- void eap_set_workaround (struct eap_sm *sm, unsigned int workaround)
  *Update EAP workarounds setting.*
- void eap_set_force_disabled (struct eap_sm *sm, int disabled)
  *Set force_disabled flag.*

- int eap_key_available (struct eap_sm *sm)
  *Get key availability (eapKeyAvailable variable).*
- void eap_notify_success (struct eap_sm *sm)
  *Notify EAP state machine about external success trigger.*
- void eap_notify_lower_layer_success (struct eap_sm *sm)
  *Notification of lower layer success.*
- const u8 * eap_get_eapKeyData (struct eap_sm *sm, size_t *len)
  *Get master session key (MSK) from EAP state machine.*
- u8 * eap_get_eapRespData (struct eap_sm *sm, size_t *len)
  *Get EAP response data.*
- void eap_register_scard_ctx (struct eap_sm *sm, void *ctx)
  *Notification of smart card context.*
- void eap_invalidate_cached_session (struct eap_sm *sm)
  *Mark cached session data invalid.*

### 6.61.1 Detailed Description

EAP peer state machine functions (RFC 4137).

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap.h.

### 6.61.2 Enumeration Type Documentation

#### 6.61.2.1 enum eapol_bool_var

enum eapol_bool_var - EAPOL boolean state variables for EAP state machine

These variables are used in the interface between EAP peer state machine and lower layer. These are defined in RFC 4137, Sect. 4.1. Lower layer code is expected to maintain these variables and register a callback functions for EAP state machine to get and set the variables.

**Enumeration values:**
***EAPOL_eapSuccess*** EAP SUCCESS state reached.
EAP state machine reads and writes this value.

***EAPOL_eapRestart*** Lower layer request to restart authentication.
Set to TRUE in lower layer, FALSE in EAP state machine.

***EAPOL_eapFail*** EAP FAILURE state reached.
EAP state machine writes this value.

***EAPOL_eapResp*** Response to send.
Set to TRUE in EAP state machine, FALSE in lower layer.

***EAPOL_eapNoResp*** Request has been process; no response to send.
Set to TRUE in EAP state machine, FALSE in lower layer.

***EAPOL_eapReq*** EAP request available from lower layer.
Set to TRUE in lower layer, FALSE in EAP state machine.

***EAPOL_portEnabled*** Lower layer is ready for communication.
EAP state machines reads this value.

***EAPOL_altAccept*** Alternate indication of success (RFC3748).
EAP state machines reads this value.

***EAPOL_altReject*** Alternate indication of failure (RFC3748).
EAP state machines reads this value.

Definition at line 42 of file eap.h.

#### 6.61.2.2 enum eapol_int_var

enum eapol_int_var - EAPOL integer state variables for EAP state machine

These variables are used in the interface between EAP peer state machine and lower layer. These are defined in RFC 4137, Sect. 4.1. Lower layer code is expected to maintain these variables and register a callback functions for EAP state machine to get and set the variables.

**Enumeration values:**

***EAPOL_idleWhile*** Outside time for EAP peer timeout.
This integer variable is used to provide an outside timer that the external (to EAP state machine) code must decrement by one every second until the value reaches zero. This is used in the same way as EAPOL state machine timers. EAP state machine reads and writes this value.

Definition at line 124 of file eap.h.

### 6.61.3 Function Documentation

#### 6.61.3.1 const u8* eap_get_eapKeyData (struct eap_sm * *sm*, size_t * *len*)

Get master session key (MSK) from EAP state machine.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Pointer to variable that will be set to number of bytes in the key

**Returns:**

Pointer to the EAP keying data or NULL on failure

Fetch EAP keying material (MSK, eapKeyData) from the EAP state machine. The key is available only after a successful authentication. EAP state machine continues to manage the key data and the caller must not change or free the returned data.

Definition at line 1907 of file eap.c.

#### 6.61.3.2 u8* eap_get_eapRespData (struct eap_sm * *sm*, size_t * *len*)

Get EAP response data.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Pointer to variable that will be set to the length of the response

**Returns:**

Pointer to the EAP response (eapRespData) or NULL on failure

Fetch EAP response (eapRespData) from the EAP state machine. This data is available when EAP state machine has processed an incoming EAP request. The EAP state machine does not maintain a reference to the response after this function is called and the caller is responsible for freeing the data.

Definition at line 1931 of file eap.c.

#### 6.61.3.3 u32 eap_get_phase2_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP phase 2 method name.

**Parameters:**

***name*** EAP method name, e.g., MD5

***vendor*** Buffer for returning EAP Vendor-Id

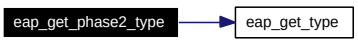
**Returns:**

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers that are allowed for Phase 2, i.e., for tunneled authentication. Phase 2 is used, e.g., with EAP-PEAP, EAP-TTLS, and EAP-FAST.

Definition at line 1648 of file eap.c.

Here is the call graph for this function:




#### 6.61.3.4 struct eap_method_type* eap_get_phase2_types (struct wpa_ssid * *config*, size_t * *count*)

Get list of allowed EAP phase 2 types.

**Parameters:**

***config*** Pointer to a network configuration

***count*** Pointer to a variable to be filled with number of returned EAP types

**Returns:**
Pointer to allocated type list or NULL on failure

This function generates an array of allowed EAP phase 2 (tunneled) types for the given network configuration.

Definition at line 1671 of file eap.c.

Here is the call graph for this function:


#### 6.61.3.5 void eap_invalidate_cached_session (struct eap_sm * *sm*)

Mark cached session data invalid.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

Definition at line 2153 of file eap.c.

#### 6.61.3.6 int eap_key_available (struct eap_sm * *sm*)

Get key availability (eapKeyAvailable variable).

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

**Returns:**
1 if EAP keying material is available, 0 if not

Definition at line 1842 of file eap.c.

#### 6.61.3.7 void eap_notify_lower_layer_success (struct eap_sm * *sm*)

Notification of lower layer success.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

Notify EAP state machines that a lower layer has detected a successful authentication. This is used to recover from dropped EAP-Success messages.

Definition at line 1875 of file eap.c.

#### 6.61.3.8 void eap_notify_success (struct eap_sm * *sm*)

Notify EAP state machine about external success trigger.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

This function is called when external event, e.g., successful completion of WPA-PSK key handshake, is indicating that EAP state machine should move to success state. This is mainly used with security modes that do not use EAP state machine (e.g., WPA-PSK).

Definition at line 1858 of file eap.c.

#### 6.61.3.9 void eap_register_scard_ctx (struct eap_sm * *sm*, void * *ctx*)

Notification of smart card context.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***ctx*** Context data for smart card operations

Notify EAP state machines of context data for smart card operations. This context data will be used as a parameter for scard_*() functions.

Definition at line 1958 of file eap.c.

#### 6.61.3.10 void eap_set_fast_reauth (struct eap_sm * *sm*, int *enabled*)

Update fast_reauth setting.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***enabled*** 1 = Fast reauthentication is enabled, 0 = Disabled

Definition at line 1712 of file eap.c.

#### 6.61.3.11 void eap_set_force_disabled (struct eap_sm * *sm*, int *disabled*)

Set force_disabled flag.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***disabled*** 1 = EAP disabled, 0 = EAP enabled

This function is used to force EAP state machine to be disabled when it is not in use (e.g., with WPA-PSK or plaintext connections).

Definition at line 2075 of file eap.c.

#### 6.61.3.12 void eap_set_workaround (struct eap_sm * *sm*, unsigned int *workaround*)

Update EAP workarounds setting.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***workaround*** 1 = Enable EAP workarounds, 0 = Disable EAP workarounds

Definition at line 1724 of file eap.c.

#### 6.61.3.13 void eap_sm_abort (struct eap_sm * *sm*)

Abort EAP authentication.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

Release system resources that have been allocated for the authentication session without fully deinitializing the EAP state machine.

Definition at line 1235 of file eap.c.

#### 6.61.3.14 u8* eap_sm_buildIdentity (struct eap_sm * *sm*, int *id*, size_t * *len*, int *encrypted*)

Build EAP-Identity/Response for the current network.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***id*** EAP identifier for the packet

***len*** Pointer to a variable that will be set to the length of the response

***encrypted*** Whether the packet is for encrypted tunnel (EAP phase 2)

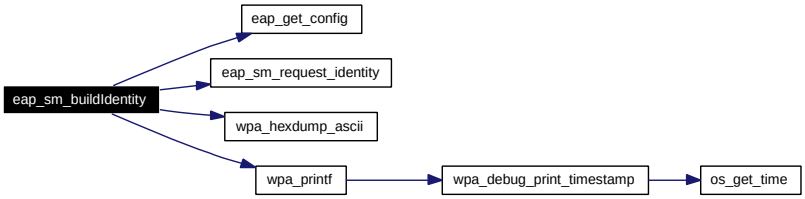
**Returns:**

Pointer to the allocated EAP-Identity/Response packet or NULL on failure

This function allocates and builds an EAP-Identity/Response packet for the current network. The caller is responsible for freeing the returned data.

Definition at line 932 of file eap.c.

Here is the call graph for this function:



#### 6.61.3.15 void eap_sm_deinit (struct eap_sm * *sm*)

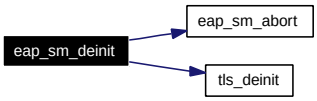
Deinitialize and free an EAP state machine.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

This function deinitializes EAP state machine and frees all allocated resources.

Definition at line 1193 of file eap.c.

Here is the call graph for this function:




#### 6.61.3.16 int eap_sm_get_status (struct eap_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get EAP state machine status.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

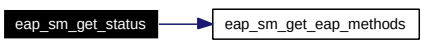
***verbose*** Whether to include verbose status information

**Returns:**
Number of bytes written to buf.

Query EAP state machine for status information. This function fills in a text area with current status information from the EAPOL state machine. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

Definition at line 1340 of file eap.c.

Here is the call graph for this function:




#### 6.61.3.17 struct eap_sm* eap_sm_init (void * *eapol_ctx*, struct eapol_callbacks * *eapol_cb*, void * *msg_ctx*, struct eap_config * *conf*)

Allocate and initialize EAP state machine.

**Parameters:**
***eapol_ctx*** Context data to be used with eapol_cb calls

***eapol_cb*** Pointer to EAPOL callback functions

***msg_ctx*** Context data for wpa_msg() calls

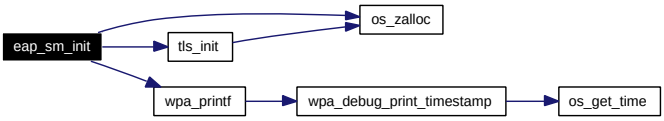
***conf*** EAP configuration

**Returns:**
Pointer to the allocated EAP state machine or NULL on failure

This function allocates and initializes an EAP state machine. In addition, this initializes TLS library for the new EAP state machine. eapol_cb pointer will be in use until eap_sm_deinit() is used to deinitialize this EAP state machine. Consequently, the caller must make sure that this data structure remains alive while the EAP state machine is active.

Definition at line 1155 of file eap.c.

Here is the call graph for this function:




#### 6.61.3.18 void eap_sm_notify_ctrl_attached (struct eap_sm * *sm*)

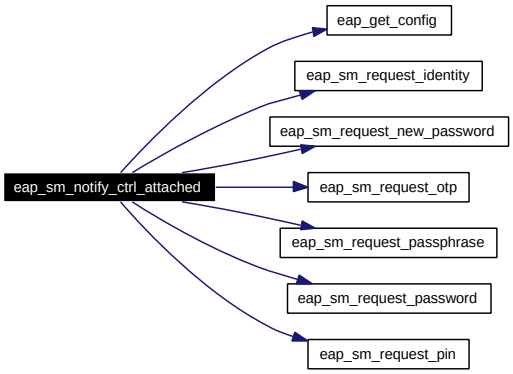
Notification of attached monitor.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

Notify EAP state machines that a monitor was attached to the control interface to trigger re-sending of pending requests for user input.

Definition at line 1602 of file eap.c.

Here is the call graph for this function:




#### 6.61.3.19 void eap_sm_request_identity (struct eap_sm * *sm*)

Request identity from user (ctrl_iface).

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request identity information for the current network. This is normally called when the identity is not included in the network configuration. The request will be sent to monitor programs through the control interface.

Definition at line 1507 of file eap.c.

#### 6.61.3.20 void eap_sm_request_new_password (struct eap_sm * *sm*)

Request new password from user (ctrl_iface).

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request new password information for the current network. This is normally called when the EAP method indicates that the current password has expired and password change is required. The request will be sent to monitor programs through the control interface.

Definition at line 1539 of file eap.c.

#### 6.61.3.21 void eap_sm_request_otp (struct eap_sm * *sm*, const char * *msg*, size_t *msg_len*)

Request one time password from user (ctrl_iface).

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***msg*** Message to be displayed to the user when asking for OTP

***msg_len*** Length of the user displayable message

EAP methods can call this function to request open time password (OTP) for the current network. The request will be sent to monitor programs through the control interface.

Definition at line 1572 of file eap.c.

#### 6.61.3.22 void eap_sm_request_passphrase (struct eap_sm * *sm*)

Request passphrase from user (ctrl_iface).

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request passphrase for a private key for the current network. This is normally called when the passphrase is not included in the network configuration. The request will be sent to monitor programs through the control interface.

Definition at line 1588 of file eap.c.

#### 6.61.3.23 void eap_sm_request_password (struct eap_sm * *sm*)

Request password from user (ctrl_iface).

**Parameters:**
*sm* Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request password information for the current network. This is normally called when the password is not included in the network configuration. The request will be sent to monitor programs through the control interface.

Definition at line 1523 of file eap.c.

#### 6.61.3.24 void eap_sm_request_pin (struct eap_sm * *sm*)

Request SIM or smart card PIN from user (ctrl_iface).

**Parameters:**
*sm* Pointer to EAP state machine allocated with eap_sm_init()

EAP methods can call this function to request SIM or smart card PIN information for the current network. This is normally called when the PIN is not included in the network configuration. The request will be sent to monitor programs through the control interface.

Definition at line 1555 of file eap.c.

#### 6.61.3.25 int eap_sm_step (struct eap_sm * *sm*)

Step EAP state machine.

**Parameters:**
*sm* Pointer to EAP state machine allocated with eap_sm_init()

**Returns:**
1 if EAP state was changed or 0 if not

This function advances EAP state machine to a new state to match with the current variables. This should be called whenever variables used by the EAP state machine have changed.

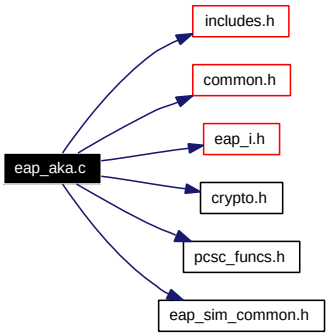
Definition at line 1214 of file eap.c.

## 6.62 eap_aka.c File Reference

EAP peer method: EAP-AKA (RFC 4187).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "crypto.h"
#include "pcsc_funcs.h"
#include "eap_sim_common.h"
```

Include dependency graph for eap_aka.c:




### Defines

- #define **CLEAR_PSEUDONYM** 0x01
- #define **CLEAR_REAUTH_ID** 0x02
- #define **CLEAR_EAP_ID** 0x04

### Functions

- int **eap_peer_aka_register** (void)

### 6.62.1 Detailed Description

EAP peer method: EAP-AKA (RFC 4187).

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

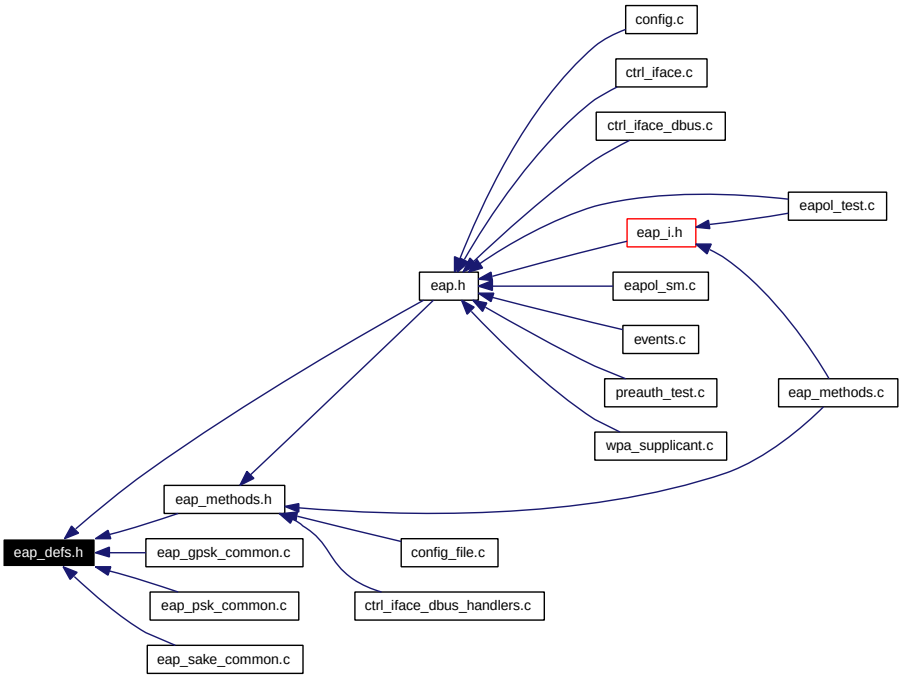
See README and COPYING for more details.

Definition in file eap_aka.c.

# 6.63 eap_defs.h File Reference

EAP server/peer: Shared EAP definitions.

This graph shows which files directly or indirectly include this file:




## Defines

- #define **EAP_MSK_LEN** 64
- #define **EAP_EMSK_LEN** 64

## Enumerations

- enum { **EAP_CODE_REQUEST** = 1, **EAP_CODE_RESPONSE** = 2, **EAP_CODE_SUCCESS** = 3, **EAP_CODE_FAILURE** = 4 }
- enum **EapType** {

  **EAP_TYPE_NONE** = 0, **EAP_TYPE_IDENTITY** = 1, **EAP_TYPE_NOTIFICATION** = 2, **EAP_TYPE_NAK** = 3,

  **EAP_TYPE_MD5** = 4, **EAP_TYPE_OTP** = 5, **EAP_TYPE_GTC** = 6, **EAP_TYPE_TLS** = 13,

  **EAP_TYPE_LEAP** = 17, **EAP_TYPE_SIM** = 18, **EAP_TYPE_TTLS** = 21, **EAP_TYPE_AKA** = 23,

  **EAP_TYPE_PEAP** = 25, **EAP_TYPE_MSCHAPV2** = 26, **EAP_TYPE_TLV** = 33, **EAP_TYPE_FAST** = 43,

  **EAP_TYPE_PAX** = 46, **EAP_TYPE_PSK** = 47, **EAP_TYPE_SAKE** = 48, **EAP_TYPE_EXPANDED** = 254,

  **EAP_TYPE_GPSK** = 255 }
- enum { **EAP_VENDOR_IETF** = 0 }

## Variables

- eap_hdr **STRUCT_PACKED**

### 6.63.1 Detailed Description

EAP server/peer: Shared EAP definitions.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

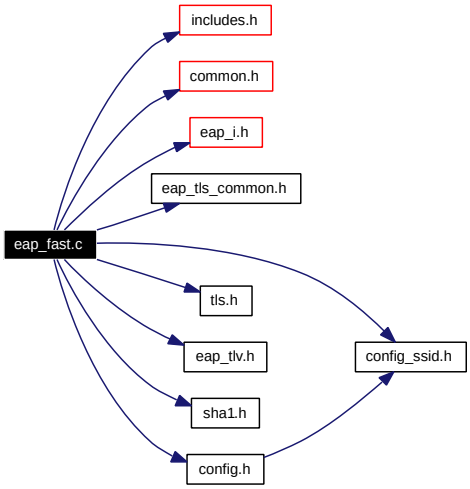
Definition in file eap_defs.h.

## 6.64 eap_fast.c File Reference

EAP peer method: EAP-FAST (draft-cam-winget-eap-fast-03.txt).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "config_ssid.h"
#include "tls.h"
#include "eap_tlv.h"
#include "sha1.h"
#include "config.h"
```

Include dependency graph for eap_fast.c:




### Defines

- #define **EAP_FAST_VERSION** 1
- #define **EAP_FAST_KEY_LEN** 64
- #define **EAP_FAST_PAC_KEY_LEN** 32
- #define **EAP_FAST_SIMCK_LEN** 40
- #define **EAP_FAST_SKS_LEN** 40
- #define **TLS_EXT_PAC_OPAQUE** 35
- #define **PAC_TYPE_PAC_KEY** 1
- #define **PAC_TYPE_PAC_OPAQUE** 2
- #define **PAC_TYPE_CRED_LIFETIME** 3
- #define **PAC_TYPE_A_ID** 4
- #define **PAC_TYPE_I_ID** 5
- #define **PAC_TYPE_SERVER_PROTECTED_DATA** 6
- #define **PAC_TYPE_A_ID_INFO** 7

- #define **PAC_TYPE_PAC_ACKNOWLEDGEMENT** 8
- #define **PAC_TYPE_PAC_INFO** 9

## Functions

- int **eap_peer_fast_register** (void)

### 6.64.1 Detailed Description

EAP peer method: EAP-FAST (draft-cam-winget-eap-fast-03.txt).

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

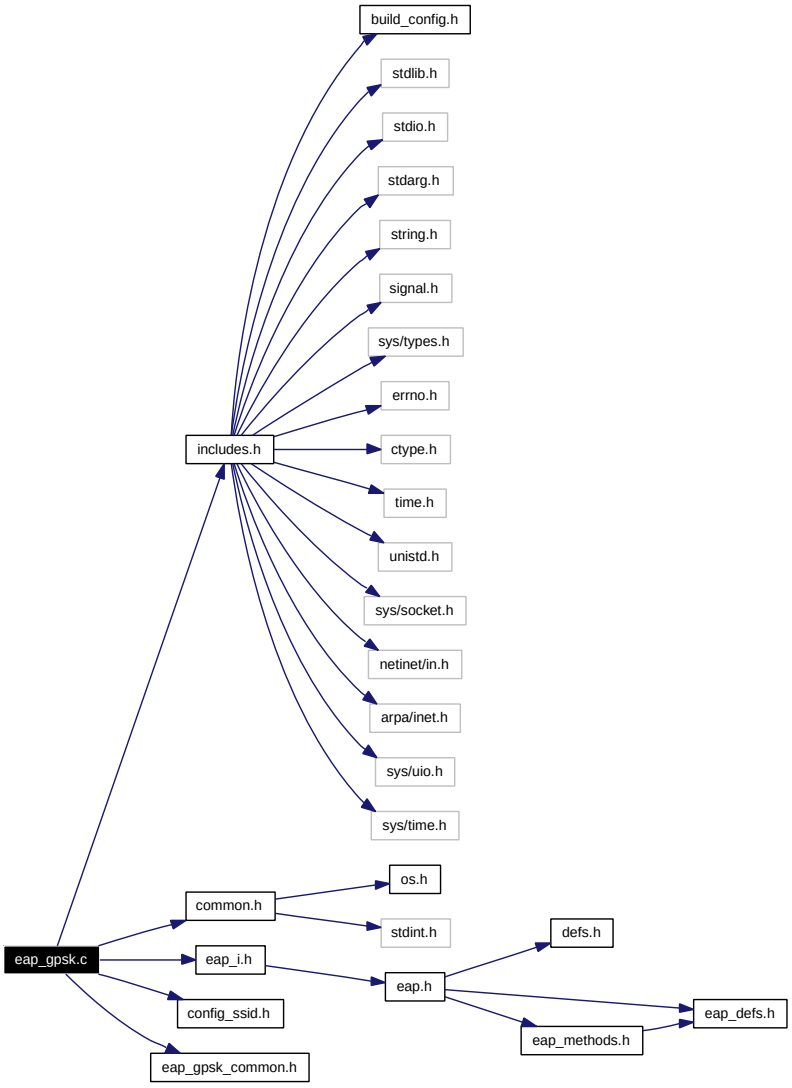
Definition in file eap_fast.c.

# 6.65 eap_gpsk.c File Reference

EAP peer method: EAP-GPSK (draft-ietf-emu-eap-gpsk-01.txt).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "config_ssid.h"
#include "eap_gpsk_common.h"
```

Include dependency graph for eap_gpsk.c:




## Functions

- int **eap_peer_gpsk_register** (void)

### 6.65.1 Detailed Description

EAP peer method: EAP-GPSK (draft-ietf-emu-eap-gpsk-01.txt).

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

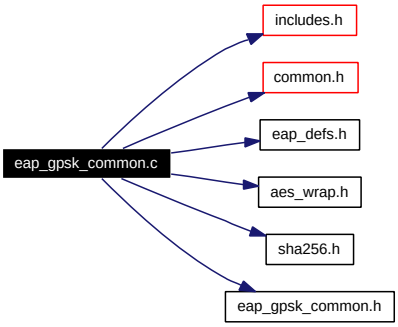
Definition in file eap_gpsk.c.

# 6.66 eap_gpsk_common.c File Reference

EAP server/peer: EAP-GPSK shared routines.

```
#include "includes.h"
#include "common.h"
#include "eap_defs.h"
#include "aes_wrap.h"
#include "sha256.h"
#include "eap_gpsk_common.h"
```

Include dependency graph for eap_gpsk_common.c:




## Defines

- #define **EAP_GPSK_SK_LEN_AES** 16
- #define **EAP_GPSK_PK_LEN_AES** 16

## Functions

- int eap_gpsk_supported_ciphersuite (int vendor, int specifier)

  *Check whether ciphersuite is supported.*

- int eap_gpsk_derive_keys (const u8 *psk, size_t psk_len, int vendor, int specifier, const u8 *rand_client, const u8 *rand_server, const u8 *id_client, size_t id_client_len, const u8 *id_server, size_t id_server_len, u8 *msk, u8 *emsk, u8 *sk, size_t *sk_len, u8 *pk, size_t *pk_len)

  *Derive EAP-GPSK keys.*

- size_t eap_gpsk_mic_len (int vendor, int specifier)

  *Get the length of the MIC.*

- int eap_gpsk_compute_mic (const u8 *sk, size_t sk_len, int vendor, int specifier, const u8 *data, size_t len, u8 *mic)

  *Compute EAP-GPSK MIC for an EAP packet.*

### 6.66.1 Detailed Description

EAP server/peer: EAP-GPSK shared routines.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_gpsk_common.c.

### 6.66.2 Function Documentation

#### 6.66.2.1 int eap_gpsk_compute_mic (const u8 * *sk*, size_t *sk_len*, int *vendor*, int *specifier*, const u8 * *data*, size_t *len*, u8 * *mic*)

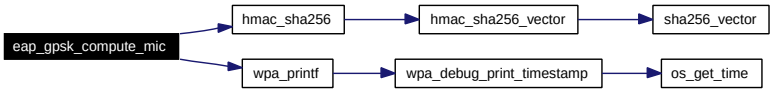
Compute EAP-GPSK MIC for an EAP packet.

**Parameters:**
***sk*** Session key SK from eap_gpsk_derive_keys()
***sk_len*** SK length in bytes from eap_gpsk_derive_keys()
***vendor*** CSuite/Vendor
***specifier*** CSuite/Specifier
***data*** Input data to MIC
***len*** Input data length in bytes
***mic*** Buffer for the computed MIC, eap_gpsk_mic_len(cipher) bytes

**Returns:**
0 on success, -1 on failure

Definition at line 434 of file eap_gpsk_common.c.

Here is the call graph for this function:




#### 6.66.2.2 int eap_gpsk_derive_keys (const u8 * *psk*, size_t *psk_len*, int *vendor*, int *specifier*, const u8 * *rand_client*, const u8 * *rand_server*, const u8 * *id_client*, size_t *id_client_len*, const u8 * *id_server*, size_t *id_server_len*, u8 * *msk*, u8 * *emsk*, u8 * *sk*, size_t * *sk_len*, u8 * *pk*, size_t * *pk_len*)

Derive EAP-GPSK keys.

**Parameters:**

***psk*** Pre-shared key (at least 16 bytes if AES is used)

***psk_len*** Length of psk in bytes

***vendor*** CSuite/Vendor

***specifier*** CSuite/Specifier

***rand_client*** 32-byte RAND_Client

***rand_server*** 32-byte RAND_Server

***id_client*** ID_Client

***id_client_len*** Length of ID_Client

***id_server*** ID_Server

***id_server_len*** Length of ID_Server

***msk*** Buffer for 64-byte MSK

***emsk*** Buffer for 64-byte EMSK

***sk*** Buffer for SK (at least EAP_GPSK_MAX_SK_LEN bytes)

***sk_len*** Buffer for returning length of SK

***pk*** Buffer for SK (at least EAP_GPSK_MAX_PK_LEN bytes)

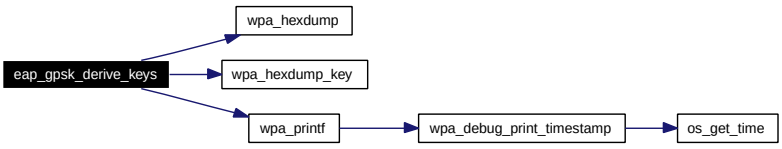
***pk_len*** Buffer for returning length of PK

**Returns:**

0 on success, -1 on failure

Definition at line 318 of file eap_gpsk_common.c.

Here is the call graph for this function:




#### 6.66.2.3 size_t eap_gpsk_mic_len (int *vendor*, int *specifier*)

Get the length of the MIC.

**Parameters:**

***vendor*** CSuite/Vendor

***specifier*** CSuite/Specifier

**Returns:**

MIC length in bytes

Definition at line 391 of file eap_gpsk_common.c.

#### 6.66.2.4 int eap_gpsk_supported_ciphersuite (int *vendor*, int *specifier*)

Check whether ciphersuite is supported.

**Parameters:**

***vendor*** CSuite/Vendor

***specifier*** CSuite/Specifier

**Returns:**

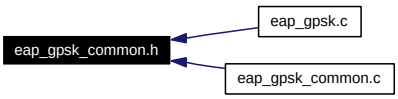
1 if ciphersuite is support, or 0 if not

Definition at line 32 of file eap_gpsk_common.c.

# 6.67 eap_gpsk_common.h File Reference

EAP server/peer: EAP-GPSK shared routines.

This graph shows which files directly or indirectly include this file:




## Defines

- #define **EAP_GPSK_OPCODE_GPSK_1** 1
- #define **EAP_GPSK_OPCODE_GPSK_2** 2
- #define **EAP_GPSK_OPCODE_GPSK_3** 3
- #define **EAP_GPSK_OPCODE_GPSK_4** 4
- #define **EAP_GPSK_RAND_LEN** 32
- #define **EAP_GPSK_MAX_SK_LEN** 32
- #define **EAP_GPSK_MAX_PK_LEN** 32
- #define **EAP_GPSK_MAX_MIC_LEN** 32
- #define **EAP_GPSK_VENDOR_IETF** 0x000000
- #define **EAP_GPSK_CIPHER_RESERVED** 0x000000
- #define **EAP_GPSK_CIPHER_AES** 0x000001
- #define **EAP_GPSK_CIPHER_SHA256** 0x000002

## Functions

- int eap_gpsk_supported_ciphersuite (int vendor, int specifier)

  *Check whether ciphersuite is supported.*

- int eap_gpsk_derive_keys (const u8 *psk, size_t psk_len, int vendor, int specifier, const u8 *rand_client, const u8 *rand_server, const u8 *id_client, size_t id_client_len, const u8 *id_server, size_t id_server_len, u8 *msk, u8 *emsk, u8 *sk, size_t *sk_len, u8 *pk, size_t *pk_len)

  *Derive EAP-GPSK keys.*

- size_t eap_gpsk_mic_len (int vendor, int specifier)

  *Get the length of the MIC.*

- int eap_gpsk_compute_mic (const u8 *sk, size_t sk_len, int vendor, int specifier, const u8 *data, size_t len, u8 *mic)

  *Compute EAP-GPSK MIC for an EAP packet.*

## Variables

- eap_gpsk_csuite **STRUCT_PACKED**

### 6.67.1 Detailed Description

EAP server/peer: EAP-GPSK shared routines.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_gpsk_common.h.

### 6.67.2 Function Documentation

#### 6.67.2.1 int eap_gpsk_compute_mic (const u8 * *sk*, size_t *sk_len*, int *vendor*, int *specifier*, const u8 * *data*, size_t *len*, u8 * *mic*)

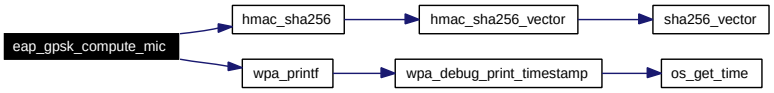
Compute EAP-GPSK MIC for an EAP packet.

**Parameters:**
***sk*** Session key SK from eap_gpsk_derive_keys()
***sk_len*** SK length in bytes from eap_gpsk_derive_keys()
***vendor*** CSuite/Vendor
***specifier*** CSuite/Specifier
***data*** Input data to MIC
***len*** Input data length in bytes
***mic*** Buffer for the computed MIC, eap_gpsk_mic_len(cipher) bytes

**Returns:**
0 on success, -1 on failure

Definition at line 434 of file eap_gpsk_common.c.

Here is the call graph for this function:




#### 6.67.2.2 int eap_gpsk_derive_keys (const u8 * *psk*, size_t *psk_len*, int *vendor*, int *specifier*, const u8 * *rand_client*, const u8 * *rand_server*, const u8 * *id_client*, size_t *id_client_len*, const u8 * *id_server*, size_t *id_server_len*, u8 * *msk*, u8 * *emsk*, u8 * *sk*, size_t * *sk_len*, u8 * *pk*, size_t * *pk_len*)

Derive EAP-GPSK keys.

**Parameters:**

***psk*** Pre-shared key (at least 16 bytes if AES is used)

***psk_len*** Length of psk in bytes

***vendor*** CSuite/Vendor

***specifier*** CSuite/Specifier

***rand_client*** 32-byte RAND_Client

***rand_server*** 32-byte RAND_Server

***id_client*** ID_Client

***id_client_len*** Length of ID_Client

***id_server*** ID_Server

***id_server_len*** Length of ID_Server

***msk*** Buffer for 64-byte MSK

***emsk*** Buffer for 64-byte EMSK

***sk*** Buffer for SK (at least EAP_GPSK_MAX_SK_LEN bytes)

***sk_len*** Buffer for returning length of SK

***pk*** Buffer for SK (at least EAP_GPSK_MAX_PK_LEN bytes)

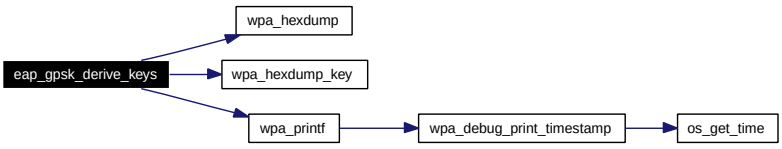
***pk_len*** Buffer for returning length of PK

**Returns:**

0 on success, -1 on failure

Definition at line 318 of file eap_gpsk_common.c.

Here is the call graph for this function:




#### 6.67.2.3 size_t eap_gpsk_mic_len (int *vendor*, int *specifier*)

Get the length of the MIC.

**Parameters:**

***vendor*** CSuite/Vendor

***specifier*** CSuite/Specifier

**Returns:**

MIC length in bytes

Definition at line 391 of file eap_gpsk_common.c.

#### 6.67.2.4 int eap_gpsk_supported_ciphersuite (int *vendor*, int *specifier*)

Check whether ciphersuite is supported.

**Parameters:**

***vendor*** CSuite/Vendor

***specifier*** CSuite/Specifier

**Returns:**

1 if ciphersuite is support, or 0 if not

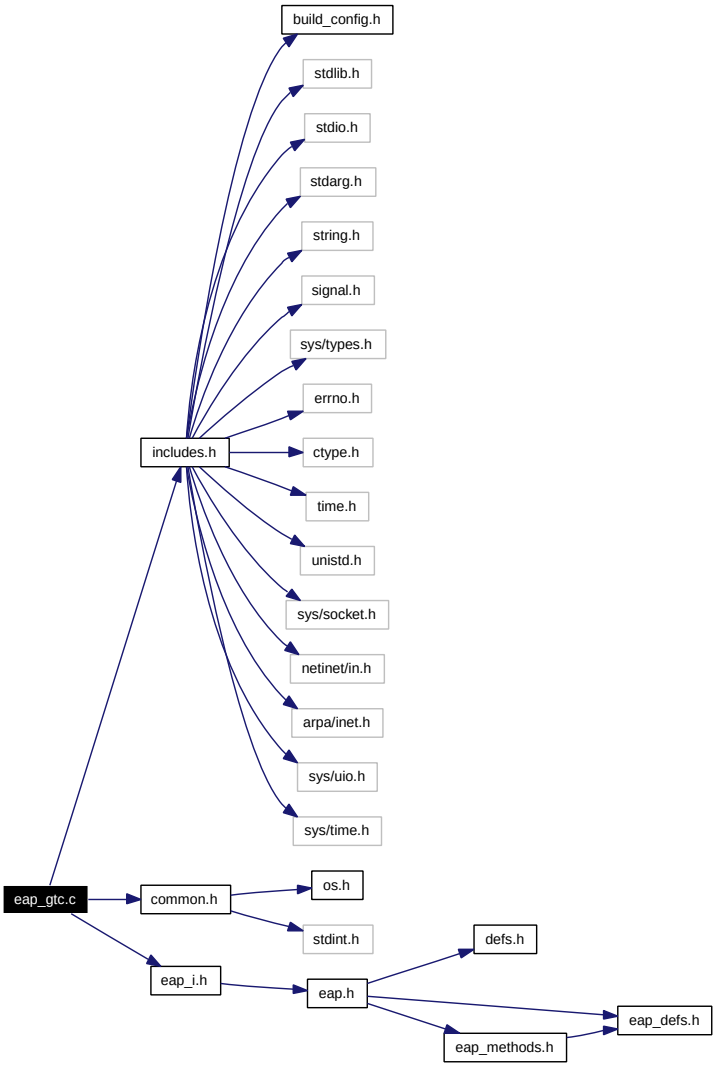
Definition at line 32 of file eap_gpsk_common.c.

# 6.68 eap_gtc.c File Reference

EAP peer method: EAP-GTC (RFC 3748).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
```

Include dependency graph for eap_gtc.c:




## Functions

- int **eap_peer_gtc_register** (void)

### 6.68.1 Detailed Description

EAP peer method: EAP-GTC (RFC 3748).

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>



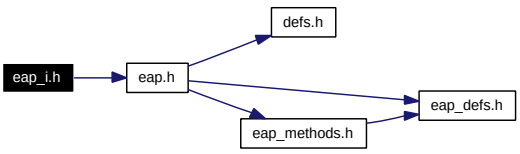
Definition in file eap_gtc.c.

# 6.69 eap_i.h File Reference

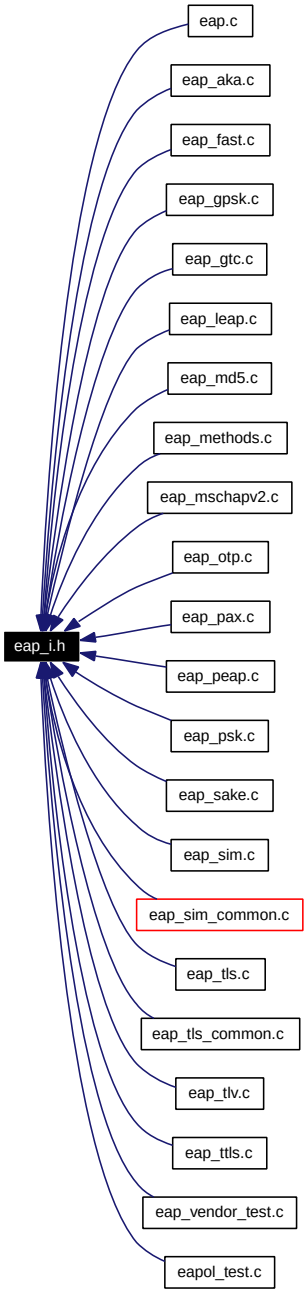
EAP peer state machines internal structures (RFC 4137).

`#include "eap.h"`

Include dependency graph for eap_i.h:




This graph shows which files directly or indirectly include this file:




## Defines

- #define **EAP_PEER_METHOD_INTERFACE_VERSION** 1

## Enumerations

- enum **EapDecision** { **DECISION_FAIL**, **DECISION_COND_SUCC**, **DECISION_UNCOND_-SUCC** }
- enum **EapMethodState** {

**METHOD_NONE**, **METHOD_INIT**, **METHOD_CONT**, **METHOD_MAY_CONT**, **METHOD_DONE** }

## Functions

- const u8 * eap_hdr_validate (int vendor, EapType eap_type, const u8 *msg, size_t msglen, size_t *plen)

  *Validate EAP header.*

- const u8 * eap_get_config_identity (struct eap_sm *sm, size_t *len)

  *Get identity from the network configuration.*

- const u8 * eap_get_config_password (struct eap_sm *sm, size_t *len)

  *Get password from the network configuration.*

- const u8 * eap_get_config_new_password (struct eap_sm *sm, size_t *len)

  *Get new password from network configuration.*

- const u8 * eap_get_config_otp (struct eap_sm *sm, size_t *len)

  *Get one-time password from the network configuration.*

- void eap_clear_config_otp (struct eap_sm *sm)

  *Clear used one-time password.*

- wpa_ssid * eap_get_config (struct eap_sm *sm)

  *Get current network configuration.*

- void eap_set_config_blob (struct eap_sm *sm, struct wpa_config_blob *blob)

  *Set or add a named configuration blob.*

- const struct wpa_config_blob * eap_get_config_blob (struct eap_sm *sm, const char *name)

  *Get a named configuration blob.*

- eap_hdr * eap_msg_alloc (int vendor, EapType type, size_t *len, size_t payload_len, u8 code, u8 identifier, u8 **payload)

  *Allocate a buffer for an EAP message.*

- void eap_notify_pending (struct eap_sm *sm)

  *Notify that EAP method is ready to re-process a request.*

### 6.69.1 Detailed Description

EAP peer state machines internal structures (RFC 4137).

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_i.h.

## 6.69.2 Function Documentation

### 6.69.2.1 void eap_clear_config_otp (struct eap_sm * *sm*)

Clear used one-time password.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

This function clears a used one-time password (OTP) from the current network configuration. This should be called when the OTP has been used and is not needed anymore.

Definition at line 1824 of file eap.c.

Here is the call graph for this function:


### 6.69.2.2 struct wpa_ssid* eap_get_config (struct eap_sm * *sm*)

Get current network configuration.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

**Returns:**

Pointer to the current network configuration or NULL if not found

EAP peer methods should avoid using this function if they can use other access functions, like eap_get_config_identity() and eap_get_config_password(), that do not require direct access to struct wpa_ssid.

Definition at line 1741 of file eap.c.

### 6.69.2.3 const struct wpa_config_blob* eap_get_config_blob (struct eap_sm * *sm*, const char * *name*)

Get a named configuration blob.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***name*** Name of the blob

**Returns:**
Pointer to blob data or NULL if not found

Definition at line 2059 of file eap.c.

#### 6.69.2.4 const u8* eap_get_config_identity (struct eap_sm * *sm*, size_t * *len*)

Get identity from the network configuration.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()
***len*** Buffer for the length of the identity

**Returns:**
Pointer to the identity or NULL if not found

Definition at line 1754 of file eap.c.

Here is the call graph for this function:


#### 6.69.2.5 const u8* eap_get_config_new_password (struct eap_sm * *sm*, size_t * *len*)

Get new password from network configuration.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()
***len*** Buffer for the length of the new password

**Returns:**
Pointer to the new password or NULL if not found

Definition at line 1788 of file eap.c.

Here is the call graph for this function:


#### 6.69.2.6 const u8* eap_get_config_otp (struct eap_sm * *sm*, size_t * *len*)

Get one-time password from the network configuration.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()
***len*** Buffer for the length of the one-time password

**Returns:**
Pointer to the one-time password or NULL if not found

Definition at line 1805 of file eap.c.

Here is the call graph for this function:


#### 6.69.2.7 const u8* eap_get_config_password (struct eap_sm * *sm*, size_t * *len*)

Get password from the network configuration.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***len*** Buffer for the length of the password

**Returns:**
Pointer to the password or NULL if not found

Definition at line 1771 of file eap.c.

Here is the call graph for this function:


#### 6.69.2.8 const u8* eap_hdr_validate (int *vendor*, EapType *eap_type*, const u8 * *msg*, size_t *msglen*, size_t * *plen*)

Validate EAP header.

**Parameters:**
***vendor*** Expected EAP Vendor-Id (0 = IETF)

***eap_type*** Expected EAP type number

***msg*** EAP frame (starting with EAP header)

***msglen*** Length of msg

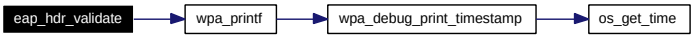
***plen*** Pointer to variable to contain the returned payload length

**Returns:**
Pointer to EAP payload (after type field), or NULL on failure

This is a helper function for EAP method implementations. This is usually called in the beginning of struct eap_method::process() function to verify that the received EAP request packet has a valid header. This function is able to process both legacy and expanded EAP headers and in most cases, the caller can just use the returned payload pointer (into *plen) for processing the payload regardless of whether the packet used the expanded EAP header or not.

Definition at line 1983 of file eap.c.

Here is the call graph for this function:




#### 6.69.2.9 struct eap_hdr* eap_msg_alloc (int *vendor*, EapType *type*, size_t * *len*, size_t *payload_len*, u8 *code*, u8 *identifier*, u8 ** *payload*)

Allocate a buffer for an EAP message.

**Parameters:**

***vendor*** Vendor-Id (0 = IETF)

***type*** EAP type

***len*** Buffer for returning message length

***payload_len*** Payload length in bytes (data after Type)

***code*** Message Code (EAP_CODE_*)

***identifier*** Identifier

***payload*** Pointer to payload pointer that will be set to point to the beginning of the payload or NULL if payload pointer is not needed

**Returns:**

Pointer to the allocated message buffer or NULL on error

This function can be used to allocate a buffer for an EAP message and fill in the EAP header. This function is automatically using expanded EAP header if the selected Vendor-Id is not IETF. In other words, most EAP methods do not need to separately select which header type to use when using this function to allocate the message buffers.

Definition at line 2100 of file eap.c.

#### 6.69.2.10 void eap_notify_pending (struct eap_sm * *sm*)

Notify that EAP method is ready to re-process a request.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

An EAP method can perform a pending operation (e.g., to get a response from an external process). Once the response is available, this function can be used to request EAPOL state machine to retry delivering the previously received (and still unanswered) EAP request to EAP state machine.

Definition at line 2142 of file eap.c.

#### 6.69.2.11 void eap_set_config_blob (struct eap_sm * *sm*, struct wpa_config_blob * *blob*)

Set or add a named configuration blob.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***blob*** New value for the blob

Adds a new configuration blob or replaces the current value of an existing blob.

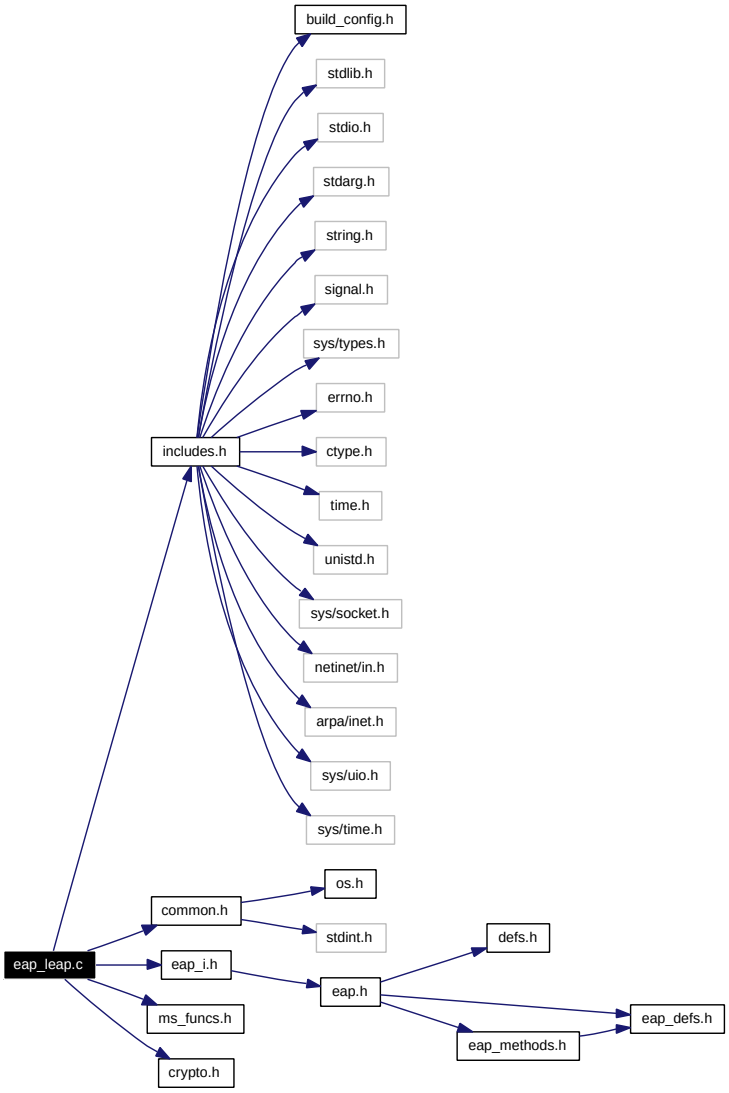
Definition at line 2046 of file eap.c.

# 6.70 eap_leap.c File Reference

EAP peer method: LEAP.

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "ms_funcs.h"
#include "crypto.h"
```

Include dependency graph for eap_leap.c:




## Defines

- #define **LEAP_VERSION** 1

- #define **LEAP_CHALLENGE_LEN** 8
- #define **LEAP_RESPONSE_LEN** 24
- #define **LEAP_KEY_LEN** 16

## Functions

- int **eap_peer_leap_register** (void)

### 6.70.1 Detailed Description

EAP peer method: LEAP.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

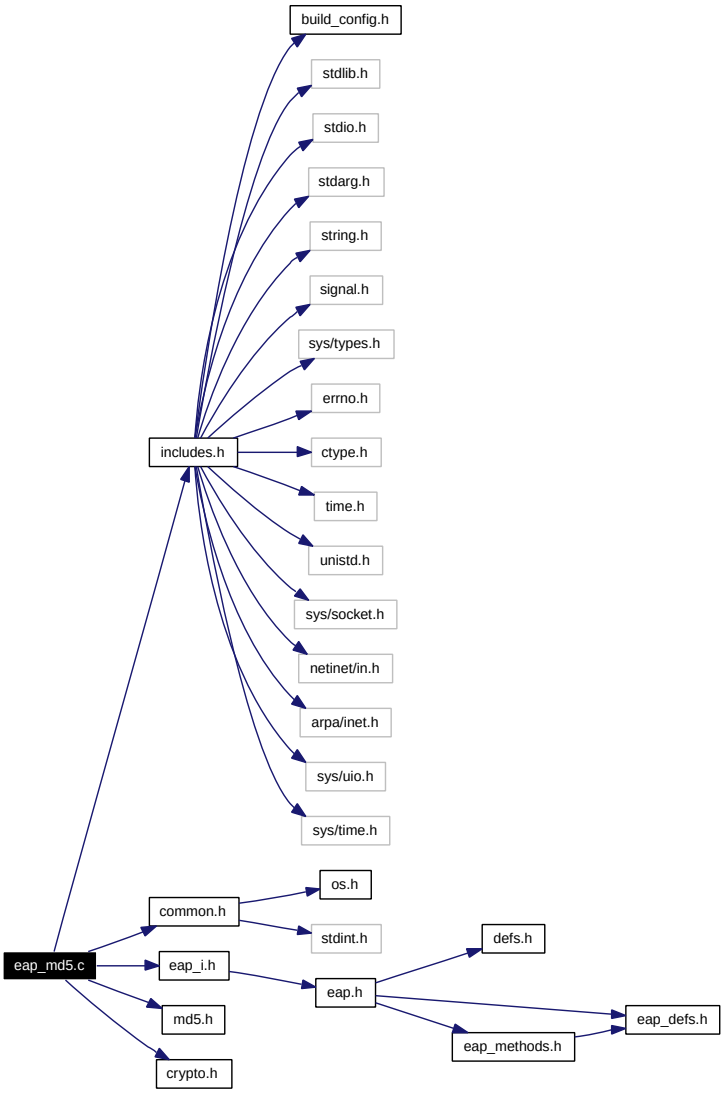
Definition in file eap_leap.c.

# 6.71 eap_md5.c File Reference

EAP peer method: EAP-MD5 (RFC 3748 and RFC 1994).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "md5.h"
#include "crypto.h"
```

Include dependency graph for eap_md5.c:




## Functions

- int **eap_peer_md5_register** (void)

### 6.71.1 Detailed Description

EAP peer method: EAP-MD5 (RFC 3748 and RFC 1994).



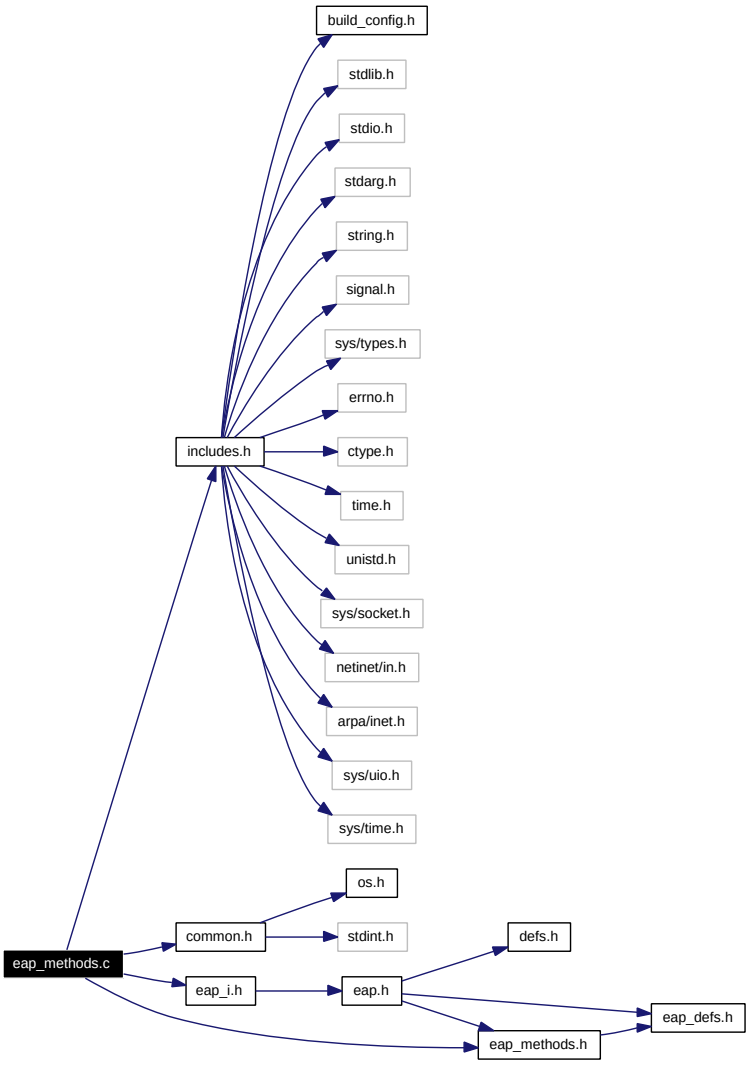
Definition in file eap_md5.c.

# 6.72 eap_methods.c File Reference

EAP peer: Method registration.

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "eap_methods.h"
```

Include dependency graph for eap_methods.c:




## Functions

- const struct eap_method * eap_sm_get_eap_methods (int vendor, EapType method)

  *Get EAP method based on type number.*

- EapType eap_get_type (const char *name, int *vendor)

*Get EAP type for the given EAP method name.*

- const char * eap_get_name (int vendor, EapType type)
  *Get EAP method name for the given EAP type.*
- size_t eap_get_names (char *buf, size_t buflen)
  *Get space separated list of names for supported EAP methods.*
- char ** eap_get_names_as_string_array (size_t *num)
  *Get supported EAP methods as string array.*
- const struct eap_method * eap_peer_get_methods (size_t *count)
  *Get a list of enabled EAP peer methods.*
- eap_method * eap_peer_method_alloc (int version, int vendor, EapType method, const char *name)
  *Allocate EAP peer method structure.*
- void eap_peer_method_free (struct eap_method *method)
  *Free EAP peer method structure.*
- int eap_peer_method_register (struct eap_method *method)
  *Register an EAP peer method.*
- int eap_peer_register_methods (void)
  *Register statically linked EAP peer methods.*
- void eap_peer_unregister_methods (void)
  *Unregister EAP peer methods.*

### 6.72.1 Detailed Description

EAP peer: Method registration.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_methods.c.

### 6.72.2 Function Documentation

#### 6.72.2.1 const char* eap_get_name (int *vendor*, EapType *type*)

Get EAP method name for the given EAP type.

**Parameters:**

***vendor*** EAP Vendor-Id (0 = IETF)

***type*** EAP method type

**Returns:**

EAP method name, e.g., TLS, or NULL if not found

This function maps EAP type numbers into EAP type names based on the list of EAP methods included in the build.

Definition at line 81 of file eap_methods.c.

#### 6.72.2.2 size_t eap_get_names (char * *buf*, size_t *buflen*)

Get space separated list of names for supported EAP methods.

**Parameters:**

***buf*** Buffer for names

***buflen*** Buffer length

**Returns:**

Number of characters written into buf (not including nul termination)

Definition at line 100 of file eap_methods.c.

#### 6.72.2.3 char** eap_get_names_as_string_array (size_t * *num*)

Get supported EAP methods as string array.

**Parameters:**

***num*** Buffer for returning the number of items in array, not including NULL terminator. This parameter can be NULL if the length is not needed.

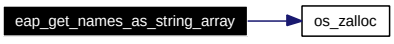
**Returns:**

A NULL-terminated array of strings, or NULL on error.

This function returns the list of names for all supported EAP methods as an array of strings. The caller must free the returned array items and the array.

Definition at line 136 of file eap_methods.c.

Here is the call graph for this function:




#### 6.72.2.4 EapType eap_get_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP method name.

**Parameters:**

***name*** EAP method name, e.g., TLS

***vendor*** Buffer for returning EAP Vendor-Id

**Returns:**

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers based on the list of EAP methods included in the build.

Definition at line 57 of file eap_methods.c.

#### 6.72.2.5 const struct eap_method* eap_peer_get_methods (size_t * *count*)

Get a list of enabled EAP peer methods.

**Parameters:**

***count*** Set to number of available methods

**Returns:**

List of enabled EAP peer methods

Definition at line 174 of file eap_methods.c.

#### 6.72.2.6 struct eap_method* eap_peer_method_alloc (int *version*, int *vendor*, EapType *method*, const char * *name*)

Allocate EAP peer method structure.

**Parameters:**

***version*** Version of the EAP peer method interface (set to EAP_PEER_METHOD_INTERFACE_VERSION)

***vendor*** EAP Vendor-ID (EAP_VENDOR_*) (0 = IETF)

***method*** EAP type number (EAP_TYPE_*)

***name*** Name of the method (e.g., "TLS")

**Returns:**

Allocated EAP method structure or NULL on failure

The returned structure should be freed with eap_peer_method_free() when it is not needed anymore.

Definition at line 293 of file eap_methods.c.

Here is the call graph for this function:

#### 6.72.2.7 void eap_peer_method_free (struct eap_method * *method*)

Free EAP peer method structure.

**Parameters:**
***method*** Method structure allocated with eap_peer_method_alloc()

Definition at line 313 of file eap_methods.c.

#### 6.72.2.8 int eap_peer_method_register (struct eap_method * *method*)

Register an EAP peer method.

**Parameters:**
***method*** EAP method to register

**Returns:**
0 on success, -1 on invalid method, or -2 if a matching EAP method has already been registered

Each EAP peer method needs to call this function to register itself as a supported EAP method.

Definition at line 329 of file eap_methods.c.

#### 6.72.2.9 int eap_peer_register_methods (void)

Register statically linked EAP peer methods.

**Returns:**
0 on success, -1 on failure

This function is called at program initialization to register all EAP peer methods that were linked in statically.

Definition at line 362 of file eap_methods.c.

Here is the call graph for this function:


#### 6.72.2.10 void eap_peer_unregister_methods (void)

Unregister EAP peer methods.

This function is called at program termination to unregister all EAP peer methods.

Definition at line 489 of file eap_methods.c.

Here is the call graph for this function:


#### 6.72.2.11 const struct eap_method* eap_sm_get_eap_methods (int *vendor*, EapType *method*)

Get EAP method based on type number.

**Parameters:**

***vendor*** EAP Vendor-Id (0 = IETF)

***method*** EAP type number

**Returns:**

Pointer to EAP method or NULL if not found

Definition at line 36 of file eap_methods.c.

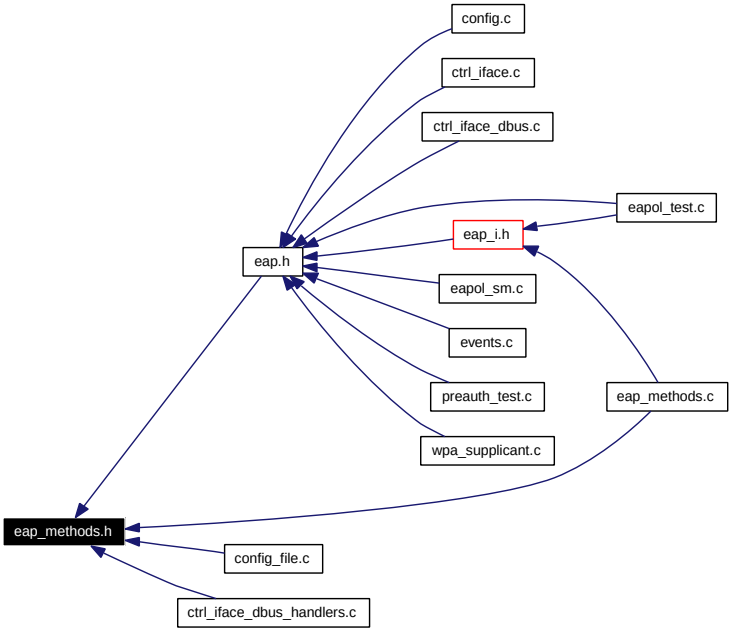
# 6.73 eap_methods.h File Reference

EAP peer: Method registration.

```
#include "eap_defs.h"
```

Include dependency graph for eap_methods.h:


This graph shows which files directly or indirectly include this file:




## Functions

- const struct eap_method * eap_sm_get_eap_methods (int vendor, EapType method)

  *Get EAP method based on type number.*

- const struct eap_method * eap_peer_get_methods (size_t *count)

  *Get a list of enabled EAP peer methods.*

- eap_method * eap_peer_method_alloc (int version, int vendor, EapType method, const char *name)

  *Allocate EAP peer method structure.*

- void eap_peer_method_free (struct eap_method *method)

  *Free EAP peer method structure.*

- int eap_peer_method_register (struct eap_method *method)

  *Register an EAP peer method.*

- EapType eap_get_type (const char *name, int *vendor)
  *Get EAP type for the given EAP method name.*
- const char * eap_get_name (int vendor, EapType type)
  *Get EAP method name for the given EAP type.*
- size_t eap_get_names (char *buf, size_t buflen)
  *Get space separated list of names for supported EAP methods.*
- char ** eap_get_names_as_string_array (size_t *num)
  *Get supported EAP methods as string array.*
- int eap_peer_register_methods (void)
  *Register statically linked EAP peer methods.*
- void eap_peer_unregister_methods (void)
  *Unregister EAP peer methods.*

### 6.73.1 Detailed Description

EAP peer: Method registration.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_methods.h.

### 6.73.2 Function Documentation

#### 6.73.2.1 const char* eap_get_name (int *vendor*, EapType *type*)

Get EAP method name for the given EAP type.

**Parameters:**
***vendor*** EAP Vendor-Id (0 = IETF)
***type*** EAP method type

**Returns:**
EAP method name, e.g., TLS, or NULL if not found

This function maps EAP type numbers into EAP type names based on the list of EAP methods included in the build.

Definition at line 81 of file eap_methods.c.

#### 6.73.2.2 size_t eap_get_names (char * *buf*, size_t *buflen*)

Get space separated list of names for supported EAP methods.

**Parameters:**

***buf*** Buffer for names

***buflen*** Buffer length

**Returns:**

Number of characters written into buf (not including nul termination)

Definition at line 100 of file eap_methods.c.

#### 6.73.2.3 char** eap_get_names_as_string_array (size_t * *num*)

Get supported EAP methods as string array.

**Parameters:**

***num*** Buffer for returning the number of items in array, not including NULL terminator. This parameter can be NULL if the length is not needed.

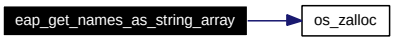
**Returns:**

A NULL-terminated array of strings, or NULL on error.

This function returns the list of names for all supported EAP methods as an array of strings. The caller must free the returned array items and the array.

Definition at line 136 of file eap_methods.c.

Here is the call graph for this function:




#### 6.73.2.4 EapType eap_get_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP method name.

**Parameters:**

***name*** EAP method name, e.g., TLS

***vendor*** Buffer for returning EAP Vendor-Id

**Returns:**

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers based on the list of EAP methods included in the build.

Definition at line 57 of file eap_methods.c.

#### 6.73.2.5 const struct eap_method* eap_peer_get_methods (size_t * *count*)

Get a list of enabled EAP peer methods.

**Parameters:**
*count* Set to number of available methods

**Returns:**
List of enabled EAP peer methods

Definition at line 174 of file eap_methods.c.

#### 6.73.2.6 struct eap_method* eap_peer_method_alloc (int *version*, int *vendor*, EapType *method*, const char * *name*)

Allocate EAP peer method structure.

**Parameters:**
*version* Version of the EAP peer method interface (set to EAP_PEER_METHOD_INTERFACE_VERSION)

*vendor* EAP Vendor-ID (EAP_VENDOR_*) (0 = IETF)

*method* EAP type number (EAP_TYPE_*)

*name* Name of the method (e.g., "TLS")

**Returns:**
Allocated EAP method structure or NULL on failure

The returned structure should be freed with eap_peer_method_free() when it is not needed anymore.

Definition at line 293 of file eap_methods.c.

Here is the call graph for this function:


#### 6.73.2.7 void eap_peer_method_free (struct eap_method * *method*)

Free EAP peer method structure.

**Parameters:**
*method* Method structure allocated with eap_peer_method_alloc()

Definition at line 313 of file eap_methods.c.

#### 6.73.2.8 int eap_peer_method_register (struct eap_method * *method*)

Register an EAP peer method.

**Parameters:**
***method*** EAP method to register

**Returns:**
0 on success, -1 on invalid method, or -2 if a matching EAP method has already been registered

Each EAP peer method needs to call this function to register itself as a supported EAP method.

Definition at line 329 of file eap_methods.c.

#### 6.73.2.9 int eap_peer_register_methods (void)

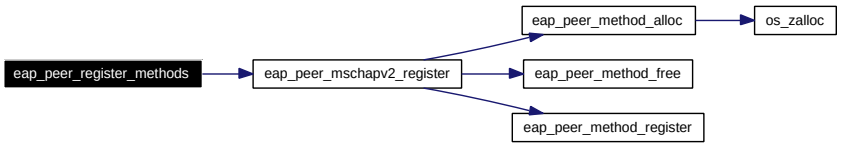
Register statically linked EAP peer methods.

**Returns:**
0 on success, -1 on failure

This function is called at program initialization to register all EAP peer methods that were linked in statically.

Definition at line 362 of file eap_methods.c.

Here is the call graph for this function:




#### 6.73.2.10 void eap_peer_unregister_methods (void)

Unregister EAP peer methods.

This function is called at program termination to unregister all EAP peer methods.

Definition at line 489 of file eap_methods.c.

Here is the call graph for this function:


#### 6.73.2.11 const struct eap_method* eap_sm_get_eap_methods (int *vendor*, EapType *method*)

Get EAP method based on type number.

**Parameters:**
***vendor*** EAP Vendor-Id (0 = IETF)

***method*** EAP type number

**Returns:**
Pointer to EAP method or NULL if not found

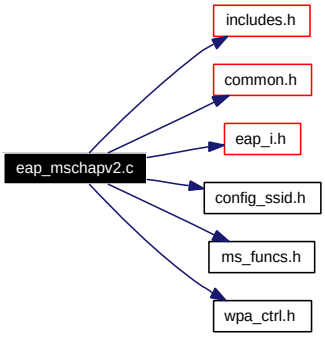
Definition at line 36 of file eap_methods.c.

# 6.74 eap_mschapv2.c File Reference

EAP peer method: EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.txt).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "config_ssid.h"
#include "ms_funcs.h"
#include "wpa_ctrl.h"
```

Include dependency graph for eap_mschapv2.c:




## Defines

- #define **MSCHAPV2_CHAL_LEN** 16
- #define **MSCHAPV2_NT_RESPONSE_LEN** 24
- #define **MSCHAPV2_OP_CHALLENGE** 1
- #define **MSCHAPV2_OP_RESPONSE** 2
- #define **MSCHAPV2_OP_SUCCESS** 3
- #define **MSCHAPV2_OP_FAILURE** 4
- #define **MSCHAPV2_OP_CHANGE_PASSWORD** 7
- #define **ERROR_RESTRICTED_LOGON_HOURS** 646
- #define **ERROR_ACCT_DISABLED** 647
- #define **ERROR_PASSWD_EXPIRED** 648
- #define **ERROR_NO_DIALIN_PERMISSION** 649
- #define **ERROR_AUTHENTICATION_FAILURE** 691
- #define **ERROR_CHANGING_PASSWORD** 709
- #define **PASSWD_CHANGE_CHAL_LEN** 16
- #define **MSCHAPV2_KEY_LEN** 16

## Functions

- int eap_peer_mschapv2_register (void)

  *Register EAP-MSCHAPv2 peer method.*

## Variables

- eap_mschapv2_hdr **STRUCT_PACKED**

### 6.74.1 Detailed Description

EAP peer method: EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.txt).

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements EAP peer part of EAP-MSCHAPV2 method (EAP type 26). draft-kamath-pppext-eap-mschapv2-00.txt defines the Microsoft EAP CHAP Extensions Protocol, Version 2, for mutual authentication and key derivation. This encapsulates MS-CHAP-v2 protocol which is defined in RFC 2759. Use of EAP-MSCHAPV2 derived keys with MPPE cipher is described in RFC 3079.

Definition in file eap_mschapv2.c.

### 6.74.2 Function Documentation

#### 6.74.2.1 int eap_peer_mschapv2_register (void)

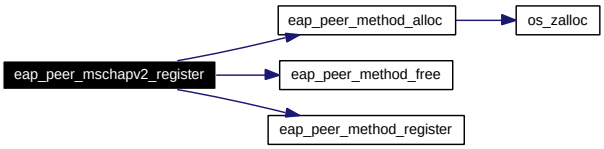
Register EAP-MSCHAPv2 peer method.

**Returns:**
0 on success, -1 on failure

This function is used to register EAP-MSCHAPv2 peer method into the EAP method list.

Definition at line 929 of file eap_mschapv2.c.

Here is the call graph for this function:

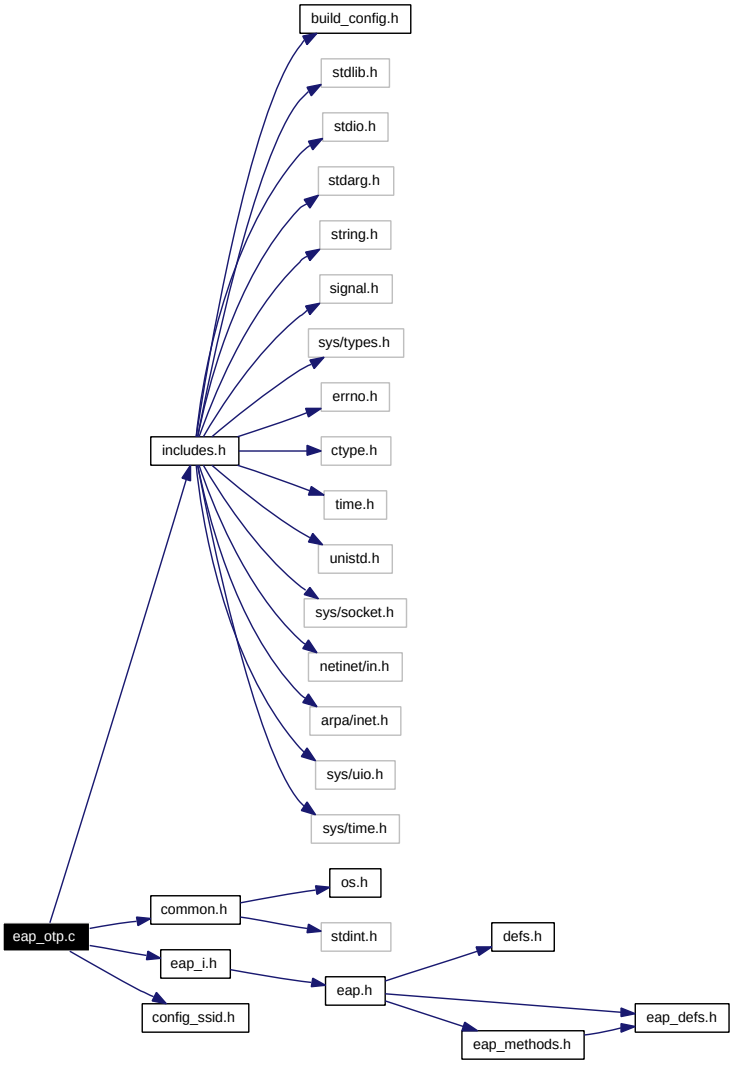


# 6.75 eap_otp.c File Reference

EAP peer method: EAP-OTP (RFC 3748).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "config_ssid.h"
```

Include dependency graph for eap_otp.c:




## Functions

- int **eap_peer_otp_register** (void)

### 6.75.1 Detailed Description

EAP peer method: EAP-OTP (RFC 3748).

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>



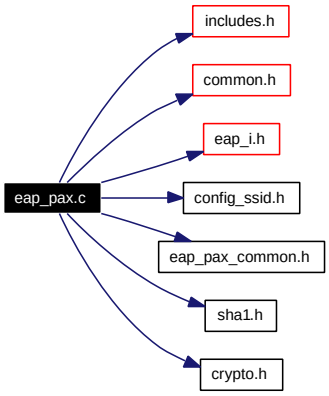
Definition in file eap_otp.c.

## 6.76 eap_pax.c File Reference

EAP peer method: EAP-PAX (draft-clancy-eap-pax-11.txt).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "config_ssid.h"
#include "eap_pax_common.h"
#include "sha1.h"
#include "crypto.h"
```

Include dependency graph for eap_pax.c:




### Functions

- int **eap_peer_pax_register** (void)

### 6.76.1 Detailed Description

EAP peer method: EAP-PAX (draft-clancy-eap-pax-11.txt).

**Copyright**

Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

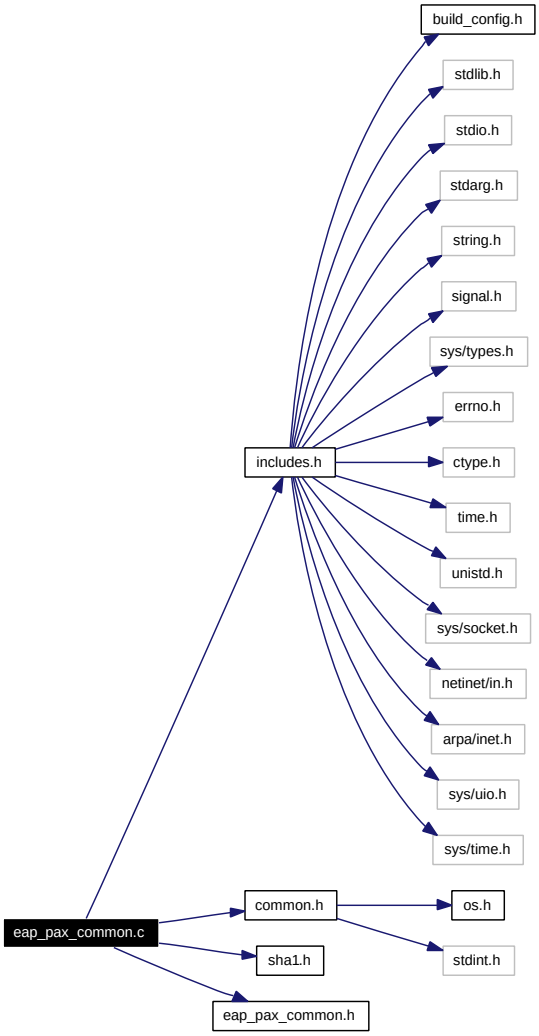
Definition in file eap_pax.c.

## 6.77 eap_pax_common.c File Reference

EAP server/peer: EAP-PAX shared routines.

```
#include "includes.h"
#include "common.h"
#include "sha1.h"
#include "eap_pax_common.h"
```

Include dependency graph for eap_pax_common.c:




### Functions

- int eap_pax_kdf (u8 mac_id, const u8 *key, size_t key_len, const char *identifier, const u8 *entropy, size_t entropy_len, size_t output_len, u8 *output)

  *PAX Key Derivation Function.*

- int eap_pax_mac (u8 mac_id, const u8 *key, size_t key_len, const u8 *data1, size_t data1_len, const u8 *data2, size_t data2_len, const u8 *data3, size_t data3_len, u8 *mac)

  *EAP-PAX MAC.*

- int eap_pax_initial_key_derivation (u8 mac_id, const u8 *ak, const u8 *e, u8 *mk, u8 *ck, u8 *ick)

  *EAP-PAX initial key derivation.*

## 6.77.1 Detailed Description

EAP server/peer: EAP-PAX shared routines.

**Copyright**

Copyright (c) 2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_pax_common.c.

## 6.77.2 Function Documentation

### 6.77.2.1 int eap_pax_initial_key_derivation (u8 *mac_id*, const u8 * *ak*, const u8 * *e*, u8 * *mk*, u8 * *ck*, u8 * *ick*)

EAP-PAX initial key derivation.

**Parameters:**

***mac_id*** MAC ID (EAP_PAX_MAC_*) / currently, only HMAC_SHA1_128 is supported

***ak*** Authentication Key

***e*** Entropy

***mk*** Buffer for the derived Master Key

***ck*** Buffer for the derived Confirmation Key

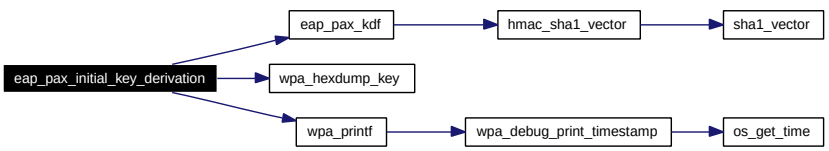
***ick*** Buffer for the derived Integrity Check Key

**Returns:**

0 on success, -1 on failure

Definition at line 136 of file eap_pax_common.c.

Here is the call graph for this function:



#### 6.77.2.2 int eap_pax_kdf (u8 *mac_id*, const u8 * *key*, size_t *key_len*, const char * *identifier*, const u8 * *entropy*, size_t *entropy_len*, size_t *output_len*, u8 * *output*)

PAX Key Derivation Function.

**Parameters:**

***mac_id*** MAC ID (EAP_PAX_MAC_*) / currently, only HMAC_SHA1_128 is supported

***key*** Secret key (X)

***key_len*** Length of the secret key in bytes

***identifier*** Public identifier for the key (Y)

***entropy*** Exchanged entropy to seed the KDF (Z)

***entropy_len*** Length of the entropy in bytes

***output_len*** Output len in bytes (W)

***output*** Buffer for the derived key

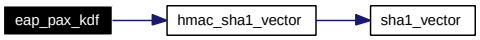
**Returns:**

0 on success, -1 failed

draft-clancy-eap-pax-04.txt, chap. 2.5: PAX-KDF-W(X, Y, Z)

Definition at line 38 of file eap_pax_common.c.

Here is the call graph for this function:




#### 6.77.2.3 int eap_pax_mac (u8 *mac_id*, const u8 * *key*, size_t *key_len*, const u8 * *data1*, size_t *data1_len*, const u8 * *data2*, size_t *data2_len*, const u8 * *data3*, size_t *data3_len*, u8 * *mac*)

EAP-PAX MAC.

**Parameters:**

***mac_id*** MAC ID (EAP_PAX_MAC_*) / currently, only HMAC_SHA1_128 is supported

***key*** Secret key

***key_len*** Length of the secret key in bytes

***data1*** Optional data, first block; NULL if not used

***data1_len*** Length of data1 in bytes

***data2*** Optional data, second block; NULL if not used

***data2_len*** Length of data2 in bytes

***data3*** Optional data, third block; NULL if not used

***data3_len*** Length of data3 in bytes

***mac*** Buffer for the MAC value (EAP_PAX_MAC_LEN = 16 bytes)

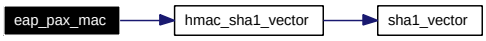
**Returns:**

0 on success, -1 on failure

Wrapper function to calculate EAP-PAX MAC.

Definition at line 95 of file eap_pax_common.c.

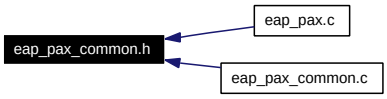
Here is the call graph for this function:



## 6.78 eap_pax_common.h File Reference

EAP server/peer: EAP-PAX shared routines.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **EAP_PAX_FLAGS_MF** 0x01
- #define **EAP_PAX_FLAGS_CE** 0x02
- #define **EAP_PAX_FLAGS_AI** 0x04
- #define **EAP_PAX_MAC_HMAC_SHA1_128** 0x01
- #define **EAP_PAX_HMAC_SHA256_128** 0x02
- #define **EAP_PAX_DH_GROUP_NONE** 0x00
- #define **EAP_PAX_DH_GROUP_2048_MODP** 0x01
- #define **EAP_PAX_DH_GROUP_3072_MODP** 0x02
- #define **EAP_PAX_DH_GROUP_NIST_ECC_P_256** 0x03
- #define **EAP_PAX_PUBLIC_KEY_NONE** 0x00
- #define **EAP_PAX_PUBLIC_KEY_RSAES_OAEP** 0x01
- #define **EAP_PAX_PUBLIC_KEY_RSA_PKCS1_V1_5** 0x02
- #define **EAP_PAX_PUBLIC_KEY_EL_GAMAL_NIST_ECC** 0x03
- #define **EAP_PAX_ADE_VENDOR_SPECIFIC** 0x01
- #define **EAP_PAX_ADE_CLIENT_CHANNEL_BINDING** 0x02
- #define **EAP_PAX_ADE_SERVER_CHANNEL_BINDING** 0x03
- #define **EAP_PAX_RAND_LEN** 32
- #define **EAP_PAX_MAC_LEN** 16
- #define **EAP_PAX_ICV_LEN** 16
- #define **EAP_PAX_AK_LEN** 16
- #define **EAP_PAX_MK_LEN** 16
- #define **EAP_PAX_CK_LEN** 16
- #define **EAP_PAX_ICK_LEN** 16

### Enumerations

- enum {

  **EAP_PAX_OP_STD_1** = 0x01, **EAP_PAX_OP_STD_2** = 0x02, **EAP_PAX_OP_STD_3** = 0x03, **EAP_PAX_OP_SEC_1** = 0x11,

  **EAP_PAX_OP_SEC_2** = 0x12, **EAP_PAX_OP_SEC_3** = 0x13, **EAP_PAX_OP_SEC_4** = 0x14, **EAP_PAX_OP_SEC_5** = 0x15,

  **EAP_PAX_OP_ACK** = 0x21 }

## Functions

- int eap_pax_kdf (u8 mac_id, const u8 *key, size_t key_len, const char *identifier, const u8 *entropy, size_t entropy_len, size_t output_len, u8 *output)

  *PAX Key Derivation Function.*

- int eap_pax_mac (u8 mac_id, const u8 *key, size_t key_len, const u8 *data1, size_t data1_len, const u8 *data2, size_t data2_len, const u8 *data3, size_t data3_len, u8 *mac)

  *EAP-PAX MAC.*

- int eap_pax_initial_key_derivation (u8 mac_id, const u8 *ak, const u8 *e, u8 *mk, u8 *ck, u8 *ick)

  *EAP-PAX initial key derivation.*

## Variables

- eap_pax_hdr **STRUCT_PACKED**

### 6.78.1 Detailed Description

EAP server/peer: EAP-PAX shared routines.

**Copyright**

Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_pax_common.h.

### 6.78.2 Function Documentation

#### 6.78.2.1 int eap_pax_initial_key_derivation (u8 *mac_id*, const u8 * *ak*, const u8 * *e*, u8 * *mk*, u8 * *ck*, u8 * *ick*)

EAP-PAX initial key derivation.

**Parameters:**

***mac_id*** MAC ID (EAP_PAX_MAC_*) / currently, only HMAC_SHA1_128 is supported

***ak*** Authentication Key

***e*** Entropy

***mk*** Buffer for the derived Master Key

***ck*** Buffer for the derived Confirmation Key

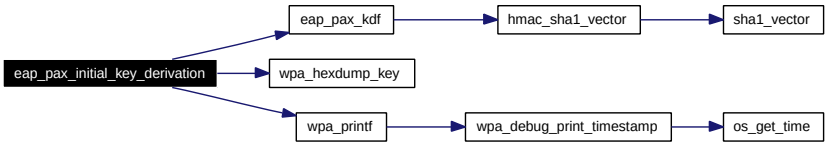
***ick*** Buffer for the derived Integrity Check Key

**Returns:**

0 on success, -1 on failure

Definition at line 136 of file eap_pax_common.c.

Here is the call graph for this function:




#### 6.78.2.2 int eap_pax_kdf (u8 *mac_id*, const u8 * *key*, size_t *key_len*, const char * *identifier*, const u8 * *entropy*, size_t *entropy_len*, size_t *output_len*, u8 * *output*)

PAX Key Derivation Function.

**Parameters:**

***mac_id*** MAC ID (EAP_PAX_MAC_*) / currently, only HMAC_SHA1_128 is supported

***key*** Secret key (X)

***key_len*** Length of the secret key in bytes

***identifier*** Public identifier for the key (Y)

***entropy*** Exchanged entropy to seed the KDF (Z)

***entropy_len*** Length of the entropy in bytes

***output_len*** Output len in bytes (W)

***output*** Buffer for the derived key

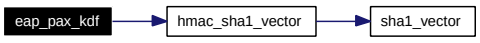
**Returns:**

0 on success, -1 failed

draft-clancy-eap-pax-04.txt, chap. 2.5: PAX-KDF-W(X, Y, Z)

Definition at line 38 of file eap_pax_common.c.

Here is the call graph for this function:




#### 6.78.2.3 int eap_pax_mac (u8 *mac_id*, const u8 * *key*, size_t *key_len*, const u8 * *data1*, size_t *data1_len*, const u8 * *data2*, size_t *data2_len*, const u8 * *data3*, size_t *data3_len*, u8 * *mac*)

EAP-PAX MAC.

**Parameters:**

***mac_id*** MAC ID (EAP_PAX_MAC_*) / currently, only HMAC_SHA1_128 is supported

***key*** Secret key

***key_len*** Length of the secret key in bytes

***data1*** Optional data, first block; NULL if not used

***data1_len*** Length of data1 in bytes

***data2*** Optional data, second block; NULL if not used

***data2_len*** Length of data2 in bytes

***data3*** Optional data, third block; NULL if not used

***data3_len*** Length of data3 in bytes

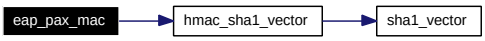
***mac*** Buffer for the MAC value (EAP_PAX_MAC_LEN = 16 bytes)

**Returns:**
0 on success, -1 on failure

Wrapper function to calculate EAP-PAX MAC.

Definition at line 95 of file eap_pax_common.c.

Here is the call graph for this function:

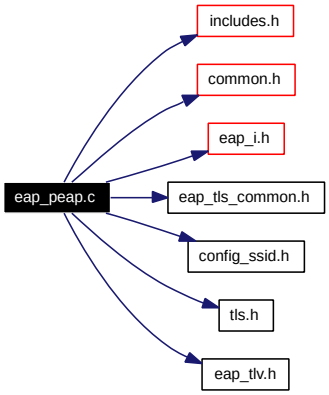


## 6.79 eap_peap.c File Reference

EAP peer method: EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-07.txt).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "config_ssid.h"
#include "tls.h"
#include "eap_tlv.h"
```

Include dependency graph for eap_peap.c:




### Defines

- #define **EAP_PEAP_VERSION** 1

### Functions

- int **eap_peer_peap_register** (void)

### 6.79.1 Detailed Description

EAP peer method: EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-07.txt).

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

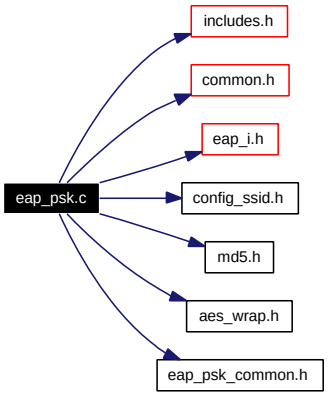
Definition in file eap_peap.c.

# 6.80 eap_psk.c File Reference

EAP peer method: EAP-PSK (draft-bersani-eap-psk-11.txt).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "config_ssid.h"
#include "md5.h"
#include "aes_wrap.h"
#include "eap_psk_common.h"
```

Include dependency graph for eap_psk.c:




## Functions

- int **eap_peer_psk_register** (void)

### 6.80.1 Detailed Description

EAP peer method: EAP-PSK (draft-bersani-eap-psk-11.txt).

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Note: EAP-PSK is an EAP authentication method and as such, completely different from WPA-PSK. This file is not needed for WPA-PSK functionality.

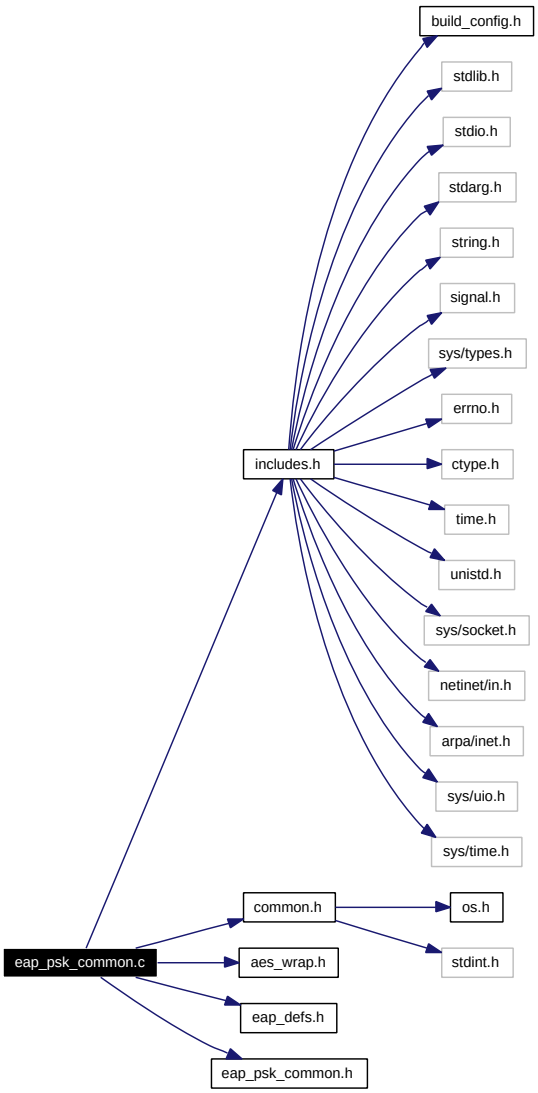
Definition in file eap_psk.c.

## 6.81 eap_psk_common.c File Reference

EAP server/peer: EAP-PSK shared routines.

```
#include "includes.h"
#include "common.h"
#include "aes_wrap.h"
#include "eap_defs.h"
#include "eap_psk_common.h"
```

Include dependency graph for eap_psk_common.c:




### Defines

- #define **aes_block_size** 16

## Functions

- void **eap_psk_key_setup** (const u8 *psk, u8 *ak, u8 *kdk)
- void **eap_psk_derive_keys** (const u8 *kdk, const u8 *rand_p, u8 *tek, u8 *msk, u8 *emsk)

### 6.81.1 Detailed Description

EAP server/peer: EAP-PSK shared routines.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

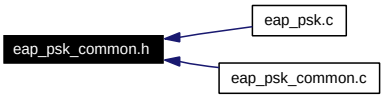
See README and COPYING for more details.

Definition in file eap_psk_common.c.

## 6.82 eap_psk_common.h File Reference

EAP server/peer: EAP-PSK shared routines.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **EAP_PSK_RAND_LEN** 16
- #define **EAP_PSK_MAC_LEN** 16
- #define **EAP_PSK_TEK_LEN** 16
- #define **EAP_PSK_PSK_LEN** 16
- #define **EAP_PSK_AK_LEN** 16
- #define **EAP_PSK_KDK_LEN** 16
- #define **EAP_PSK_R_FLAG_CONT** 1
- #define **EAP_PSK_R_FLAG_DONE_SUCCESS** 2
- #define **EAP_PSK_R_FLAG_DONE_FAILURE** 3
- #define **EAP_PSK_E_FLAG** 0x20
- #define **EAP_PSK_FLAGS_GET_T**(flags) (((flags) & 0xc0) >> 6)
- #define **EAP_PSK_FLAGS_SET_T**(t) ((u8) (t) << 6)

### Functions

- void **eap_psk_key_setup** (const u8 *psk, u8 *ak, u8 *kdk)
- void **eap_psk_derive_keys** (const u8 *kdk, const u8 *rand_p, u8 *tek, u8 *msk, u8 *emsk)

### Variables

- eap_psk_hdr **STRUCT_PACKED**

### 6.82.1 Detailed Description

EAP server/peer: EAP-PSK shared routines.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

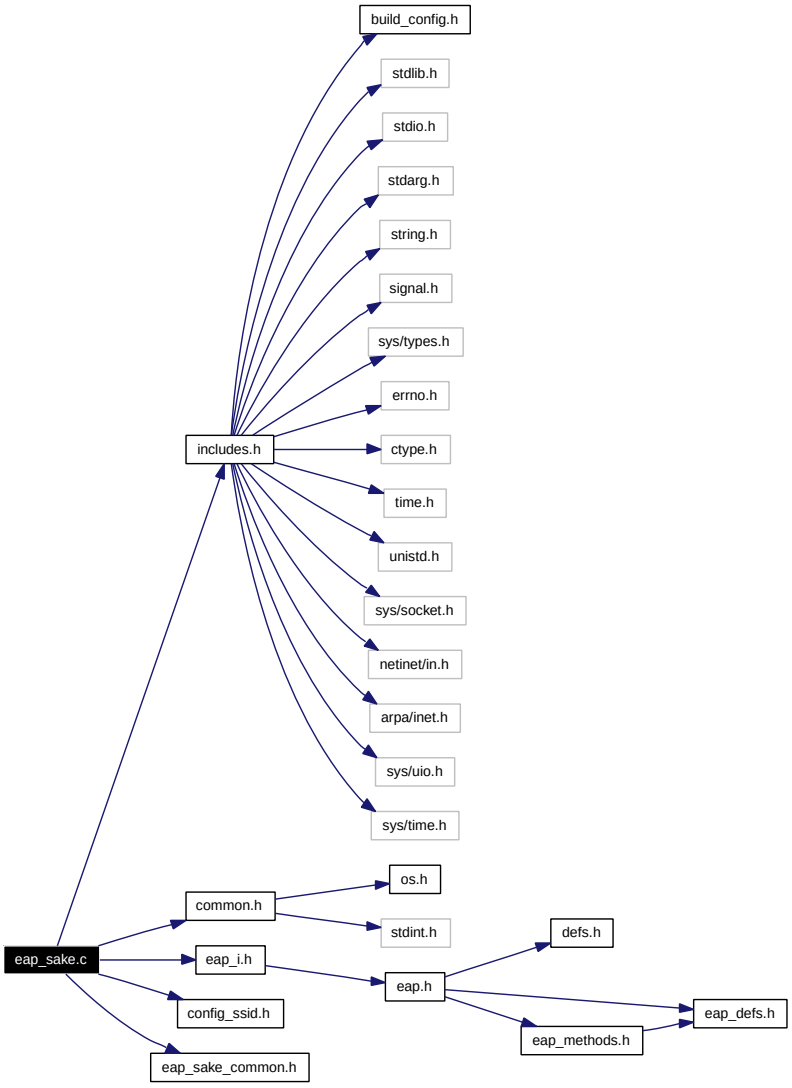
Definition in file eap_psk_common.h.

## 6.83 eap_sake.c File Reference

EAP peer method: EAP-SAKE (RFC 4763).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "config_ssid.h"
#include "eap_sake_common.h"
```

Include dependency graph for eap_sake.c:




## Functions

- int **eap_peer_sake_register** (void)

### 6.83.1 Detailed Description

EAP peer method: EAP-SAKE (RFC 4763).

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>



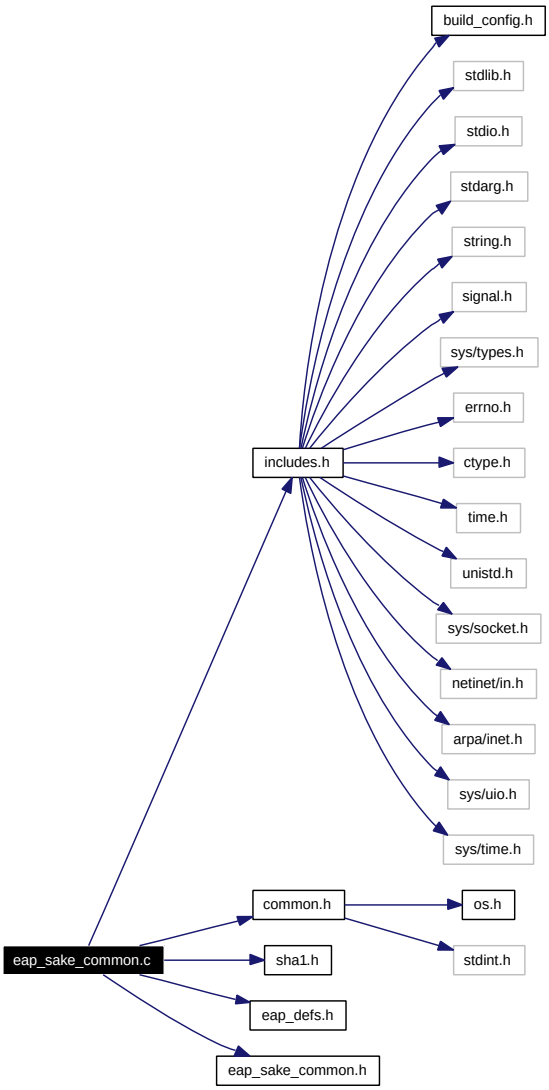
Definition in file eap_sake.c.

# 6.84 eap_sake_common.c File Reference

EAP server/peer: EAP-SAKE shared routines.

```
#include "includes.h"
#include "common.h"
#include "sha1.h"
#include "eap_defs.h"
#include "eap_sake_common.h"
```

Include dependency graph for eap_sake_common.c:




## Functions

- int eap_sake_parse_attributes (const u8 *buf, size_t len, struct eap_sake_parse_attr *attr)

*Parse EAP-SAKE attributes.*

- void eap_sake_derive_keys (const u8 *root_secret_a, const u8 *root_secret_b, const u8 *rand_s, const u8 *rand_p, u8 *tek, u8 *msk, u8 *emsk)

  *Derive EAP-SAKE keys.*

- int eap_sake_compute_mic (const u8 *tek_auth, const u8 *rand_s, const u8 *rand_p, const u8 *serverid, size_t serverid_len, const u8 *peerid, size_t peerid_len, int peer, const u8 *eap, size_t eap_len, const u8 *mic_pos, u8 *mic)

  *Compute EAP-SAKE MIC for an EAP packet.*

### 6.84.1 Detailed Description

EAP server/peer: EAP-SAKE shared routines.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_sake_common.c.

### 6.84.2 Function Documentation

#### 6.84.2.1 int eap_sake_compute_mic (const u8 * *tek_auth*, const u8 * *rand_s*, const u8 * *rand_p*, const u8 * *serverid*, size_t *serverid_len*, const u8 * *peerid*, size_t *peerid_len*, int *peer*, const u8 * *eap*, size_t *eap_len*, const u8 * *mic_pos*, u8 * *mic*)

Compute EAP-SAKE MIC for an EAP packet.

**Parameters:**
***tek_auth*** 16-byte TEK-Auth
***rand_s*** 16-byte RAND_S
***rand_p*** 16-byte RAND_P
***serverid*** SERVERID
***serverid_len*** SERVERID length
***peerid*** PEERID
***peerid_len*** PEERID length
***peer*** MIC calculation for 0 = Server, 1 = Peer message
***eap*** EAP packet
***eap_len*** EAP packet length
***mic_pos*** MIC position in the EAP packet (must be [eap .. eap + eap_len])
***mic*** Buffer for the computed 16-byte MIC

Definition at line 326 of file eap_sake_common.c.

#### 6.84.2.2 void eap_sake_derive_keys (const u8 * *root_secret_a*, const u8 * *root_secret_b*, const u8 * *rand_s*, const u8 * *rand_p*, u8 * *tek*, u8 * *msk*, u8 * *emsk*)

Derive EAP-SAKE keys.

**Parameters:**

***root_secret_a*** 16-byte Root-Secret-A

***root_secret_b*** 16-byte Root-Secret-B

***rand_s*** 16-byte RAND_S

***rand_p*** 16-byte RAND_P

***tek*** Buffer for Temporary EAK Keys (TEK-Auth[16] | TEK-Cipher[16])

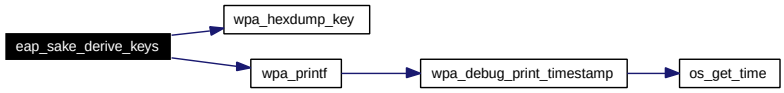
***msk*** Buffer for 64-byte MSK

***emsk*** Buffer for 64-byte EMSK

This function derives EAP-SAKE keys as defined in RFC 4763, section 3.2.6.

Definition at line 268 of file eap_sake_common.c.

Here is the call graph for this function:




#### 6.84.2.3 int eap_sake_parse_attributes (const u8 * *buf*, size_t *len*, struct eap_sake_parse_attr * *attr*)

Parse EAP-SAKE attributes.

**Parameters:**

***buf*** Packet payload (starting with the first attribute)

***len*** Payload length

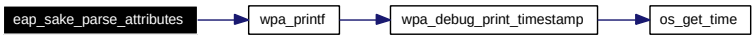
***attr*** Structure to be filled with found attributes

**Returns:**

0 on success or -1 on failure

Definition at line 167 of file eap_sake_common.c.

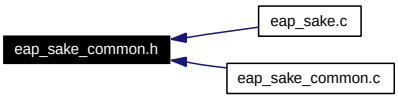
Here is the call graph for this function:



## 6.85 eap_sake_common.h File Reference

EAP server/peer: EAP-SAKE shared routines.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **EAP_SAKE_VERSION** 2
- #define **EAP_SAKE_SUBTYPE_CHALLENGE** 1
- #define **EAP_SAKE_SUBTYPE_CONFIRM** 2
- #define **EAP_SAKE_SUBTYPE_AUTH_REJECT** 3
- #define **EAP_SAKE_SUBTYPE_IDENTITY** 4
- #define **EAP_SAKE_AT_RAND_S** 1
- #define **EAP_SAKE_AT_RAND_P** 2
- #define **EAP_SAKE_AT_MIC_S** 3
- #define **EAP_SAKE_AT_MIC_P** 4
- #define **EAP_SAKE_AT_SERVERID** 5
- #define **EAP_SAKE_AT_PEERID** 6
- #define **EAP_SAKE_AT_SPI_S** 7
- #define **EAP_SAKE_AT_SPI_P** 8
- #define **EAP_SAKE_AT_ANY_ID_REQ** 9
- #define **EAP_SAKE_AT_PERM_ID_REQ** 10
- #define **EAP_SAKE_AT_ENCR_DATA** 128
- #define **EAP_SAKE_AT_IV** 129
- #define **EAP_SAKE_AT_PADDING** 130
- #define **EAP_SAKE_AT_NEXT_TMPID** 131
- #define **EAP_SAKE_AT_MSK_LIFE** 132
- #define **EAP_SAKE_RAND_LEN** 16
- #define **EAP_SAKE_MIC_LEN** 16
- #define **EAP_SAKE_ROOT_SECRET_LEN** 16
- #define **EAP_SAKE_SMS_LEN** 16
- #define **EAP_SAKE_TEK_AUTH_LEN** 16
- #define **EAP_SAKE_TEK_CIPHER_LEN** 16
- #define **EAP_SAKE_TEK_LEN** (EAP_SAKE_TEK_AUTH_LEN + EAP_SAKE_TEK_-CIPHER_LEN)

### Functions

- int eap_sake_parse_attributes (const u8 *buf, size_t len, struct eap_sake_parse_attr *attr)

  *Parse EAP-SAKE attributes.*

- void eap_sake_derive_keys (const u8 *root_secret_a, const u8 *root_secret_b, const u8 *rand_s, const u8 *rand_p, u8 *tek, u8 *msk, u8 *emsk)

  *Derive EAP-SAKE keys.*

- int eap_sake_compute_mic (const u8 *tek_auth, const u8 *rand_s, const u8 *rand_p, const u8 *serverid, size_t serverid_len, const u8 *peerid, size_t peerid_len, int peer, const u8 *eap, size_t eap_len, const u8 *mic_pos, u8 *mic)

  *Compute EAP-SAKE MIC for an EAP packet.*

## Variables

- eap_sake_hdr **STRUCT_PACKED**

### 6.85.1 Detailed Description

EAP server/peer: EAP-SAKE shared routines.

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_sake_common.h.

### 6.85.2 Function Documentation

#### 6.85.2.1 int eap_sake_compute_mic (const u8 * *tek_auth*, const u8 * *rand_s*, const u8 * *rand_p*, const u8 * *serverid*, size_t *serverid_len*, const u8 * *peerid*, size_t *peerid_len*, int *peer*, const u8 * *eap*, size_t *eap_len*, const u8 * *mic_pos*, u8 * *mic*)

Compute EAP-SAKE MIC for an EAP packet.

**Parameters:**

***tek_auth*** 16-byte TEK-Auth

***rand_s*** 16-byte RAND_S

***rand_p*** 16-byte RAND_P

***serverid*** SERVERID

***serverid_len*** SERVERID length

***peerid*** PEERID

***peerid_len*** PEERID length

***peer*** MIC calculation for 0 = Server, 1 = Peer message

***eap*** EAP packet

***eap_len*** EAP packet length

***mic_pos*** MIC position in the EAP packet (must be [eap .. eap + eap_len])

***mic*** Buffer for the computed 16-byte MIC

Definition at line 326 of file eap_sake_common.c.

#### 6.85.2.2 void eap_sake_derive_keys (const u8 * *root_secret_a*, const u8 * *root_secret_b*, const u8 * *rand_s*, const u8 * *rand_p*, u8 * *tek*, u8 * *msk*, u8 * *emsk*)

Derive EAP-SAKE keys.

**Parameters:**

***root_secret_a*** 16-byte Root-Secret-A

***root_secret_b*** 16-byte Root-Secret-B

***rand_s*** 16-byte RAND_S

***rand_p*** 16-byte RAND_P

***tek*** Buffer for Temporary EAK Keys (TEK-Auth[16] | TEK-Cipher[16])

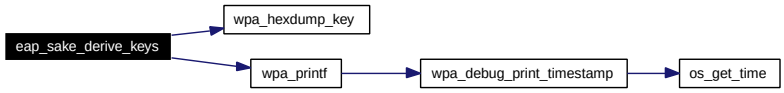
***msk*** Buffer for 64-byte MSK

***emsk*** Buffer for 64-byte EMSK

This function derives EAP-SAKE keys as defined in RFC 4763, section 3.2.6.

Definition at line 268 of file eap_sake_common.c.

Here is the call graph for this function:




#### 6.85.2.3 int eap_sake_parse_attributes (const u8 * *buf*, size_t *len*, struct eap_sake_parse_attr * *attr*)

Parse EAP-SAKE attributes.

**Parameters:**

***buf*** Packet payload (starting with the first attribute)

***len*** Payload length

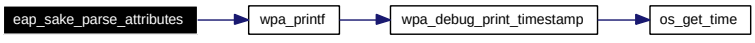
***attr*** Structure to be filled with found attributes

**Returns:**

0 on success or -1 on failure

Definition at line 167 of file eap_sake_common.c.

Here is the call graph for this function:

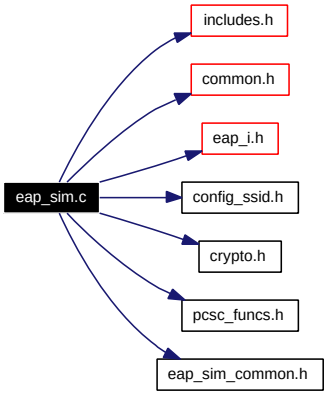


# 6.86 eap_sim.c File Reference

EAP peer method: EAP-SIM (RFC 4186).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "config_ssid.h"
#include "crypto.h"
#include "pcsc_funcs.h"
#include "eap_sim_common.h"
```

Include dependency graph for eap_sim.c:




## Defines

- #define **CLEAR_PSEUDONYM** 0x01
- #define **CLEAR_REAUTH_ID** 0x02
- #define **CLEAR_EAP_ID** 0x04

## Functions

- int **eap_peer_sim_register** (void)

### 6.86.1 Detailed Description

EAP peer method: EAP-SIM (RFC 4186).

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

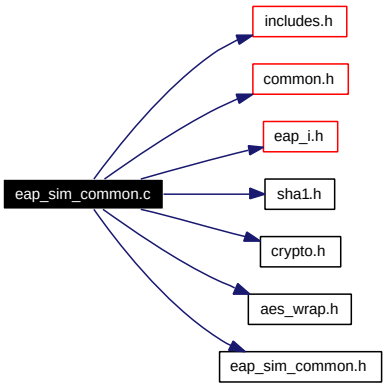
Definition in file eap_sim.c.

# 6.87 eap_sim_common.c File Reference

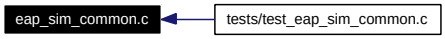
EAP peer: EAP-SIM/AKA shared routines.

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "sha1.h"
#include "crypto.h"
#include "aes_wrap.h"
#include "eap_sim_common.h"
```

Include dependency graph for eap_sim_common.c:




This graph shows which files directly or indirectly include this file:




## Defines

- #define **EAP_SIM_INIT_LEN** 128

## Functions

- void **eap_sim_derive_mk** (const u8 ∗identity, size_t identity_len, const u8 ∗nonce_mt, u16 selected_version, const u8 ∗ver_list, size_t ver_list_len, int num_chal, const u8 ∗kc, u8 ∗mk)
- void **eap_aka_derive_mk** (const u8 ∗identity, size_t identity_len, const u8 ∗ik, const u8 ∗ck, u8 ∗mk)
- int **eap_sim_derive_keys** (const u8 ∗mk, u8 ∗k_encr, u8 ∗k_aut, u8 ∗msk, u8 ∗emsk)
- int **eap_sim_derive_keys_reauth** (u16 _counter, const u8 ∗identity, size_t identity_len, const u8 ∗nonce_s, const u8 ∗mk, u8 ∗msk, u8 ∗emsk)
- int **eap_sim_verify_mac** (const u8 ∗k_aut, const u8 ∗req, size_t req_len, const u8 ∗mac, const u8 ∗extra, size_t extra_len)

- void **eap_sim_add_mac** (const u8 *k_aut, u8 *msg, size_t msg_len, u8 *mac, const u8 *extra, size_t extra_len)
- int **eap_sim_parse_attr** (const u8 *start, const u8 *end, struct eap_sim_attrs *attr, int aka, int encr)
- u8 * **eap_sim_parse_encr** (const u8 *k_encr, const u8 *encr_data, size_t encr_data_len, const u8 *iv, struct eap_sim_attrs *attr, int aka)
- eap_sim_msg * **eap_sim_msg_init** (int code, int id, int type, int subtype)
- u8 * **eap_sim_msg_finish** (struct eap_sim_msg *msg, size_t *len, const u8 *k_aut, const u8 *extra, size_t extra_len)
- void **eap_sim_msg_free** (struct eap_sim_msg *msg)
- u8 * **eap_sim_msg_add_full** (struct eap_sim_msg *msg, u8 attr, const u8 *data, size_t len)
- u8 * **eap_sim_msg_add** (struct eap_sim_msg *msg, u8 attr, u16 value, const u8 *data, size_t len)
- u8 * **eap_sim_msg_add_mac** (struct eap_sim_msg *msg, u8 attr)
- int **eap_sim_msg_add_encr_start** (struct eap_sim_msg *msg, u8 attr_iv, u8 attr_encr)
- int **eap_sim_msg_add_encr_end** (struct eap_sim_msg *msg, u8 *k_encr, int attr_pad)
- void **eap_sim_report_notification** (void *msg_ctx, int notification, int aka)

### 6.87.1 Detailed Description

EAP peer: EAP-SIM/AKA shared routines.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

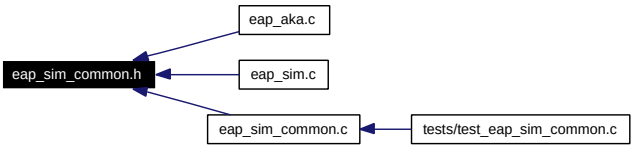


Definition in file eap_sim_common.c.

# 6.88 eap_sim_common.h File Reference

EAP peer: EAP-SIM/AKA shared routines.

This graph shows which files directly or indirectly include this file:




## Defines

- #define **EAP_SIM_NONCE_S_LEN** 16
- #define **EAP_SIM_NONCE_MT_LEN** 16
- #define **EAP_SIM_MAC_LEN** 16
- #define **EAP_SIM_MK_LEN** 20
- #define **EAP_SIM_K_AUT_LEN** 16
- #define **EAP_SIM_K_ENCR_LEN** 16
- #define **EAP_SIM_KEYING_DATA_LEN** 64
- #define **EAP_SIM_IV_LEN** 16
- #define **EAP_SIM_KC_LEN** 8
- #define **EAP_SIM_SRES_LEN** 4
- #define **GSM_RAND_LEN** 16
- #define **EAP_SIM_VERSION** 1
- #define **EAP_SIM_SUBTYPE_START** 10
- #define **EAP_SIM_SUBTYPE_CHALLENGE** 11
- #define **EAP_SIM_SUBTYPE_NOTIFICATION** 12
- #define **EAP_SIM_SUBTYPE_REAUTHENTICATION** 13
- #define **EAP_SIM_SUBTYPE_CLIENT_ERROR** 14
- #define **EAP_SIM_UNABLE_TO_PROCESS_PACKET** 0
- #define **EAP_SIM_UNSUPPORTED_VERSION** 1
- #define **EAP_SIM_INSUFFICIENT_NUM_OF_CHAL** 2
- #define **EAP_SIM_RAND_NOT_FRESH** 3
- #define **EAP_SIM_MAX_FAST_REAUTHS** 1000
- #define **EAP_SIM_MAX_CHAL** 3
- #define **EAP_AKA_SUBTYPE_CHALLENGE** 1
- #define **EAP_AKA_SUBTYPE_AUTHENTICATION_REJECT** 2
- #define **EAP_AKA_SUBTYPE_SYNCHRONIZATION_FAILURE** 4
- #define **EAP_AKA_SUBTYPE_IDENTITY** 5
- #define **EAP_AKA_SUBTYPE_NOTIFICATION** 12
- #define **EAP_AKA_SUBTYPE_REAUTHENTICATION** 13
- #define **EAP_AKA_SUBTYPE_CLIENT_ERROR** 14
- #define **EAP_AKA_UNABLE_TO_PROCESS_PACKET** 0
- #define **EAP_AKA_RAND_LEN** 16
- #define **EAP_AKA_AUTN_LEN** 16
- #define **EAP_AKA_AUTS_LEN** 14
- #define **EAP_AKA_RES_MAX_LEN** 16
- #define **EAP_AKA_IK_LEN** 16

- #define **EAP_AKA_CK_LEN** 16
- #define **EAP_AKA_MAX_FAST_REAUTHS** 1000
- #define **EAP_AKA_MIN_RES_LEN** 4
- #define **EAP_AKA_MAX_RES_LEN** 16
- #define **EAP_SIM_AT_RAND** 1
- #define **EAP_SIM_AT_AUTN** 2
- #define **EAP_SIM_AT_RES** 3
- #define **EAP_SIM_AT_AUTS** 4
- #define **EAP_SIM_AT_PADDING** 6
- #define **EAP_SIM_AT_NONCE_MT** 7
- #define **EAP_SIM_AT_PERMANENT_ID_REQ** 10
- #define **EAP_SIM_AT_MAC** 11
- #define **EAP_SIM_AT_NOTIFICATION** 12
- #define **EAP_SIM_AT_ANY_ID_REQ** 13
- #define **EAP_SIM_AT_IDENTITY** 14
- #define **EAP_SIM_AT_VERSION_LIST** 15
- #define **EAP_SIM_AT_SELECTED_VERSION** 16
- #define **EAP_SIM_AT_FULLAUTH_ID_REQ** 17
- #define **EAP_SIM_AT_COUNTER** 19
- #define **EAP_SIM_AT_COUNTER_TOO_SMALL** 20
- #define **EAP_SIM_AT_NONCE_S** 21
- #define **EAP_SIM_AT_CLIENT_ERROR_CODE** 22
- #define **EAP_SIM_AT_IV** 129
- #define **EAP_SIM_AT_ENCR_DATA** 130
- #define **EAP_SIM_AT_NEXT_PSEUDONYM** 132
- #define **EAP_SIM_AT_NEXT_REAUTH_ID** 133
- #define **EAP_SIM_AT_CHECKCODE** 134
- #define **EAP_SIM_AT_RESULT_IND** 135
- #define **EAP_SIM_GENERAL_FAILURE_AFTER_AUTH** 0
- #define **EAP_SIM_TEMPORARILY_DENIED** 1026
- #define **EAP_SIM_NOT_SUBSCRIBED** 1031
- #define **EAP_SIM_GENERAL_FAILURE_BEFORE_AUTH** 16384
- #define **EAP_SIM_SUCCESS** 32768

## Enumerations

- enum **eap_sim_id_req** { **NO_ID_REQ**, **ANY_ID**, **FULLAUTH_ID**, **PERMANENT_ID** }

## Functions

- void **eap_sim_derive_mk** (const u8 ∗identity, size_t identity_len, const u8 ∗nonce_mt, u16 selected_version, const u8 ∗ver_list, size_t ver_list_len, int num_chal, const u8 ∗kc, u8 ∗mk)
- void **eap_aka_derive_mk** (const u8 ∗identity, size_t identity_len, const u8 ∗ik, const u8 ∗ck, u8 ∗mk)
- int **eap_sim_derive_keys** (const u8 ∗mk, u8 ∗k_encr, u8 ∗k_aut, u8 ∗msk, u8 ∗emsk)
- int **eap_sim_derive_keys_reauth** (u16 _counter, const u8 ∗identity, size_t identity_len, const u8 ∗nonce_s, const u8 ∗mk, u8 ∗msk, u8 ∗emsk)
- int **eap_sim_verify_mac** (const u8 ∗k_aut, const u8 ∗req, size_t req_len, const u8 ∗mac, const u8 ∗extra, size_t extra_len)

- void **eap_sim_add_mac** (const u8 *k_aut, u8 *msg, size_t msg_len, u8 *mac, const u8 *extra, size_t extra_len)
- int **eap_sim_parse_attr** (const u8 *start, const u8 *end, struct eap_sim_attrs *attr, int aka, int encr)
- u8 * **eap_sim_parse_encr** (const u8 *k_encr, const u8 *encr_data, size_t encr_data_len, const u8 *iv, struct eap_sim_attrs *attr, int aka)
- eap_sim_msg * **eap_sim_msg_init** (int code, int id, int type, int subtype)
- u8 * **eap_sim_msg_finish** (struct eap_sim_msg *msg, size_t *len, const u8 *k_aut, const u8 *extra, size_t extra_len)
- void **eap_sim_msg_free** (struct eap_sim_msg *msg)
- u8 * **eap_sim_msg_add_full** (struct eap_sim_msg *msg, u8 attr, const u8 *data, size_t len)
- u8 * **eap_sim_msg_add** (struct eap_sim_msg *msg, u8 attr, u16 value, const u8 *data, size_t len)
- u8 * **eap_sim_msg_add_mac** (struct eap_sim_msg *msg, u8 attr)
- int **eap_sim_msg_add_encr_start** (struct eap_sim_msg *msg, u8 attr_iv, u8 attr_encr)
- int **eap_sim_msg_add_encr_end** (struct eap_sim_msg *msg, u8 *k_encr, int attr_pad)
- void **eap_sim_report_notification** (void *msg_ctx, int notification, int aka)

## 6.88.1 Detailed Description

EAP peer: EAP-SIM/AKA shared routines.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

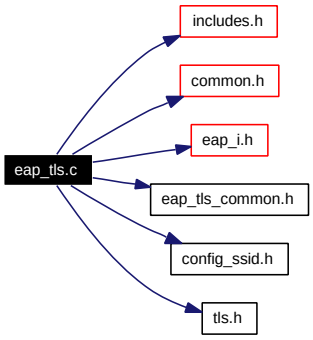
Definition in file eap_sim_common.h.

## 6.89 eap_tls.c File Reference

EAP peer method: EAP-TLS (RFC 2716).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "config_ssid.h"
#include "tls.h"
```

Include dependency graph for eap_tls.c:




### Functions

- int **eap_peer_tls_register** (void)

### 6.89.1 Detailed Description

EAP peer method: EAP-TLS (RFC 2716).

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

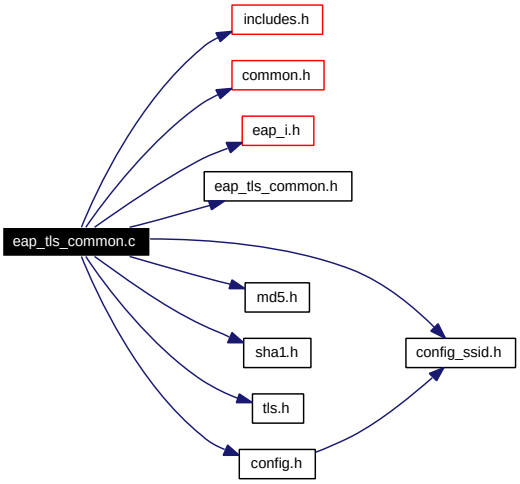
Definition in file eap_tls.c.

# 6.90 eap_tls_common.c File Reference

EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "config_ssid.h"
#include "md5.h"
#include "sha1.h"
#include "tls.h"
#include "config.h"
```

Include dependency graph for eap_tls_common.c:




## Functions

- int eap_tls_ssl_init (struct eap_sm *sm, struct eap_ssl_data *data, struct wpa_ssid *config)

  *Initialize shared TLS functionality.*

- void eap_tls_ssl_deinit (struct eap_sm *sm, struct eap_ssl_data *data)

  *Deinitialize shared TLS functionality.*

- u8 * eap_tls_derive_key (struct eap_sm *sm, struct eap_ssl_data *data, const char *label, size_t len)

  *Derive a key based on TLS session data.*

- const u8 * eap_tls_data_reassemble (struct eap_sm *sm, struct eap_ssl_data *data, const u8 *in_data, size_t in_len, size_t *out_len, int *need_more_input)

  *Reassemble TLS data.*

- int eap_tls_process_helper (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_type, int peap_version, u8 id, const u8 *in_data, size_t in_len, u8 **out_data, size_t *out_len)

  *Process TLS handshake message.*

- u8 * eap_tls_build_ack (struct eap_ssl_data *data, size_t *respDataLen, u8 id, EapType eap_type, int peap_version)

  *Build a TLS ACK frames.*

- int eap_tls_reauth_init (struct eap_sm *sm, struct eap_ssl_data *data)

  *Re-initialize shared TLS for session resumption.*

- int eap_tls_status (struct eap_sm *sm, struct eap_ssl_data *data, char *buf, size_t buflen, int verbose)

  *Get TLS status.*

- const u8 * eap_tls_process_init (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_type, struct eap_method_ret *ret, const u8 *reqData, size_t reqDataLen, size_t *len, u8 *flags)

  *Initial validation and processing of EAP requests.*

## 6.90.1 Detailed Description

EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_tls_common.c.

## 6.90.2 Function Documentation

### 6.90.2.1 u8* eap_tls_build_ack (struct eap_ssl_data * *data*, size_t * *respDataLen*, u8 *id*, EapType *eap_type*, int *peap_version*)

Build a TLS ACK frames.

**Parameters:**

***data*** Data for TLS processing

***respDataLen*** Buffer for returning the length of the response message

***id*** EAP identifier for the response

***eap_type*** EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

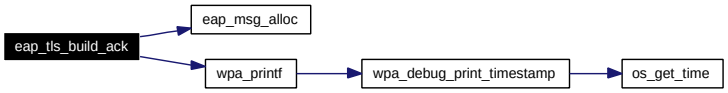
***peap_version*** Version number for EAP-PEAP/TTLS

**Returns:**

Pointer to allocated ACK frames or NULL on failure

Definition at line 561 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.90.2.2 const u8* eap_tls_data_reassemble (struct eap_sm * *sm*, struct eap_ssl_data * *data*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, int * *need_more_input*)

Reassemble TLS data.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***in_data*** Next incoming TLS segment

***in_len*** Length of in_data

***out_len*** Variable for returning output data length

***need_more_input*** Variable for returning whether more input data is needed to reassemble this TLS packet

**Returns:**

Pointer to output data, NULL on error or when more data is needed for the full message (in which case, *need_more_input is also set to 1).

This function reassembles TLS fragments. Caller must not free the returned data buffer since an internal pointer to it is maintained.

Definition at line 303 of file eap_tls_common.c.

Here is the call graph for this function:


#### 6.90.2.3 u8* eap_tls_derive_key (struct eap_sm * *sm*, struct eap_ssl_data * *data*, const char * *label*, size_t *len*)

Derive a key based on TLS session data.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***label*** Label string for deriving the keys, e.g., "client EAP encryption"

***len*** Length of the key material to generate (usually 64 for MSK)

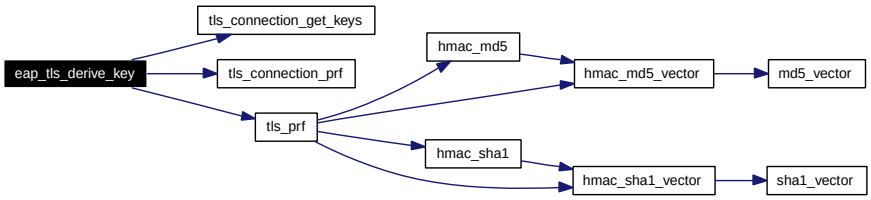
**Returns:**
Pointer to allocated key on success or NULL on failure

This function uses TLS-PRF to generate pseudo-random data based on the TLS session data (client/server random and master key). Each key type may use a different label to bind the key usage into the generated material.

The caller is responsible for freeing the returned buffer.

Definition at line 238 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.90.2.4 int eap_tls_process_helper (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, int *peap_version*, u8 *id*, const u8 * *in_data*, size_t *in_len*, u8 ** *out_data*, size_t * *out_len*)

Process TLS handshake message.

**Parameters:**
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***eap_type*** EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

***peap_version*** Version number for EAP-PEAP/TTLS

***id*** EAP identifier for the response

***in_data*** Message received from the server

***in_len*** Length of in_data

***out_data*** Buffer for returning a pointer to the response message

***out_len*** Buffer for returning the length of the response message

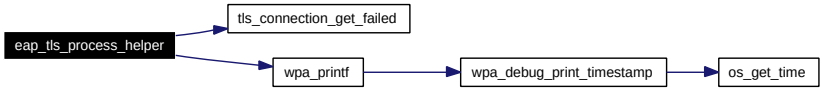
**Returns:**
0 on success, 1 if more input data is needed, or -1 on failure

This function can be used to process TLS handshake messages. It reassembles the received fragments and uses a TLS library to process the messages. The response data from the TLS library is fragmented to suitable output messages that the caller can send out.

out_data is used to return the response message if the return value of this function is 0 or -1. In case of failure, the message is likely a TLS alarm message. The caller is responsible for freeing the allocated buffer if *out_data is not NULL.

Definition at line 504 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.90.2.5 const u8* eap_tls_process_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, struct eap_method_ret * *ret*, const u8 * *reqData*, size_t *reqDataLen*, size_t * *len*, u8 * *flags*)

Initial validation and processing of EAP requests.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***eap_type*** EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

***ret*** Return values from EAP request validation and processing

***reqData*** EAP request to be processed (eapReqData)

***reqDataLen*** Length of the EAP request

***len*** Buffer for returning length of the remaining payload

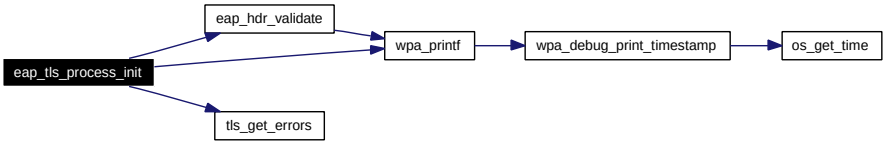
***flags*** Buffer for returning TLS flags

**Returns:**

Buffer to payload after TLS flags and length or NULL on failure

Definition at line 638 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.90.2.6 int eap_tls_reauth_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*)

Re-initialize shared TLS for session resumption.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

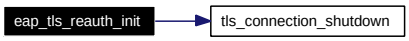
***data*** Data for TLS processing

**Returns:**

0 on success, -1 on failure

Definition at line 584 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.90.2.7 void eap_tls_ssl_deinit (struct eap_sm * *sm*, struct eap_ssl_data * *data*)

Deinitialize shared TLS functionality.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

This function deinitializes shared TLS functionality that was initialized with eap_tls_ssl_init().

Definition at line 215 of file eap_tls_common.c.

Here is the call graph for this function:


#### 6.90.2.8 int eap_tls_ssl_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*, struct wpa_ssid * *config*)

Initialize shared TLS functionality.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***config*** Pointer to the network configuration

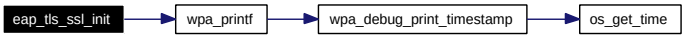
**Returns:**

0 on success, -1 on failure

This function is used to initialize shared TLS functionality for EAP-TLS, EAP-PEAP, EAP-TTLS, and EAP-FAST.

Definition at line 165 of file eap_tls_common.c.

Here is the call graph for this function:



#### 6.90.2.9 int eap_tls_status (struct eap_sm * *sm*, struct eap_ssl_data * *data*, char * *buf*, size_t *buflen*, int *verbose*)

Get TLS status.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

**Returns:**

Number of bytes written to buf.

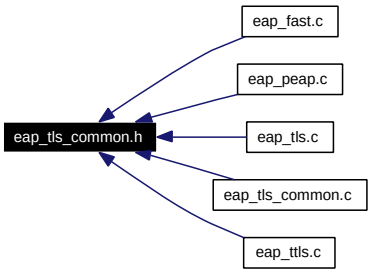
Definition at line 607 of file eap_tls_common.c.

Here is the call graph for this function:

## 6.91 eap_tls_common.h File Reference

EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **EAP_TLS_FLAGS_LENGTH_INCLUDED** 0x80
- #define **EAP_TLS_FLAGS_MORE_FRAGMENTS** 0x40
- #define **EAP_TLS_FLAGS_START** 0x20
- #define **EAP_PEAP_VERSION_MASK** 0x07
- #define **EAP_TLS_KEY_LEN** 64

### Functions

- int eap_tls_ssl_init (struct eap_sm *sm, struct eap_ssl_data *data, struct wpa_ssid *config)

  *Initialize shared TLS functionality.*

- void eap_tls_ssl_deinit (struct eap_sm *sm, struct eap_ssl_data *data)

  *Deinitialize shared TLS functionality.*

- u8 * eap_tls_derive_key (struct eap_sm *sm, struct eap_ssl_data *data, const char *label, size_t len)

  *Derive a key based on TLS session data.*

- const u8 * eap_tls_data_reassemble (struct eap_sm *sm, struct eap_ssl_data *data, const u8 *in_data, size_t in_len, size_t *out_len, int *need_more_input)

  *Reassemble TLS data.*

- int eap_tls_process_helper (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_type, int peap_version, u8 id, const u8 *in_data, size_t in_len, u8 **out_data, size_t *out_len)

  *Process TLS handshake message.*

- u8 * eap_tls_build_ack (struct eap_ssl_data *data, size_t *respDataLen, u8 id, EapType eap_type, int peap_version)

  *Build a TLS ACK frames.*

- int eap_tls_reauth_init (struct eap_sm *sm, struct eap_ssl_data *data)

  *Re-initialize shared TLS for session resumption.*

- int eap_tls_status (struct eap_sm *sm, struct eap_ssl_data *data, char *buf, size_t buflen, int verbose)

  *Get TLS status.*

- const u8 * eap_tls_process_init (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_type, struct eap_method_ret *ret, const u8 *reqData, size_t reqDataLen, size_t *len, u8 *flags)

  *Initial validation and processing of EAP requests.*

## 6.91.1 Detailed Description

EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_tls_common.h.

## 6.91.2 Function Documentation

### 6.91.2.1 u8* eap_tls_build_ack (struct eap_ssl_data * *data*, size_t * *respDataLen*, u8 *id*, EapType *eap_type*, int *peap_version*)

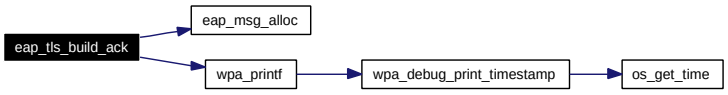
Build a TLS ACK frames.

**Parameters:**
***data*** Data for TLS processing
***respDataLen*** Buffer for returning the length of the response message
***id*** EAP identifier for the response
***eap_type*** EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)
***peap_version*** Version number for EAP-PEAP/TTLS

**Returns:**
Pointer to allocated ACK frames or NULL on failure

Definition at line 561 of file eap_tls_common.c.

Here is the call graph for this function:



#### 6.91.2.2 const u8* eap_tls_data_reassemble (struct eap_sm * *sm*, struct eap_ssl_data * *data*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, int * *need_more_input*)

Reassemble TLS data.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***in_data*** Next incoming TLS segment

***in_len*** Length of in_data

***out_len*** Variable for returning output data length

***need_more_input*** Variable for returning whether more input data is needed to reassemble this TLS packet

**Returns:**

Pointer to output data, NULL on error or when more data is needed for the full message (in which case, *need_more_input is also set to 1).

This function reassembles TLS fragments. Caller must not free the returned data buffer since an internal pointer to it is maintained.

Definition at line 303 of file eap_tls_common.c.

Here is the call graph for this function:


#### 6.91.2.3 u8* eap_tls_derive_key (struct eap_sm * *sm*, struct eap_ssl_data * *data*, const char * *label*, size_t *len*)

Derive a key based on TLS session data.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***label*** Label string for deriving the keys, e.g., "client EAP encryption"

***len*** Length of the key material to generate (usually 64 for MSK)

**Returns:**

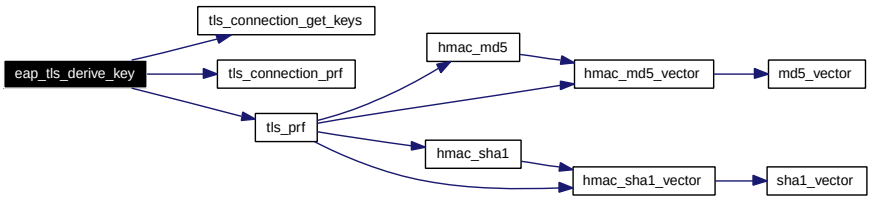
Pointer to allocated key on success or NULL on failure

This function uses TLS-PRF to generate pseudo-random data based on the TLS session data (client/server random and master key). Each key type may use a different label to bind the key usage into the generated material.

The caller is responsible for freeing the returned buffer.

Definition at line 238 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.91.2.4 int eap_tls_process_helper (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, int *peap_version*, u8 *id*, const u8 * *in_data*, size_t *in_len*, u8 ** *out_data*, size_t * *out_len*)

Process TLS handshake message.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***eap_type*** EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

***peap_version*** Version number for EAP-PEAP/TTLS

***id*** EAP identifier for the response

***in_data*** Message received from the server

***in_len*** Length of in_data

***out_data*** Buffer for returning a pointer to the response message

***out_len*** Buffer for returning the length of the response message

**Returns:**

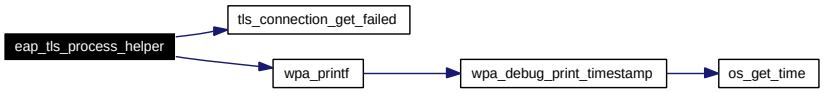
0 on success, 1 if more input data is needed, or -1 on failure

This function can be used to process TLS handshake messages. It reassembles the received fragments and uses a TLS library to process the messages. The response data from the TLS library is fragmented to suitable output messages that the caller can send out.

out_data is used to return the response message if the return value of this function is 0 or -1. In case of failure, the message is likely a TLS alarm message. The caller is responsible for freeing the allocated buffer if *out_data is not NULL.

Definition at line 504 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.91.2.5 const u8* eap_tls_process_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, struct eap_method_ret * *ret*, const u8 * *reqData*, size_t *reqDataLen*, size_t * *len*, u8 * *flags*)

Initial validation and processing of EAP requests.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***eap_type*** EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

***ret*** Return values from EAP request validation and processing

***reqData*** EAP request to be processed (eapReqData)

***reqDataLen*** Length of the EAP request

***len*** Buffer for returning length of the remaining payload

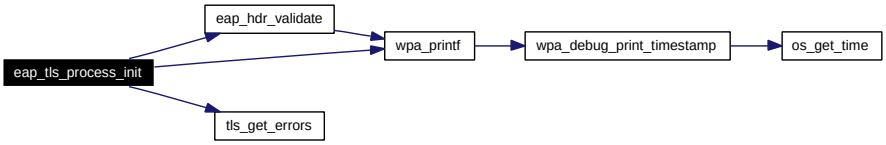
***flags*** Buffer for returning TLS flags

**Returns:**

Buffer to payload after TLS flags and length or NULL on failure

Definition at line 638 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.91.2.6 int eap_tls_reauth_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*)

Re-initialize shared TLS for session resumption.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

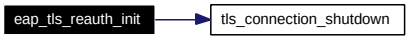
***data*** Data for TLS processing

**Returns:**

0 on success, -1 on failure

Definition at line 584 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.91.2.7 void eap_tls_ssl_deinit (struct eap_sm * *sm*, struct eap_ssl_data * *data*)

Deinitialize shared TLS functionality.

**Parameters:**

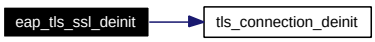
***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

This function deinitializes shared TLS functionality that was initialized with eap_tls_ssl_init().

Definition at line 215 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.91.2.8 int eap_tls_ssl_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*, struct wpa_ssid * *config*)

Initialize shared TLS functionality.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***config*** Pointer to the network configuration

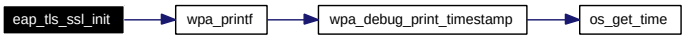
**Returns:**

0 on success, -1 on failure

This function is used to initialize shared TLS functionality for EAP-TLS, EAP-PEAP, EAP-TTLS, and EAP-FAST.

Definition at line 165 of file eap_tls_common.c.

Here is the call graph for this function:




#### 6.91.2.9 int eap_tls_status (struct eap_sm * *sm*, struct eap_ssl_data * *data*, char * *buf*, size_t *buflen*, int *verbose*)

Get TLS status.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***data*** Data for TLS processing

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

**Returns:**

Number of bytes written to buf.

Definition at line 607 of file eap_tls_common.c.

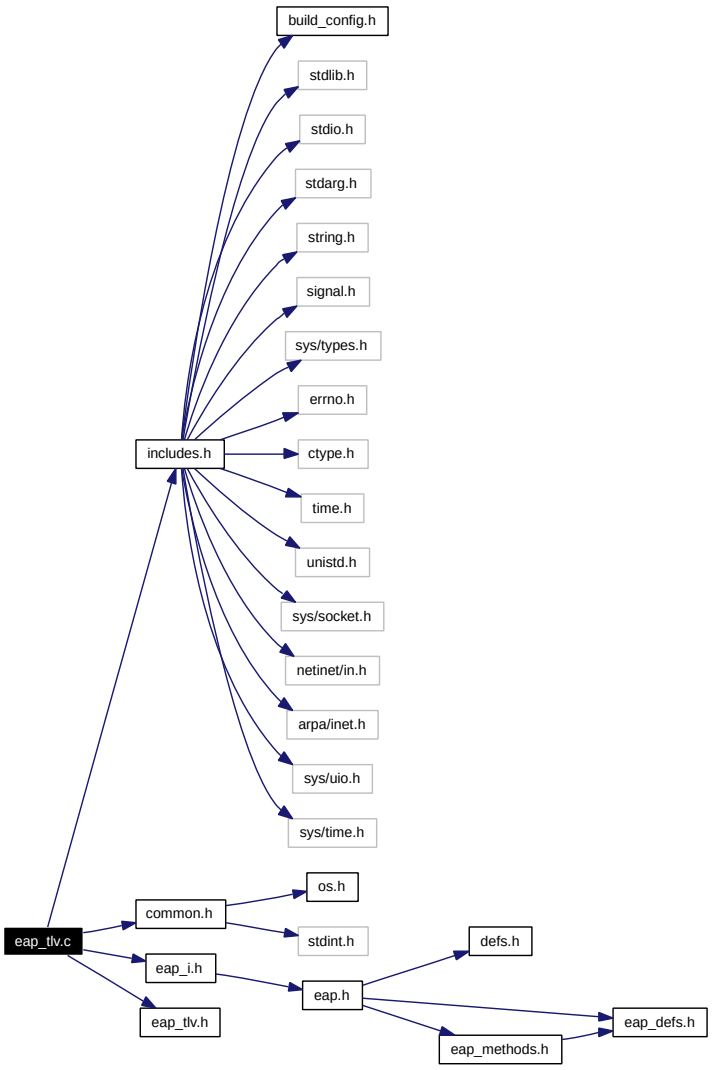
Here is the call graph for this function:

# 6.92 eap_tlv.c File Reference

EAP peer method: EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "eap_tlv.h"
```

Include dependency graph for eap_tlv.c:




## Functions

- u8 * eap_tlv_build_nak (int id, u16 nak_type, size_t *resp_len)
  *Build EAP-TLV NAK message.*
- u8 * eap_tlv_build_result (int id, u16 status, size_t *resp_len)

*Build EAP-TLV Result message.*

- int eap_tlv_process (struct eap_sm *sm, struct eap_method_ret *ret, const struct eap_hdr *hdr, u8 **resp, size_t *resp_len)

  *Process a received EAP-TLV message and generate a response.*

### 6.92.1 Detailed Description

EAP peer method: EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt).

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_tlv.c.

### 6.92.2 Function Documentation

#### 6.92.2.1 u8* eap_tlv_build_nak (int *id*, u16 *nak_type*, size_t * *resp_len*)

Build EAP-TLV NAK message.

**Parameters:**
***id*** EAP identifier for the header
***nak_type*** TLV type (EAP_TLV_*)
***resp_len*** Buffer for returning the response length

**Returns:**
Buffer to the allocated EAP-TLV NAK message or NULL on failure

This funtion builds an EAP-TLV NAK message. The caller is responsible for freeing the returned buffer.

Definition at line 34 of file eap_tlv.c.

Here is the call graph for this function:


#### 6.92.2.2 u8* eap_tlv_build_result (int *id*, u16 *status*, size_t * *resp_len*)

Build EAP-TLV Result message.

**Parameters:**
***id*** EAP identifier for the header

***status*** Status (EAP_TLV_RESULT_SUCCESS or EAP_TLV_RESULT_FAILURE)

***resp_len*** Buffer for returning the response length

**Returns:**

Buffer to the allocated EAP-TLV Result message or NULL on failure

This funtion builds an EAP-TLV Result message. The caller is responsible for freeing the returned buffer.

Definition at line 72 of file eap_tlv.c.

Here is the call graph for this function:


#### 6.92.2.3 int eap_tlv_process (struct eap_sm * *sm*, struct eap_method_ret * *ret*, const struct eap_hdr * *hdr*, u8 ** *resp*, size_t * *resp_len*)

Process a received EAP-TLV message and generate a response.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***ret*** Return values from EAP request validation and processing

***hdr*** EAP-TLV request to be processed. The caller must have validated that the buffer is large enough to contain full request (hdr->length bytes) and that the EAP type is EAP_TYPE_TLV.

***resp*** Buffer to return a pointer to the allocated response message. This field should be initialized to NULL before the call. The value will be updated if a response message is generated. The caller is responsible for freeing the allocated message.

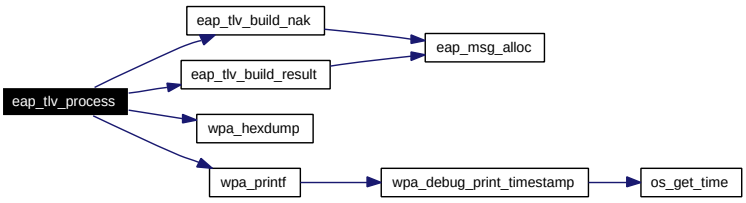
***resp_len*** Buffer for returning the response length

**Returns:**

0 on success, -1 on failure

Definition at line 109 of file eap_tlv.c.

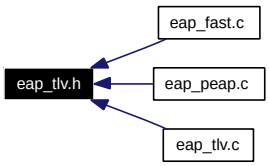
Here is the call graph for this function:



## 6.93 eap_tlv.h File Reference

EAP peer method: EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt).

This graph shows which files directly or indirectly include this file:




### Defines

- #define **EAP_TLV_RESULT_TLV** 3
- #define **EAP_TLV_NAK_TLV** 4
- #define **EAP_TLV_CRYPTO_BINDING_TLV** 5
- #define **EAP_TLV_CONNECTION_BINDING_TLV** 6
- #define **EAP_TLV_VENDOR_SPECIFIC_TLV** 7
- #define **EAP_TLV_URI_TLV** 8
- #define **EAP_TLV_EAP_PAYLOAD_TLV** 9
- #define **EAP_TLV_INTERMEDIATE_RESULT_TLV** 10
- #define **EAP_TLV_PAC_TLV** 11
- #define **EAP_TLV_CRYPTO_BINDING_TLV_** 12
- #define **EAP_TLV_RESULT_SUCCESS** 1
- #define **EAP_TLV_RESULT_FAILURE** 2
- #define **EAP_TLV_TYPE_MANDATORY** 0x8000
- #define **EAP_TLV_CRYPTO_BINDING_SUBTYPE_REQUEST** 0
- #define **EAP_TLV_CRYPTO_BINDING_SUBTYPE_RESPONSE** 1

### Functions

- u8 * eap_tlv_build_nak (int id, u16 nak_type, size_t *resp_len)

  *Build EAP-TLV NAK message.*

- u8 * eap_tlv_build_result (int id, u16 status, size_t *resp_len)

  *Build EAP-TLV Result message.*

- int eap_tlv_process (struct eap_sm *sm, struct eap_method_ret *ret, const struct eap_hdr *hdr, u8 **resp, size_t *resp_len)

  *Process a received EAP-TLV message and generate a response.*

### Variables

- eap_tlv_hdr **STRUCT_PACKED**

### 6.93.1 Detailed Description

EAP peer method: EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt).

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eap_tlv.h.

### 6.93.2 Function Documentation

#### 6.93.2.1 u8* eap_tlv_build_nak (int *id*, u16 *nak_type*, size_t * *resp_len*)

Build EAP-TLV NAK message.

**Parameters:**
***id*** EAP identifier for the header
***nak_type*** TLV type (EAP_TLV_*)
***resp_len*** Buffer for returning the response length

**Returns:**
Buffer to the allocated EAP-TLV NAK message or NULL on failure

This funtion builds an EAP-TLV NAK message. The caller is responsible for freeing the returned buffer.

Definition at line 34 of file eap_tlv.c.

Here is the call graph for this function:


#### 6.93.2.2 u8* eap_tlv_build_result (int *id*, u16 *status*, size_t * *resp_len*)

Build EAP-TLV Result message.

**Parameters:**
***id*** EAP identifier for the header
***status*** Status (EAP_TLV_RESULT_SUCCESS or EAP_TLV_RESULT_FAILURE)
***resp_len*** Buffer for returning the response length

**Returns:**
Buffer to the allocated EAP-TLV Result message or NULL on failure

This funtion builds an EAP-TLV Result message. The caller is responsible for freeing the returned buffer.

Definition at line 72 of file eap_tlv.c.

Here is the call graph for this function:


#### 6.93.2.3 int eap_tlv_process (struct eap_sm * *sm*, struct eap_method_ret * *ret*, const struct eap_hdr * *hdr*, u8 ** *resp*, size_t * *resp_len*)

Process a received EAP-TLV message and generate a response.

**Parameters:**

***sm*** Pointer to EAP state machine allocated with eap_sm_init()

***ret*** Return values from EAP request validation and processing

***hdr*** EAP-TLV request to be processed. The caller must have validated that the buffer is large enough to contain full request (hdr->length bytes) and that the EAP type is EAP_TYPE_TLV.

***resp*** Buffer to return a pointer to the allocated response message. This field should be initialized to NULL before the call. The value will be updated if a response message is generated. The caller is responsible for freeing the allocated message.

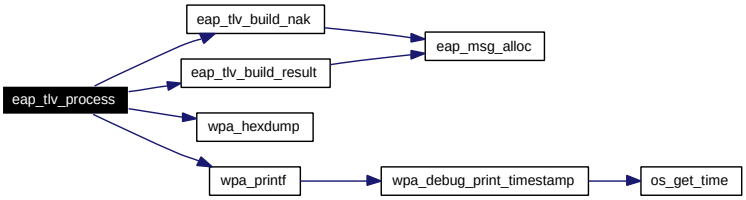
***resp_len*** Buffer for returning the response length

**Returns:**

0 on success, -1 on failure

Definition at line 109 of file eap_tlv.c.

Here is the call graph for this function:

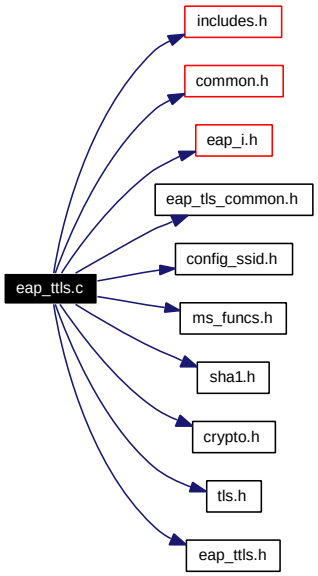


## 6.94 eap_ttls.c File Reference

EAP peer method: EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt).

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "config_ssid.h"
#include "ms_funcs.h"
#include "sha1.h"
#include "crypto.h"
#include "tls.h"
#include "eap_ttls.h"
```

Include dependency graph for eap_ttls.c:




### Defines

- #define **EAP_TTLS_VERSION** 0
- #define **MSCHAPV2_KEY_LEN** 16

### Functions

- int **eap_peer_ttls_register** (void)

### 6.94.1 Detailed Description

EAP peer method: EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt).

**Copyright**



Definition in file eap_ttls.c.

# 6.95 eap_ttls.h File Reference

EAP server/peer: EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt).

This graph shows which files directly or indirectly include this file:


## Defines

- #define **AVP_FLAGS_VENDOR** 0x80
- #define **AVP_FLAGS_MANDATORY** 0x40
- #define **AVP_PAD**(start, pos)
- #define **RADIUS_ATTR_USER_NAME** 1
- #define **RADIUS_ATTR_USER_PASSWORD** 2
- #define **RADIUS_ATTR_CHAP_PASSWORD** 3
- #define **RADIUS_ATTR_REPLY_MESSAGE** 18
- #define **RADIUS_ATTR_CHAP_CHALLENGE** 60
- #define **RADIUS_ATTR_EAP_MESSAGE** 79
- #define **RADIUS_VENDOR_ID_MICROSOFT** 311
- #define **RADIUS_ATTR_MS_CHAP_RESPONSE** 1
- #define **RADIUS_ATTR_MS_CHAP_ERROR** 2
- #define **RADIUS_ATTR_MS_CHAP_NT_ENC_PW** 6
- #define **RADIUS_ATTR_MS_CHAP_CHALLENGE** 11
- #define **RADIUS_ATTR_MS_CHAP2_RESPONSE** 25
- #define **RADIUS_ATTR_MS_CHAP2_SUCCESS** 26
- #define **RADIUS_ATTR_MS_CHAP2_CPW** 27
- #define **EAP_TTLS_MSCHAPV2_CHALLENGE_LEN** 16
- #define **EAP_TTLS_MSCHAPV2_RESPONSE_LEN** 50
- #define **EAP_TTLS_MSCHAP_CHALLENGE_LEN** 8
- #define **EAP_TTLS_MSCHAP_RESPONSE_LEN** 50
- #define **EAP_TTLS_CHAP_CHALLENGE_LEN** 16
- #define **EAP_TTLS_CHAP_PASSWORD_LEN** 16

### 6.95.1 Detailed Description

EAP server/peer: EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt).

**Copyright**



Definition in file eap_ttls.h.

## 6.95.2 Define Documentation

### 6.95.2.1 #define AVP_PAD(start, pos)

**Value:**

```
do { \
        int __pad; \
        __pad = (4 - (((pos) - (start)) & 3)) & 3; \
        os_memset((pos), 0, __pad); \
        pos += __pad; \
} while (0)
```

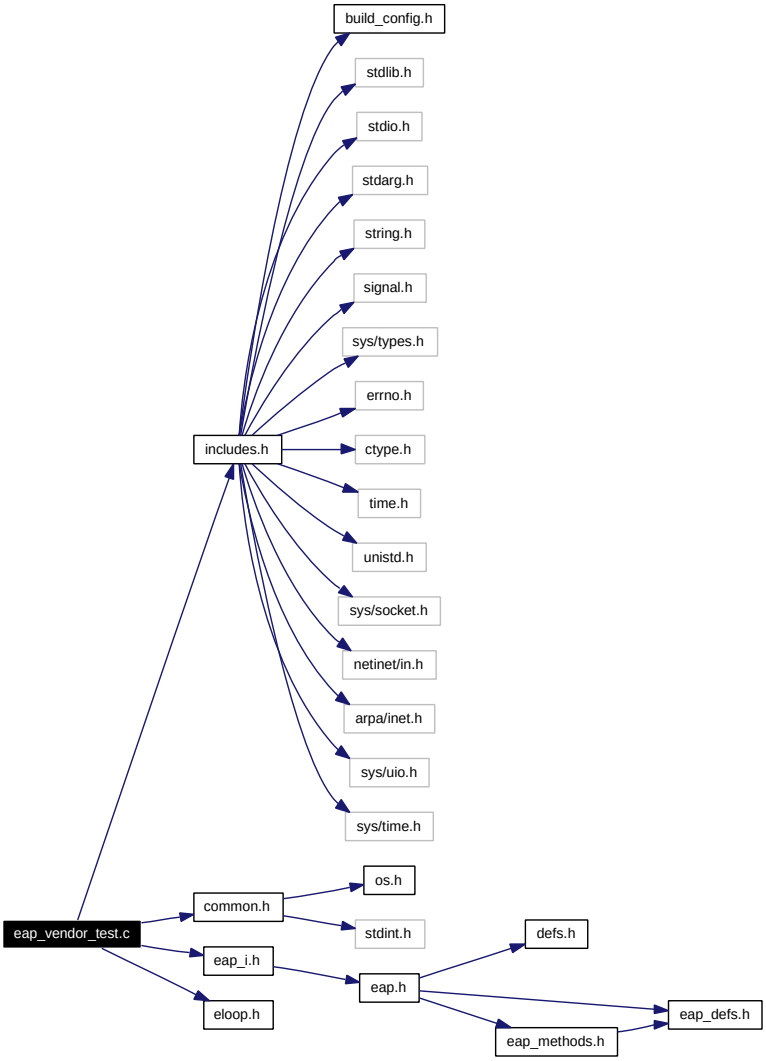
Definition at line 38 of file eap_ttls.h.

# 6.96 eap_vendor_test.c File Reference

EAP peer method: Test method for vendor specific (expanded) EAP type.

```
#include "includes.h"
#include "common.h"
#include "eap_i.h"
#include "eloop.h"
```

Include dependency graph for eap_vendor_test.c:




## Defines

- #define **EAP_VENDOR_ID** 0xfffefd
- #define **EAP_VENDOR_TYPE** 0xfcfbfaf9

## Functions

- int **eap_peer_vendor_test_register** (void)

### 6.96.1 Detailed Description

EAP peer method: Test method for vendor specific (expanded) EAP type.

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements a vendor specific test method using EAP expanded types. This is only for test use and must not be used for authentication since no security is provided.

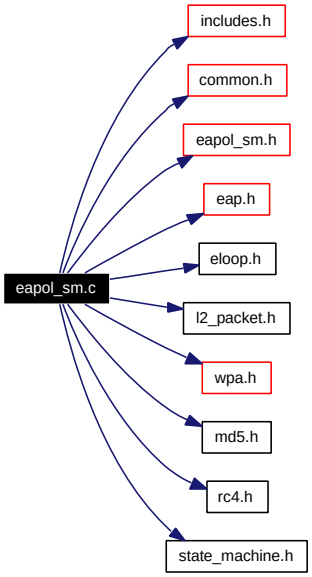
Definition in file eap_vendor_test.c.

# 6.97 eapol_sm.c File Reference

WPA Supplicant / EAPOL state machines.

```
#include "includes.h"
#include "common.h"
#include "eapol_sm.h"
#include "eap.h"
#include "eloop.h"
#include "l2_packet.h"
#include "wpa.h"
#include "md5.h"
#include "rc4.h"
#include "state_machine.h"
```

Include dependency graph for eapol_sm.c:




## Defines

- #define **STATE_MACHINE_DATA** struct eapol_sm
- #define **STATE_MACHINE_DEBUG_PREFIX** "EAPOL"
- #define **IEEE8021X_REPLAY_COUNTER_LEN** 8
- #define **IEEE8021X_KEY_SIGN_LEN** 16
- #define **IEEE8021X_KEY_IV_LEN** 16
- #define **IEEE8021X_KEY_INDEX_FLAG** 0x80
- #define **IEEE8021X_KEY_INDEX_MASK** 0x03
- #define **IEEE8021X_ENCR_KEY_LEN** 32
- #define **IEEE8021X_SIGN_KEY_LEN** 32

## Functions

- **SM_STATE** (SUPP_PAE, LOGOFF)
- **SM_STATE** (SUPP_PAE, DISCONNECTED)
- **SM_STATE** (SUPP_PAE, CONNECTING)
- **SM_STATE** (SUPP_PAE, AUTHENTICATING)
- **SM_STATE** (SUPP_PAE, HELD)
- **SM_STATE** (SUPP_PAE, AUTHENTICATED)
- **SM_STATE** (SUPP_PAE, RESTART)
- **SM_STATE** (SUPP_PAE, S_FORCE_AUTH)
- **SM_STATE** (SUPP_PAE, S_FORCE_UNAUTH)
- **SM_STEP** (SUPP_PAE)
- **SM_STATE** (KEY_RX, NO_KEY_RECEIVE)
- **SM_STATE** (KEY_RX, KEY_RECEIVE)
- **SM_STEP** (KEY_RX)
- **SM_STATE** (SUPP_BE, REQUEST)
- **SM_STATE** (SUPP_BE, RESPONSE)
- **SM_STATE** (SUPP_BE, SUCCESS)
- **SM_STATE** (SUPP_BE, FAIL)
- **SM_STATE** (SUPP_BE, TIMEOUT)
- **SM_STATE** (SUPP_BE, IDLE)
- **SM_STATE** (SUPP_BE, INITIALIZE)
- **SM_STATE** (SUPP_BE, RECEIVE)
- **SM_STEP** (SUPP_BE)
- void eapol_sm_step (struct eapol_sm *sm)

  *EAPOL state machine step function.*

- void eapol_sm_configure (struct eapol_sm *sm, int heldPeriod, int authPeriod, int startPeriod, int maxStart)

  *Set EAPOL variables.*

- int eapol_sm_get_status (struct eapol_sm *sm, char *buf, size_t buflen, int verbose)

  *Get EAPOL state machine status.*

- int eapol_sm_get_mib (struct eapol_sm *sm, char *buf, size_t buflen)

  *Get EAPOL state machine MIBs.*

- int eapol_sm_rx_eapol (struct eapol_sm *sm, const u8 *src, const u8 *buf, size_t len)

  *Process received EAPOL frames.*

- void eapol_sm_notify_tx_eapol_key (struct eapol_sm *sm)

  *Notification about transmitted EAPOL packet.*

- void eapol_sm_notify_portEnabled (struct eapol_sm *sm, Boolean enabled)

  *Notification about portEnabled change.*

- void eapol_sm_notify_portValid (struct eapol_sm *sm, Boolean valid)

  *Notification about portValid change.*

- void eapol_sm_notify_eap_success (struct eapol_sm *sm, Boolean success)

*Notification of external EAP success trigger.*

- void eapol_sm_notify_eap_fail (struct eapol_sm *sm, Boolean fail)

  *Notification of external EAP failure trigger.*

- void eapol_sm_notify_config (struct eapol_sm *sm, struct wpa_ssid *config, const struct eapol_config *conf)

  *Notification of EAPOL configuration change.*

- int eapol_sm_get_key (struct eapol_sm *sm, u8 *key, size_t len)

  *Get master session key (MSK) from EAP.*

- void eapol_sm_notify_logoff (struct eapol_sm *sm, Boolean logoff)

  *Notification of logon/logoff commands.*

- void eapol_sm_notify_cached (struct eapol_sm *sm)

  *Notification of successful PMKSA caching.*

- void eapol_sm_notify_pmkid_attempt (struct eapol_sm *sm, int attempt)

  *Notification of PMKSA caching.*

- void eapol_sm_register_scard_ctx (struct eapol_sm *sm, void *ctx)

  *Notification of smart card context.*

- void eapol_sm_notify_portControl (struct eapol_sm *sm, PortControl portControl)

  *Notification of portControl changes.*

- void eapol_sm_notify_ctrl_attached (struct eapol_sm *sm)

  *Notification of attached monitor.*

- void eapol_sm_notify_ctrl_response (struct eapol_sm *sm)

  *Notification of received user input.*

- void eapol_sm_request_reauth (struct eapol_sm *sm)

  *Request reauthentication.*

- void eapol_sm_notify_lower_layer_success (struct eapol_sm *sm)

  *Notification of lower layer success.*

- void eapol_sm_invalidate_cached_session (struct eapol_sm *sm)

  *Mark cached EAP session data invalid.*

- eapol_sm * eapol_sm_init (struct eapol_ctx *ctx)

  *Initialize EAPOL state machine.*

- void eapol_sm_deinit (struct eapol_sm *sm)

  *Deinitialize EAPOL state machine.*

## Variables

- ieee802_1x_eapol_key **STRUCT_PACKED**

### 6.97.1 Detailed Description

WPA Supplicant / EAPOL state machines.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eapol_sm.c.

### 6.97.2 Function Documentation

#### 6.97.2.1 void eapol_sm_configure (struct eapol_sm * *sm*, int *heldPeriod*, int *authPeriod*, int *startPeriod*, int *maxStart*)

Set EAPOL variables.

**Parameters:**
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***heldPeriod*** dot1xSuppHeldPeriod

***authPeriod*** dot1xSuppAuthPeriod

***startPeriod*** dot1xSuppStartPeriod

***maxStart*** dot1xSuppMaxStart

Set configurable EAPOL state machine variables. Each variable can be set to the given value or ignored if set to -1 (to set only some of the variables).

Definition at line 973 of file eapol_sm.c.

#### 6.97.2.2 void eapol_sm_deinit (struct eapol_sm * *sm*)

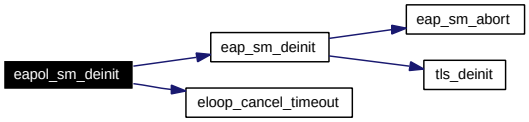
Deinitialize EAPOL state machine.

**Parameters:**
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Deinitialize and free EAPOL state machine.

Definition at line 1801 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.3 int eapol_sm_get_key (struct eapol_sm * *sm*, u8 * *key*, size_t *len*)

Get master session key (MSK) from EAP.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***key*** Pointer for key buffer

***len*** Number of bytes to copy to key

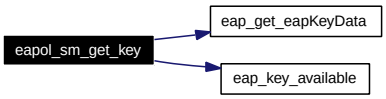
**Returns:**

0 on success (len of key available), maximum available key len (>0) if key is available but it is shorter than len, or -1 on failure.

Fetch EAP keying material (MSK, eapKeyData) from EAP state machine. The key is available only after a successful authentication.

Definition at line 1367 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.4 int eapol_sm_get_mib (struct eapol_sm * *sm*, char * *buf*, size_t *buflen*)

Get EAPOL state machine MIBs.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***buf*** Buffer for MIB information

***buflen*** Maximum buffer length

**Returns:**

Number of bytes written to buf.

Query EAPOL state machine for MIB information. This function fills in a text area with current MIB information from the EAPOL state machine. If the buffer (buf) is not large enough, MIB information will be truncated to fit the buffer.

Definition at line 1057 of file eapol_sm.c.

#### 6.97.2.5 int eapol_sm_get_status (struct eapol_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get EAPOL state machine status.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

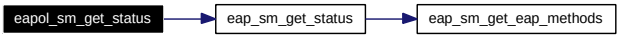
**Returns:**

Number of bytes written to buf.

Query EAPOL state machine for status information. This function fills in a text area with current status information from the EAPOL state machine. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

Definition at line 1004 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.6 struct eapol_sm* eapol_sm_init (struct eapol_ctx * *ctx*)

Initialize EAPOL state machine.

**Parameters:**

***ctx*** Pointer to EAPOL context data; this needs to be an allocated buffer and EAPOL state machine will free it in eapol_sm_deinit()

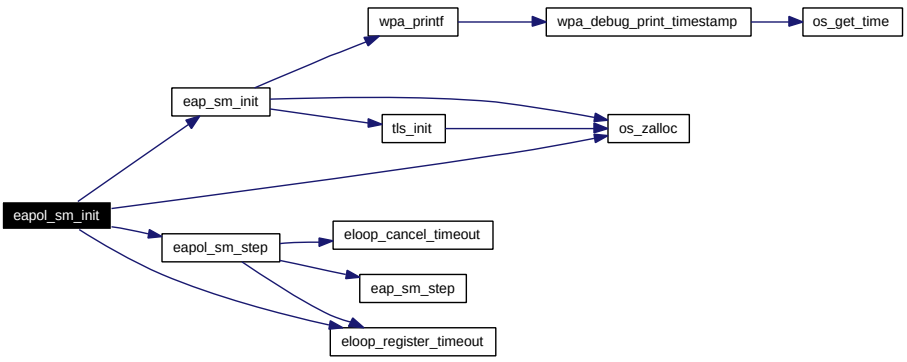
**Returns:**

Pointer to the allocated EAPOL state machine or NULL on failure

Allocate and initialize an EAPOL state machine.

Definition at line 1752 of file eapol_sm.c.

Here is the call graph for this function:



#### 6.97.2.7 void eapol_sm_invalidate_cached_session (struct eapol_sm * *sm*)

Mark cached EAP session data invalid.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Definition at line 1574 of file eapol_sm.c.

Here is the call graph for this function:


#### 6.97.2.8 void eapol_sm_notify_cached (struct eapol_sm * *sm*)

Notification of successful PMKSA caching.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that PMKSA caching was successful. This is used to move EAPOL and EAP state machines into authenticated/successful state.

Definition at line 1409 of file eapol_sm.c.

Here is the call graph for this function:


#### 6.97.2.9 void eapol_sm_notify_config (struct eapol_sm * *sm*, struct wpa_ssid * *config*, const struct eapol_config * *conf*)

Notification of EAPOL configuration change.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

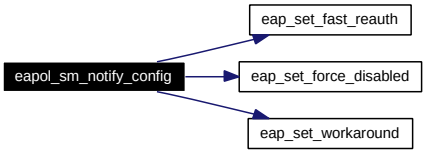
***config*** Pointer to current network configuration

***conf*** Pointer to EAPOL configuration data

Notify EAPOL station machine that configuration has changed. config will be stored as a backpointer to network configuration. This can be NULL to clear the stored pointed. conf will be copied to local EAPOL/EAP configuration data. If conf is NULL, this part of the configuration change will be skipped.

Definition at line 1333 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.10 void eapol_sm_notify_ctrl_attached (struct eapol_sm * *sm*)

Notification of attached monitor.

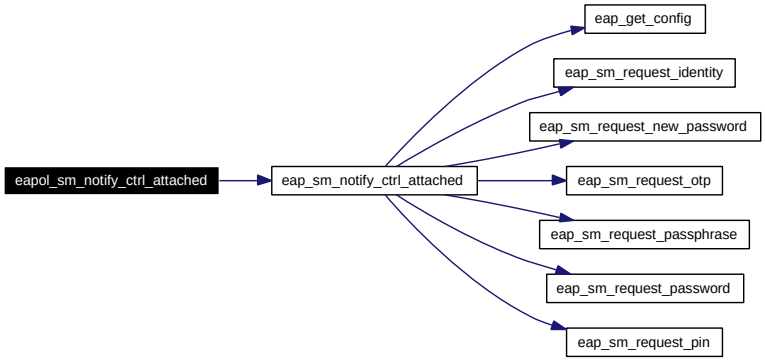
**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that a monitor was attached to the control interface to trigger re-sending of pending requests for user input.

Definition at line 1505 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.11 void eapol_sm_notify_ctrl_response (struct eapol_sm * *sm*)

Notification of received user input.

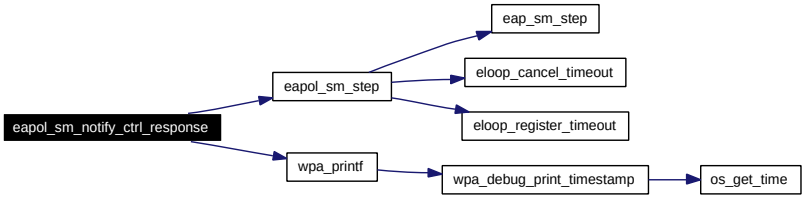
**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that a control response, i.e., user input, was received in order to trigger retrying of a pending EAP request.

Definition at line 1521 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.12 void eapol_sm_notify_eap_fail (struct eapol_sm * *sm*, Boolean *fail*)

Notification of external EAP failure trigger.

**Parameters:**

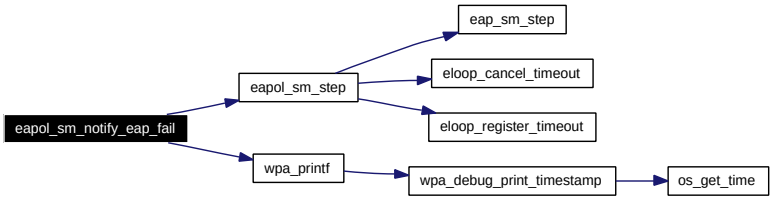
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***fail*** TRUE = set failure, FALSE = clear failure

Notify EAPOL station machine that external event has forced EAP state to failure (fail = TRUE). This can be cleared by setting fail = FALSE.

Definition at line 1308 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.13 void eapol_sm_notify_eap_success (struct eapol_sm * *sm*, Boolean *success*)

Notification of external EAP success trigger.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

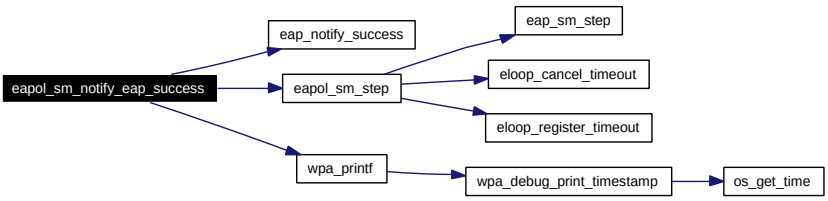
***success*** TRUE = set success, FALSE = clear success

Notify EAPOL station machine that external event has forced EAP state to success (success = TRUE). This can be cleared by setting success = FALSE.

This function is called to update EAP state when WPA-PSK key handshake has been completed successfully since WPA-PSK does not use EAP state machine.

Definition at line 1285 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.14 void eapol_sm_notify_logoff (struct eapol_sm * *sm*, Boolean *logoff*)

Notification of logon/logoff commands.

**Parameters:**

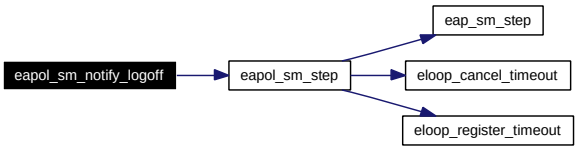
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***logoff*** Whether command was logoff

Notify EAPOL state machines that user requested logon/logoff.

Definition at line 1392 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.97.2.15 void eapol_sm_notify_lower_layer_success (struct eapol_sm * *sm*)

Notification of lower layer success.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL (and EAP) state machines that a lower layer has detected a successful authentication. This is used to recover from dropped EAP-Success messages.

Definition at line 1561 of file eapol_sm.c.

Here is the call graph for this function:


#### 6.97.2.16 void eapol_sm_notify_pmkid_attempt (struct eapol_sm * *sm*, int *attempt*)

Notification of PMKSA caching.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***attempt*** Whether PMKSA caching is tried

Notify EAPOL state machines whether PMKSA caching is used.

Definition at line 1427 of file eapol_sm.c.

Here is the call graph for this function:


### 6.97.2.17 void eapol_sm_notify_portControl (struct eapol_sm * *sm*, PortControl *portControl*)

Notification of portControl changes.

**Parameters:**

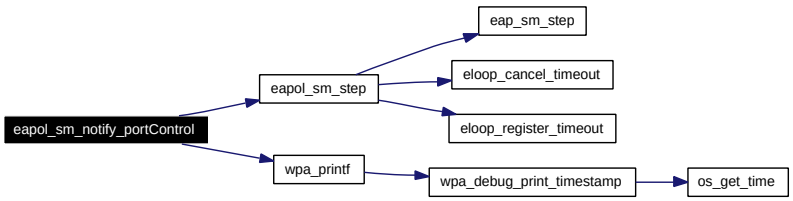
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***portControl*** New value for portControl variable

Notify EAPOL state machines that portControl variable has changed.

Definition at line 1486 of file eapol_sm.c.

Here is the call graph for this function:




### 6.97.2.18 void eapol_sm_notify_portEnabled (struct eapol_sm * *sm*, Boolean *enabled*)

Notification about portEnabled change.

**Parameters:**

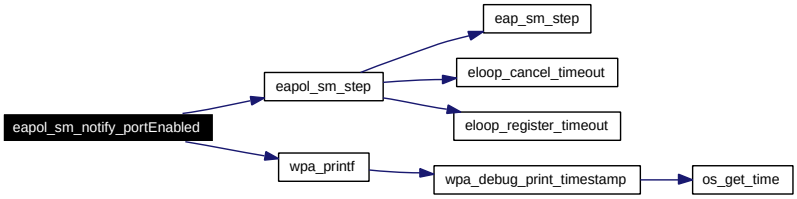
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***enabled*** New portEnabled value

Notify EAPOL station machine about new portEnabled value.

Definition at line 1243 of file eapol_sm.c.

Here is the call graph for this function:




### 6.97.2.19 void eapol_sm_notify_portValid (struct eapol_sm * *sm*, Boolean *valid*)

Notification about portValid change.

**Parameters:**

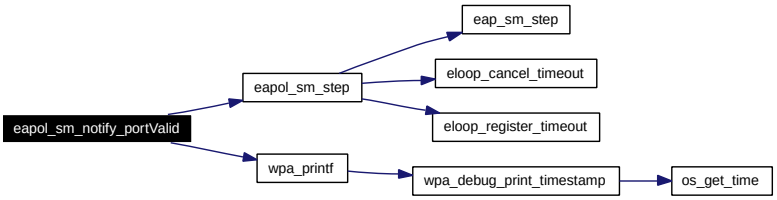
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***valid*** New portValid value

Notify EAPOL station machine about new portValid value.

Definition at line 1262 of file eapol_sm.c.

Here is the call graph for this function:




### 6.97.2.20 void eapol_sm_notify_tx_eapol_key (struct eapol_sm * *sm*)

Notification about transmitted EAPOL packet.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL station machine about transmitted EAPOL packet from an external component, e.g., WPA. This will update the statistics.

Definition at line 1228 of file eapol_sm.c.

### 6.97.2.21 void eapol_sm_register_scard_ctx (struct eapol_sm * *sm*, void * *ctx*)

Notification of smart card context.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***ctx*** Context data for smart card operations

Notify EAPOL state machines of context data for smart card operations. This context data will be used as a parameter for scard_∗() functions.

Definition at line 1469 of file eapol_sm.c.

Here is the call graph for this function:


#### 6.97.2.22 void eapol_sm_request_reauth (struct eapol_sm ∗ *sm*)

Request reauthentication.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

This function can be used to request EAPOL reauthentication, e.g., when the current PMKSA entry is nearing expiration.

Definition at line 1544 of file eapol_sm.c.

#### 6.97.2.23 int eapol_sm_rx_eapol (struct eapol_sm ∗ *sm*, const u8 ∗ *src*, const u8 ∗ *buf*, size_t *len*)

Process received EAPOL frames.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***src*** Source MAC address of the EAPOL packet

***buf*** Pointer to the beginning of the EAPOL data (EAPOL header)

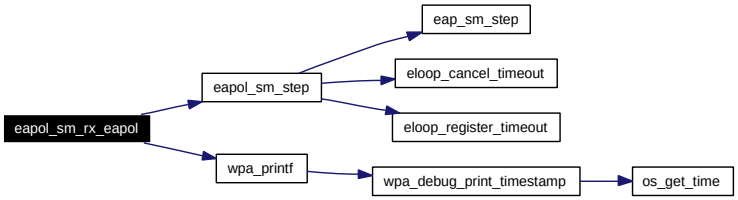
***len*** Length of the EAPOL frame

**Returns:**

1 = EAPOL frame processed, 0 = not for EAPOL state machine, -1 failure

Definition at line 1128 of file eapol_sm.c.

Here is the call graph for this function:



#### 6.97.2.24 void eapol_sm_step (struct eapol_sm * *sm*)

EAPOL state machine step function.

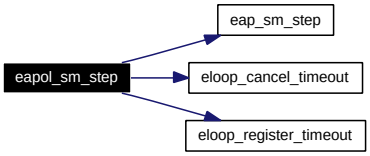
**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

This function is called to notify the state machine about changed external variables. It will step through the EAPOL state machines in loop to process all triggered state changes.

Definition at line 851 of file eapol_sm.c.

Here is the call graph for this function:



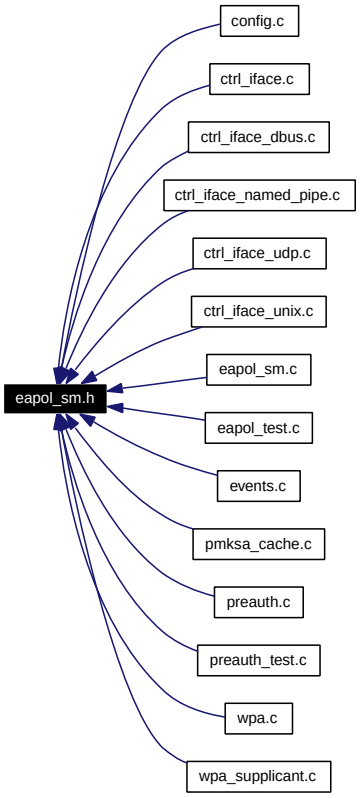
# 6.98 eapol_sm.h File Reference

WPA Supplicant / EAPOL state machines.

```
#include "defs.h"
```

Include dependency graph for eapol_sm.h:


This graph shows which files directly or indirectly include this file:




## Defines

- #define **EAPOL_REQUIRE_KEY_UNICAST** BIT(0)
- #define **EAPOL_REQUIRE_KEY_BROADCAST** BIT(1)

## Enumerations

- enum **PortStatus** { **Unauthorized**, **Authorized** }
- enum **PortControl** { **Auto**, **ForceUnauthorized**, **ForceAuthorized** }

## Functions

- eapol_sm * eapol_sm_init (struct eapol_ctx *ctx)

  *Initialize EAPOL state machine.*

- void eapol_sm_deinit (struct eapol_sm *sm)

  *Deinitialize EAPOL state machine.*

- void eapol_sm_step (struct eapol_sm *sm)

  *EAPOL state machine step function.*

- int eapol_sm_get_status (struct eapol_sm *sm, char *buf, size_t buflen, int verbose)

  *Get EAPOL state machine status.*

- int eapol_sm_get_mib (struct eapol_sm *sm, char *buf, size_t buflen)

  *Get EAPOL state machine MIBs.*

- void eapol_sm_configure (struct eapol_sm *sm, int heldPeriod, int authPeriod, int startPeriod, int maxStart)

  *Set EAPOL variables.*

- int eapol_sm_rx_eapol (struct eapol_sm *sm, const u8 *src, const u8 *buf, size_t len)

  *Process received EAPOL frames.*

- void eapol_sm_notify_tx_eapol_key (struct eapol_sm *sm)

  *Notification about transmitted EAPOL packet.*

- void eapol_sm_notify_portEnabled (struct eapol_sm *sm, Boolean enabled)

  *Notification about portEnabled change.*

- void eapol_sm_notify_portValid (struct eapol_sm *sm, Boolean valid)

  *Notification about portValid change.*

- void eapol_sm_notify_eap_success (struct eapol_sm *sm, Boolean success)

  *Notification of external EAP success trigger.*

- void eapol_sm_notify_eap_fail (struct eapol_sm *sm, Boolean fail)

  *Notification of external EAP failure trigger.*

- void eapol_sm_notify_config (struct eapol_sm *sm, struct wpa_ssid *config, const struct eapol_config *conf)

  *Notification of EAPOL configuration change.*

- int eapol_sm_get_key (struct eapol_sm *sm, u8 *key, size_t len)

  *Get master session key (MSK) from EAP.*

- void eapol_sm_notify_logoff (struct eapol_sm *sm, Boolean logoff)

  *Notification of logon/logoff commands.*

- void eapol_sm_notify_cached (struct eapol_sm *sm)

*Notification of successful PMKSA caching.*

- void eapol_sm_notify_pmkid_attempt (struct eapol_sm *sm, int attempt)
  *Notification of PMKSA caching.*
- void eapol_sm_register_scard_ctx (struct eapol_sm *sm, void *ctx)
  *Notification of smart card context.*
- void eapol_sm_notify_portControl (struct eapol_sm *sm, PortControl portControl)
  *Notification of portControl changes.*
- void eapol_sm_notify_ctrl_attached (struct eapol_sm *sm)
  *Notification of attached monitor.*
- void eapol_sm_notify_ctrl_response (struct eapol_sm *sm)
  *Notification of received user input.*
- void eapol_sm_request_reauth (struct eapol_sm *sm)
  *Request reauthentication.*
- void eapol_sm_notify_lower_layer_success (struct eapol_sm *sm)
  *Notification of lower layer success.*
- void eapol_sm_invalidate_cached_session (struct eapol_sm *sm)
  *Mark cached EAP session data invalid.*

## 6.98.1 Detailed Description

WPA Supplicant / EAPOL state machines.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eapol_sm.h.

## 6.98.2 Function Documentation

### 6.98.2.1 void eapol_sm_configure (struct eapol_sm * *sm*, int *heldPeriod*, int *authPeriod*, int *startPeriod*, int *maxStart*)

Set EAPOL variables.

**Parameters:**
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***heldPeriod*** dot1xSuppHeldPeriod

***authPeriod*** dot1xSuppAuthPeriod

***startPeriod*** dot1xSuppStartPeriod

***maxStart*** dot1xSuppMaxStart

Set configurable EAPOL state machine variables. Each variable can be set to the given value or ignored if set to -1 (to set only some of the variables).

Definition at line 973 of file eapol_sm.c.

#### 6.98.2.2 void eapol_sm_deinit (struct eapol_sm * *sm*)

Deinitialize EAPOL state machine.

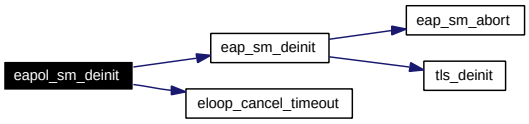
**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Deinitialize and free EAPOL state machine.

Definition at line 1801 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.98.2.3 int eapol_sm_get_key (struct eapol_sm * *sm*, u8 * *key*, size_t *len*)

Get master session key (MSK) from EAP.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***key*** Pointer for key buffer

***len*** Number of bytes to copy to key

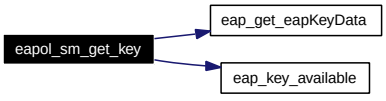
**Returns:**

0 on success (len of key available), maximum available key len (>0) if key is available but it is shorter than len, or -1 on failure.

Fetch EAP keying material (MSK, eapKeyData) from EAP state machine. The key is available only after a successful authentication.

Definition at line 1367 of file eapol_sm.c.

Here is the call graph for this function:



#### 6.98.2.4 int eapol_sm_get_mib (struct eapol_sm * *sm*, char * *buf*, size_t *buflen*)

Get EAPOL state machine MIBs.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***buf*** Buffer for MIB information

***buflen*** Maximum buffer length

**Returns:**

Number of bytes written to buf.

Query EAPOL state machine for MIB information. This function fills in a text area with current MIB information from the EAPOL state machine. If the buffer (buf) is not large enough, MIB information will be truncated to fit the buffer.

Definition at line 1057 of file eapol_sm.c.

#### 6.98.2.5 int eapol_sm_get_status (struct eapol_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get EAPOL state machine status.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

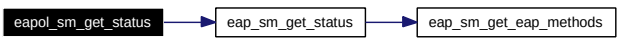
**Returns:**

Number of bytes written to buf.

Query EAPOL state machine for status information. This function fills in a text area with current status information from the EAPOL state machine. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

Definition at line 1004 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.98.2.6 struct eapol_sm* eapol_sm_init (struct eapol_ctx * *ctx*)

Initialize EAPOL state machine.

**Parameters:**

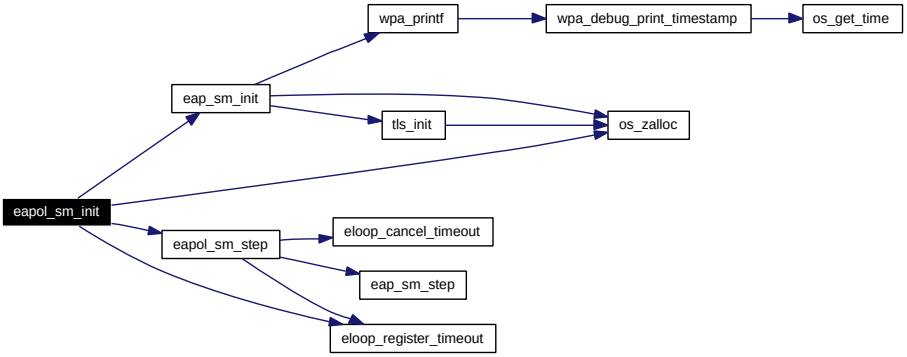
***ctx*** Pointer to EAPOL context data; this needs to be an allocated buffer and EAPOL state machine will free it in eapol_sm_deinit()

**Returns:**
Pointer to the allocated EAPOL state machine or NULL on failure

Allocate and initialize an EAPOL state machine.

Definition at line 1752 of file eapol_sm.c.

Here is the call graph for this function:




### 6.98.2.7 void eapol_sm_invalidate_cached_session (struct eapol_sm * *sm*)

Mark cached EAP session data invalid.

**Parameters:**
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Definition at line 1574 of file eapol_sm.c.

Here is the call graph for this function:


### 6.98.2.8 void eapol_sm_notify_cached (struct eapol_sm * *sm*)

Notification of successful PMKSA caching.

**Parameters:**
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that PMKSA caching was successful. This is used to move EAPOL and EAP state machines into authenticated/successful state.

Definition at line 1409 of file eapol_sm.c.

Here is the call graph for this function:

#### 6.98.2.9 void eapol_sm_notify_config (struct eapol_sm * *sm*, struct wpa_ssid * *config*, const struct eapol_config * *conf*)

Notification of EAPOL configuration change.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

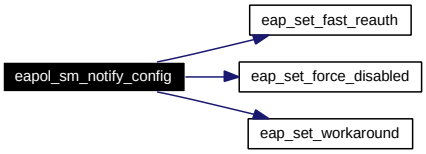
***config*** Pointer to current network configuration

***conf*** Pointer to EAPOL configuration data

Notify EAPOL station machine that configuration has changed. config will be stored as a backpointer to network configuration. This can be NULL to clear the stored pointed. conf will be copied to local EAPOL/EAP configuration data. If conf is NULL, this part of the configuration change will be skipped.

Definition at line 1333 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.98.2.10 void eapol_sm_notify_ctrl_attached (struct eapol_sm * *sm*)

Notification of attached monitor.

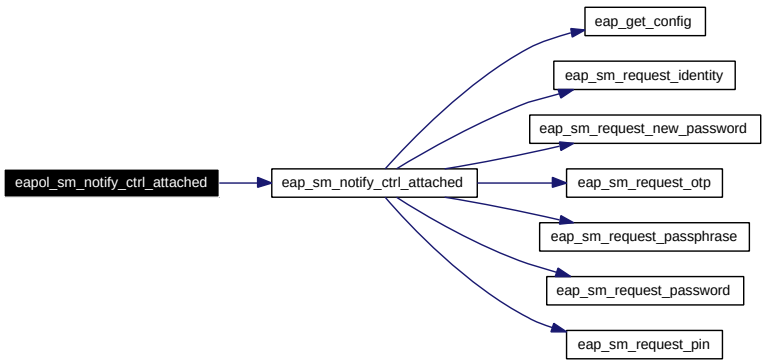
**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that a monitor was attached to the control interface to trigger re-sending of pending requests for user input.

Definition at line 1505 of file eapol_sm.c.

Here is the call graph for this function:



#### 6.98.2.11 void eapol_sm_notify_ctrl_response (struct eapol_sm * *sm*)

Notification of received user input.

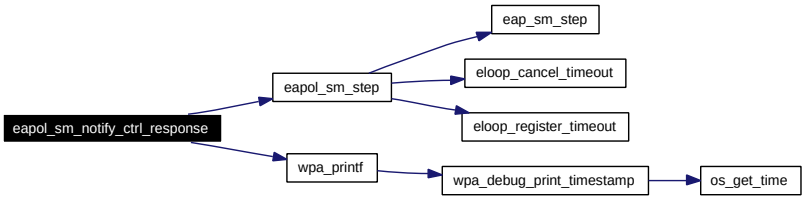
**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that a control response, i.e., user input, was received in order to trigger retrying of a pending EAP request.

Definition at line 1521 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.98.2.12 void eapol_sm_notify_eap_fail (struct eapol_sm * *sm*, Boolean *fail*)

Notification of external EAP failure trigger.

**Parameters:**

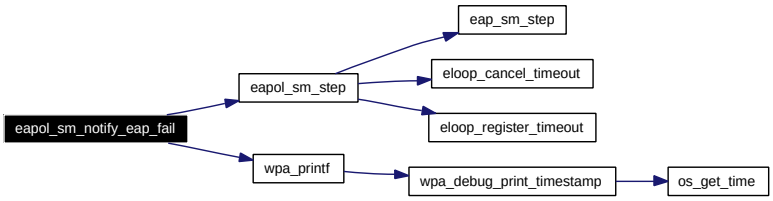
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***fail*** TRUE = set failure, FALSE = clear failure

Notify EAPOL station machine that external event has forced EAP state to failure (fail = TRUE). This can be cleared by setting fail = FALSE.

Definition at line 1308 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.98.2.13 void eapol_sm_notify_eap_success (struct eapol_sm * *sm*, Boolean *success*)

Notification of external EAP success trigger.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

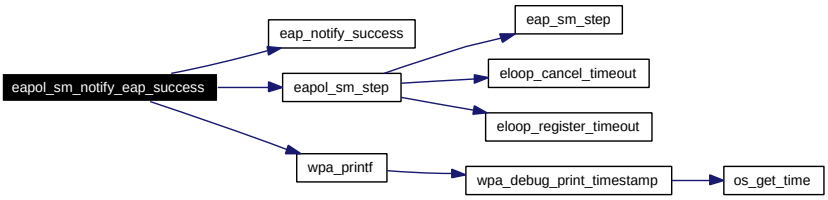
***success*** TRUE = set success, FALSE = clear success

Notify EAPOL station machine that external event has forced EAP state to success (success = TRUE). This can be cleared by setting success = FALSE.

This function is called to update EAP state when WPA-PSK key handshake has been completed successfully since WPA-PSK does not use EAP state machine.

Definition at line 1285 of file eapol_sm.c.

Here is the call graph for this function:




### 6.98.2.14 void eapol_sm_notify_logoff (struct eapol_sm * *sm*, Boolean *logoff*)

Notification of logon/logoff commands.

**Parameters:**

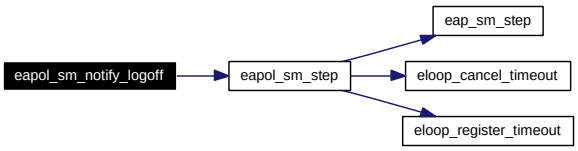
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***logoff*** Whether command was logoff

Notify EAPOL state machines that user requested logon/logoff.

Definition at line 1392 of file eapol_sm.c.

Here is the call graph for this function:




### 6.98.2.15 void eapol_sm_notify_lower_layer_success (struct eapol_sm * *sm*)

Notification of lower layer success.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL (and EAP) state machines that a lower layer has detected a successful authentication. This is used to recover from dropped EAP-Success messages.

Definition at line 1561 of file eapol_sm.c.

Here is the call graph for this function:


#### 6.98.2.16 void eapol_sm_notify_pmkid_attempt (struct eapol_sm * *sm*, int *attempt*)

Notification of PMKSA caching.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***attempt*** Whether PMKSA caching is tried

Notify EAPOL state machines whether PMKSA caching is used.

Definition at line 1427 of file eapol_sm.c.

Here is the call graph for this function:


#### 6.98.2.17 void eapol_sm_notify_portControl (struct eapol_sm * *sm*, PortControl *portControl*)

Notification of portControl changes.

**Parameters:**

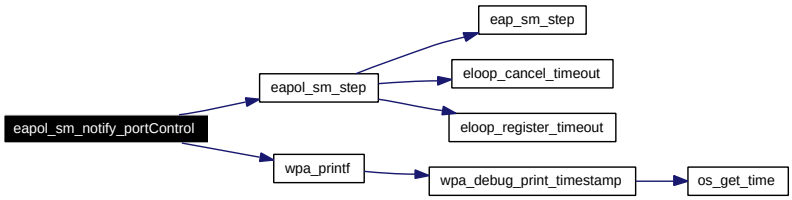
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***portControl*** New value for portControl variable

Notify EAPOL state machines that portControl variable has changed.

Definition at line 1486 of file eapol_sm.c.

Here is the call graph for this function:




#### 6.98.2.18 void eapol_sm_notify_portEnabled (struct eapol_sm * *sm*, Boolean *enabled*)

Notification about portEnabled change.

**Parameters:**

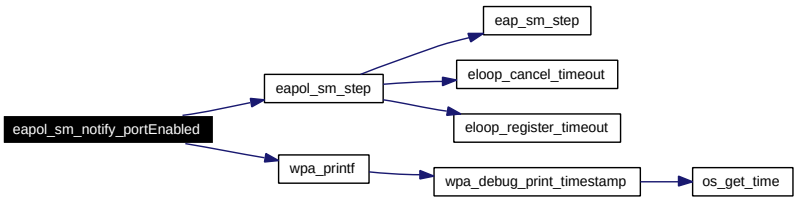
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***enabled*** New portEnabled value

Notify EAPOL station machine about new portEnabled value.

Definition at line 1243 of file eapol_sm.c.

Here is the call graph for this function:




### 6.98.2.19 void eapol_sm_notify_portValid (struct eapol_sm * *sm*, Boolean *valid*)

Notification about portValid change.

**Parameters:**

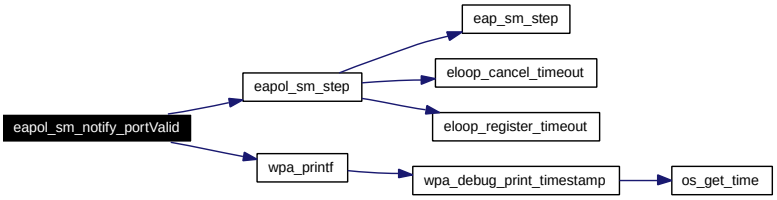
***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***valid*** New portValid value

Notify EAPOL station machine about new portValid value.

Definition at line 1262 of file eapol_sm.c.

Here is the call graph for this function:




### 6.98.2.20 void eapol_sm_notify_tx_eapol_key (struct eapol_sm * *sm*)

Notification about transmitted EAPOL packet.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL station machine about transmitted EAPOL packet from an external component, e.g., WPA. This will update the statistics.

Definition at line 1228 of file eapol_sm.c.

#### 6.98.2.21 void eapol_sm_register_scard_ctx (struct eapol_sm * *sm*, void * *ctx*)

Notification of smart card context.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***ctx*** Context data for smart card operations

Notify EAPOL state machines of context data for smart card operations. This context data will be used as a parameter for scard_*() functions.

Definition at line 1469 of file eapol_sm.c.

Here is the call graph for this function:


#### 6.98.2.22 void eapol_sm_request_reauth (struct eapol_sm * *sm*)

Request reauthentication.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

This function can be used to request EAPOL reauthentication, e.g., when the current PMKSA entry is nearing expiration.

Definition at line 1544 of file eapol_sm.c.

#### 6.98.2.23 int eapol_sm_rx_eapol (struct eapol_sm * *sm*, const u8 * *src*, const u8 * *buf*, size_t *len*)

Process received EAPOL frames.

**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

***src*** Source MAC address of the EAPOL packet

***buf*** Pointer to the beginning of the EAPOL data (EAPOL header)

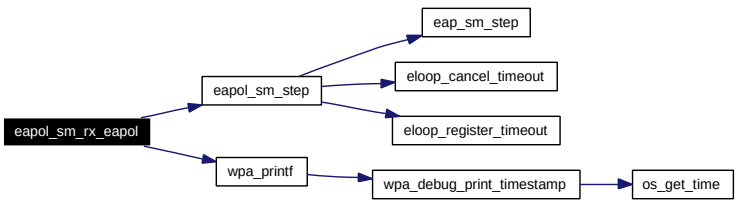
***len*** Length of the EAPOL frame

**Returns:**

1 = EAPOL frame processed, 0 = not for EAPOL state machine, -1 failure

Definition at line 1128 of file eapol_sm.c.

Here is the call graph for this function:



#### 6.98.2.24 void eapol_sm_step (struct eapol_sm * *sm*)

EAPOL state machine step function.

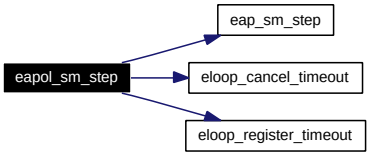
**Parameters:**

***sm*** Pointer to EAPOL state machine allocated with eapol_sm_init()

This function is called to notify the state machine about changed external variables. It will step through the EAPOL state machines in loop to process all triggered state changes.

Definition at line 851 of file eapol_sm.c.

Here is the call graph for this function:

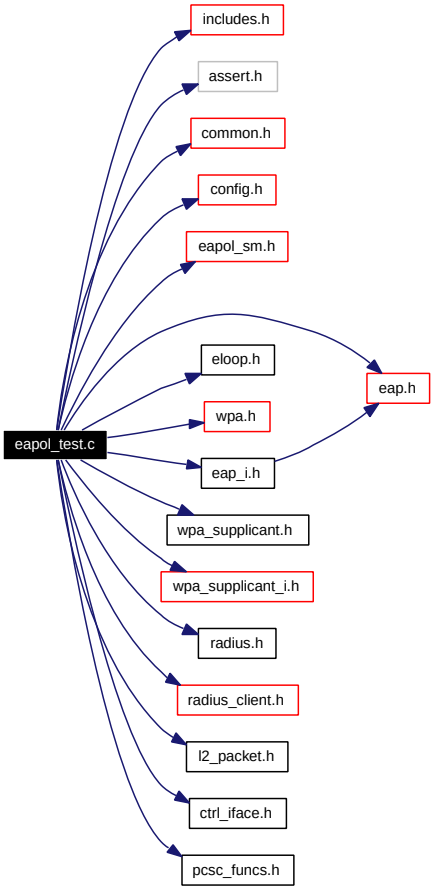


## 6.99 eapol_test.c File Reference

WPA Supplicant - test code.

```
#include "includes.h"
#include <assert.h>
#include "common.h"
#include "config.h"
#include "eapol_sm.h"
#include "eap.h"
#include "eloop.h"
#include "wpa.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "wpa_supplicant_i.h"
#include "radius.h"
#include "radius_client.h"
#include "l2_packet.h"
#include "ctrl_iface.h"
#include "pcsc_funcs.h"
```

Include dependency graph for eapol_test.c:




## Defines

- #define **num_triplets** 5
- #define **AKA_RAND_LEN** 16
- #define **AKA_AUTN_LEN** 16
- #define **AKA_AUTS_LEN** 14
- #define **RES_MAX_LEN** 16
- #define **IK_LEN** 16
- #define **CK_LEN** 16

## Functions

- void **hostapd_logger** (void ∗ctx, const u8 ∗addr, unsigned int module, int level, char ∗fmt,...)
- const char ∗ **hostapd_ip_txt** (const struct hostapd_ip_addr ∗addr, char ∗buf, size_t buflen)
- int **hostapd_ip_diff** (struct hostapd_ip_addr ∗a, struct hostapd_ip_addr ∗b)
- int **main** (int argc, char ∗argv[ ])

## Variables

- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- wpa_driver_ops ∗ **wpa_supplicant_drivers** [ ] = { NULL }

### 6.99.1 Detailed Description

WPA Supplicant - test code.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

IEEE 802.1X Supplicant test code (to be used in place of wpa_supplicant.c. Not used in production version.

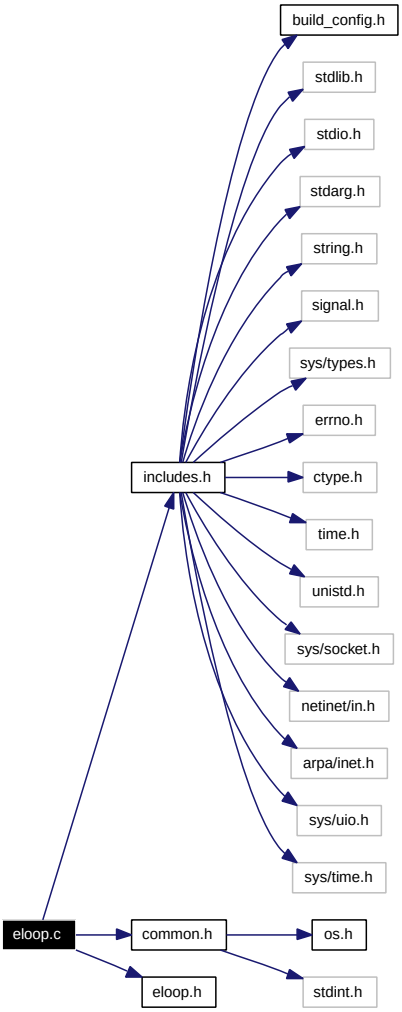
Definition in file eapol_test.c.

# 6.100 eloop.c File Reference

Event loop based on select() loop.

```
#include "includes.h"
#include "common.h"
#include "eloop.h"
```

Include dependency graph for eloop.c:




## Functions

- int eloop_init (void *user_data)

  *Initialize global event loop data.*

- int eloop_register_read_sock (int sock, eloop_sock_handler handler, void *eloop_data, void *user_data)

  *Register handler for read events.*

- void eloop_unregister_read_sock (int sock)

  *Unregister handler for read events.*

- int eloop_register_sock (int sock, eloop_event_type type, eloop_sock_handler handler, void *eloop_data, void *user_data)

  *Register handler for socket events.*

- void eloop_unregister_sock (int sock, eloop_event_type type)

  *Unregister handler for socket events.*

- int eloop_register_timeout (unsigned int secs, unsigned int usecs, eloop_timeout_handler handler, void *eloop_data, void *user_data)

  *Register timeout.*

- int eloop_cancel_timeout (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  *Cancel timeouts.*

- int eloop_register_signal (int sig, eloop_signal_handler handler, void *user_data)

  *Register handler for signals.*

- int eloop_register_signal_terminate (eloop_signal_handler handler, void *user_data)

  *Register handler for terminate signals.*

- int eloop_register_signal_reconfig (eloop_signal_handler handler, void *user_data)

  *Register handler for reconfig signals.*

- void eloop_run (void)

  *Start the event loop.*

- void eloop_terminate (void)

  *Terminate event loop.*

- void eloop_destroy (void)

  *Free any resources allocated for the event loop.*

- int eloop_terminated (void)

  *Check whether event loop has been terminated.*

- void eloop_wait_for_read_sock (int sock)

  *Wait for a single reader.*

- void * eloop_get_user_data (void)

  *Get global user data.*

### 6.100.1 Detailed Description

Event loop based on select() loop.

**Copyright**
Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eloop.c.

### 6.100.2 Function Documentation

#### 6.100.2.1 int eloop_cancel_timeout (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Cancel timeouts.

**Parameters:**
***handler*** Matching callback function
***eloop_data*** Matching eloop_data or ELOOP_ALL_CTX to match all
***user_data*** Matching user_data or ELOOP_ALL_CTX to match all

**Returns:**
Number of cancelled timeouts

Cancel matching <handler,eloop_data,user_data> timeouts registered with eloop_register_timeout(). ELOOP_ALL_CTX can be used as a wildcard for cancelling all timeouts regardless of eloop_data/user_data.

Definition at line 274 of file eloop.c.

#### 6.100.2.2 void eloop_destroy (void)

Free any resources allocated for the event loop.

After calling eloop_destroy(), other eloop_* functions must not be called before re-running eloop_init().

Definition at line 493 of file eloop.c.

#### 6.100.2.3 void* eloop_get_user_data (void)

Get global user data.

**Returns:**
user_data pointer that was registered with eloop_init()

Definition at line 529 of file eloop.c.

#### 6.100.2.4 int eloop_init (void * *user_data*)

Initialize global event loop data.

**Parameters:**
***user_data*** Pointer to global data passed as eloop_ctx to signal handlers

**Returns:**
0 on success, -1 on failure

This function must be called before any other eloop_* function. user_data can be used to configure a global (to the process) pointer that will be passed as eloop_ctx parameter to signal handlers.

Definition at line 73 of file eloop.c.

#### 6.100.2.5 int eloop_register_read_sock (int *sock*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for read events.

**Parameters:**
***sock*** File descriptor number for the socket
***handler*** Callback function to be called when data is available for reading
***eloop_data*** Callback context data (eloop_ctx)
***user_data*** Callback context data (sock_ctx)

**Returns:**
0 on success, -1 on failure

Register a read socket notifier for the given file descriptor. The handler function will be called whenever data is available for reading from the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

Definition at line 177 of file eloop.c.

#### 6.100.2.6 int eloop_register_signal (int *sig*, eloop_signal_handler *handler*, void * *user_data*)

Register handler for signals.

**Parameters:**
***sig*** Signal number (e.g., SIGHUP)
***handler*** Callback function to be called when the signal is received
***user_data*** Callback context data (signal_ctx)

**Returns:**
0 on success, -1 on failure

Register a callback function that will be called when a signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

Definition at line 369 of file eloop.c.

#### 6.100.2.7 int eloop_register_signal_reconfig (eloop_signal_handler *handler*, void * *user_data*)

Register handler for reconfig signals.

**Parameters:**

***handler*** Callback function to be called when the signal is received

***user_data*** Callback context data (signal_ctx)

**Returns:**

0 on success, -1 on failure

Register a callback function that will be called when a reconfiguration / hangup signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers a handler for SIGHUP.

Definition at line 403 of file eloop.c.

#### 6.100.2.8 int eloop_register_signal_terminate (eloop_signal_handler *handler*, void * *user_data*)

Register handler for terminate signals.

**Parameters:**

***handler*** Callback function to be called when the signal is received

***user_data*** Callback context data (signal_ctx)

**Returns:**

0 on success, -1 on failure

Register a callback function that will be called when a process termination signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers handlers for SIGINT and SIGTERM.

Definition at line 393 of file eloop.c.

#### 6.100.2.9 int eloop_register_sock (int *sock*, eloop_event_type *type*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for socket events.

**Parameters:**

***sock*** File descriptor number for the socket

***type*** Type of event to wait for

***handler*** Callback function to be called when the event is triggered

***eloop_data*** Callback context data (eloop_ctx)

***user_data*** Callback context data (sock_ctx)

**Returns:**
0 on success, -1 on failure

Register an event notifier for the given socket's file descriptor. The handler function will be called whenever the that event is triggered for the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

Definition at line 206 of file eloop.c.

#### 6.100.2.10 int eloop_register_timeout (unsigned int *secs*, unsigned int *usecs*, eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Register timeout.

**Parameters:**
***secs*** Number of seconds to the timeout

***usecs*** Number of microseconds to the timeout

***handler*** Callback function to be called when timeout occurs

***eloop_data*** Callback context data (eloop_ctx)

***user_data*** Callback context data (sock_ctx)

**Returns:**
0 on success, -1 on failure

Register a timeout that will cause the handler function to be called after given time.

Definition at line 227 of file eloop.c.

#### 6.100.2.11 void eloop_run (void)

Start the event loop.

Start the event loop and continue running as long as there are any registered event handlers. This function is run after event loop has been initialized with event_init() and one or more events have been registered.

Definition at line 414 of file eloop.c.

#### 6.100.2.12 void eloop_terminate (void)

Terminate event loop.

Terminate event loop even if there are registered events. This can be used to request the program to be terminated cleanly.

Definition at line 487 of file eloop.c.

#### 6.100.2.13 int eloop_terminated (void)

Check whether event loop has been terminated.

**Returns:**
1 = event loop terminate, 0 = event loop still running

This function can be used to check whether eloop_terminate() has been called to request termination of the event loop. This is normally used to abort operations that may still be queued to be run when eloop_terminate() was called.

Definition at line 510 of file eloop.c.

#### 6.100.2.14 void eloop_unregister_read_sock (int *sock*)

Unregister handler for read events.

**Parameters:**
***sock*** File descriptor number for the socket

Unregister a read socket notifier that was previously registered with eloop_register_read_sock().

Definition at line 185 of file eloop.c.

#### 6.100.2.15 void eloop_unregister_sock (int *sock*, eloop_event_type *type*)

Unregister handler for socket events.

**Parameters:**
***sock*** File descriptor number for the socket

***type*** Type of event for which sock was registered

Unregister a socket event notifier that was previously registered with eloop_register_sock().

Definition at line 218 of file eloop.c.

#### 6.100.2.16 void eloop_wait_for_read_sock (int *sock*)

Wait for a single reader.

**Parameters:**
***sock*** File descriptor number for the socket

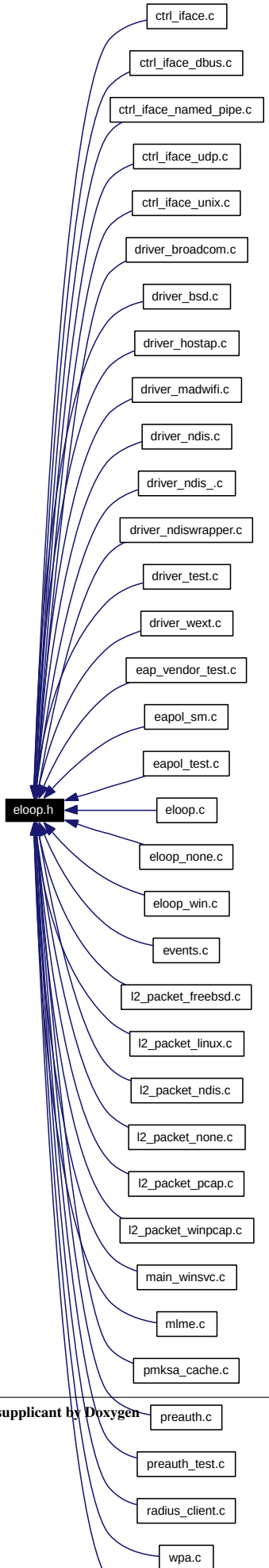
Do a blocking wait for a single read socket.

Definition at line 516 of file eloop.c.

## 6.101 eloop.h File Reference

Event loop.

This graph shows which files directly or indirectly include this file:


ctrl_iface.c
ctrl_iface_dbus.c
ctrl_iface_named_pipe.c
ctrl_iface_udp.c
ctrl_iface_unix.c
driver_broadcom.c
driver_bsd.c
driver_hostap.c
driver_madwifi.c
driver_ndis.c
driver_ndis_.c
driver_ndiswrapper.c
driver_test.c
driver_wext.c
eap_vendor_test.c
eapol_sm.c
eapol_test.c
eloop.h
eloop.c
eloop_none.c
eloop_win.c
events.c
l2_packet_freebsd.c
l2_packet_linux.c
l2_packet_ndis.c
l2_packet_none.c
l2_packet_pcap.c
l2_packet_winpcap.c
main_winsvc.c
mlme.c
pmksa_cache.c
preauth.c
preauth_test.c
radius_client.c
wpa.c

## Defines

- #define ELOOP_ALL_CTX (void *) -1

  *eloop_cancel_timeout() magic number to match all timeouts*

## Typedefs

- typedef void(* eloop_sock_handler )(int sock, void *eloop_ctx, void *sock_ctx)

  *eloop socket event callback type*

- typedef void(* eloop_event_handler )(void *eloop_data, void *user_ctx)

  *eloop generic event callback type*

- typedef void(* eloop_timeout_handler )(void *eloop_data, void *user_ctx)

  *eloop timeout event callback type*

- typedef void(* eloop_signal_handler )(int sig, void *eloop_ctx, void *signal_ctx)

  *eloop signal event callback type*

## Enumerations

- enum eloop_event_type { **EVENT_TYPE_READ** = 0, **EVENT_TYPE_WRITE**, **EVENT_TYPE_EXCEPTION** }

  *eloop socket event type for eloop_register_sock()*

## Functions

- int eloop_init (void *user_data)

  *Initialize global event loop data.*

- int eloop_register_read_sock (int sock, eloop_sock_handler handler, void *eloop_data, void *user_data)

  *Register handler for read events.*

- void eloop_unregister_read_sock (int sock)

  *Unregister handler for read events.*

- int eloop_register_sock (int sock, eloop_event_type type, eloop_sock_handler handler, void *eloop_data, void *user_data)

  *Register handler for socket events.*

- void eloop_unregister_sock (int sock, eloop_event_type type)

  *Unregister handler for socket events.*

- int eloop_register_event (void *event, size_t event_size, eloop_event_handler handler, void *eloop_data, void *user_data)

*Register handler for generic events.*

- void eloop_unregister_event (void *event, size_t event_size)

  *Unregister handler for a generic event.*

- int eloop_register_timeout (unsigned int secs, unsigned int usecs, eloop_timeout_handler handler, void *eloop_data, void *user_data)

  *Register timeout.*

- int eloop_cancel_timeout (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  *Cancel timeouts.*

- int eloop_register_signal (int sig, eloop_signal_handler handler, void *user_data)

  *Register handler for signals.*

- int eloop_register_signal_terminate (eloop_signal_handler handler, void *user_data)

  *Register handler for terminate signals.*

- int eloop_register_signal_reconfig (eloop_signal_handler handler, void *user_data)

  *Register handler for reconfig signals.*

- void eloop_run (void)

  *Start the event loop.*

- void eloop_terminate (void)

  *Terminate event loop.*

- void eloop_destroy (void)

  *Free any resources allocated for the event loop.*

- int eloop_terminated (void)

  *Check whether event loop has been terminated.*

- void eloop_wait_for_read_sock (int sock)

  *Wait for a single reader.*

- void * eloop_get_user_data (void)

  *Get global user data.*

### 6.101.1 Detailed Description

Event loop.

**Copyright**

Copyright (c) 2002-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file defines an event loop interface that supports processing events from registered timeouts (i.e., do something after N seconds), sockets (e.g., a new packet available for reading), and signals. eloop.c is an implementation of this interface using select() and sockets. This is suitable for most UNIX/POSIX systems. When porting to other operating systems, it may be necessary to replace that implementation with OS specific mechanisms.

Definition in file eloop.h.

## 6.101.2 Typedef Documentation

### 6.101.2.1 typedef void(* eloop_event_handler)(void *eloop_data, void *user_ctx)

eloop generic event callback type

**Parameters:**

***eloop_ctx*** Registered callback context data (eloop_data)

***sock_ctx*** Registered callback context data (user_data)

Definition at line 61 of file eloop.h.

### 6.101.2.2 typedef void(* eloop_signal_handler)(int sig, void *eloop_ctx, void *signal_ctx)

eloop signal event callback type

**Parameters:**

***sig*** Signal number

***eloop_ctx*** Registered callback context data (global user_data from eloop_init() call)

***signal_ctx*** Registered callback context data (user_data from eloop_register_signal(), eloop_register_signal_terminate(), or eloop_register_signal_reconfig() call)

Definition at line 81 of file eloop.h.

### 6.101.2.3 typedef void(* eloop_sock_handler)(int sock, void *eloop_ctx, void *sock_ctx)

eloop socket event callback type

**Parameters:**

***sock*** File descriptor number for the socket

***eloop_ctx*** Registered callback context data (eloop_data)

***sock_ctx*** Registered callback context data (user_data)

Definition at line 53 of file eloop.h.

#### 6.101.2.4 typedef void(* eloop_timeout_handler)(void *eloop_data, void *user_ctx)

eloop timeout event callback type

**Parameters:**

***eloop_ctx*** Registered callback context data (eloop_data)

***sock_ctx*** Registered callback context data (user_data)

Definition at line 69 of file eloop.h.

### 6.101.3 Enumeration Type Documentation

#### 6.101.3.1 enum eloop_event_type

eloop socket event type for eloop_register_sock()

**Parameters:**

***EVENT_TYPE_READ*** Socket has data available for reading

***EVENT_TYPE_WRITE*** Socket has room for new data to be written

***EVENT_TYPE_EXCEPTION*** An exception has been reported

Definition at line 40 of file eloop.h.

### 6.101.4 Function Documentation

#### 6.101.4.1 int eloop_cancel_timeout (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Cancel timeouts.

**Parameters:**

***handler*** Matching callback function

***eloop_data*** Matching eloop_data or ELOOP_ALL_CTX to match all

***user_data*** Matching user_data or ELOOP_ALL_CTX to match all

**Returns:**

Number of cancelled timeouts

Cancel matching <handler,eloop_data,user_data> timeouts registered with eloop_register_timeout(). ELOOP_ALL_CTX can be used as a wildcard for cancelling all timeouts regardless of eloop_data/user_data.

Definition at line 274 of file eloop.c.

#### 6.101.4.2 void eloop_destroy (void)

Free any resources allocated for the event loop.

After calling eloop_destroy(), other eloop_* functions must not be called before re-running eloop_init().

Definition at line 493 of file eloop.c.

#### 6.101.4.3 void* eloop_get_user_data (void)

Get global user data.

**Returns:**
user_data pointer that was registered with eloop_init()

Definition at line 529 of file eloop.c.

#### 6.101.4.4 int eloop_init (void * *user_data*)

Initialize global event loop data.

**Parameters:**
***user_data*** Pointer to global data passed as eloop_ctx to signal handlers

**Returns:**
0 on success, -1 on failure

This function must be called before any other eloop_* function. user_data can be used to configure a global (to the process) pointer that will be passed as eloop_ctx parameter to signal handlers.

Definition at line 73 of file eloop.c.

#### 6.101.4.5 int eloop_register_event (void * *event*, size_t *event_size*, eloop_event_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for generic events.

**Parameters:**
***event*** Event to wait (eloop implementation specific)

***event_size*** Size of event data

***handler*** Callback function to be called when event is triggered

***eloop_data*** Callback context data (eloop_data)

***user_data*** Callback context data (user_data)

**Returns:**
0 on success, -1 on failure

Register an event handler for the given event. This function is used to register eloop implementation specific events which are mainly targetted for operating system specific code (driver interface and l2_packet) since the portable code will not be able to use such an OS-specific call. The handler function will be called whenever the event is triggered. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

In case of Windows implementation (eloop_win.c), event pointer is of HANDLE type, i.e., void*. The callers are likely to have 'HANDLE h' type variable, and they would call this function with eloop_register_event(h, sizeof(h), ...).

Definition at line 191 of file eloop_win.c.

#### 6.101.4.6 int eloop_register_read_sock (int *sock*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for read events.

**Parameters:**

***sock*** File descriptor number for the socket

***handler*** Callback function to be called when data is available for reading

***eloop_data*** Callback context data (eloop_ctx)

***user_data*** Callback context data (sock_ctx)

**Returns:**

0 on success, -1 on failure

Register a read socket notifier for the given file descriptor. The handler function will be called whenever data is available for reading from the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

Definition at line 177 of file eloop.c.

Here is the call graph for this function:


#### 6.101.4.7 int eloop_register_signal (int *sig*, eloop_signal_handler *handler*, void * *user_data*)

Register handler for signals.

**Parameters:**

***sig*** Signal number (e.g., SIGHUP)

***handler*** Callback function to be called when the signal is received

***user_data*** Callback context data (signal_ctx)

**Returns:**

0 on success, -1 on failure

Register a callback function that will be called when a signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

Definition at line 369 of file eloop.c.

#### 6.101.4.8 int eloop_register_signal_reconfig (eloop_signal_handler *handler*, void * *user_data*)

Register handler for reconfig signals.

**Parameters:**

***handler*** Callback function to be called when the signal is received

***user_data*** Callback context data (signal_ctx)

**Returns:**

0 on success, -1 on failure

Register a callback function that will be called when a reconfiguration / hangup signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers a handler for SIGHUP.

Definition at line 403 of file eloop.c.

Here is the call graph for this function:


### 6.101.4.9 int eloop_register_signal_terminate (eloop_signal_handler *handler*, void * *user_data*)

Register handler for terminate signals.

**Parameters:**

***handler*** Callback function to be called when the signal is received

***user_data*** Callback context data (signal_ctx)

**Returns:**

0 on success, -1 on failure

Register a callback function that will be called when a process termination signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers handlers for SIGINT and SIGTERM.

Definition at line 393 of file eloop.c.

Here is the call graph for this function:

#### 6.101.4.10 int eloop_register_sock (int *sock*, eloop_event_type *type*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for socket events.

**Parameters:**

***sock*** File descriptor number for the socket

***type*** Type of event to wait for

***handler*** Callback function to be called when the event is triggered

***eloop_data*** Callback context data (eloop_ctx)

***user_data*** Callback context data (sock_ctx)

**Returns:**

0 on success, -1 on failure

Register an event notifier for the given socket's file descriptor. The handler function will be called whenever the that event is triggered for the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

Definition at line 206 of file eloop.c.

#### 6.101.4.11 int eloop_register_timeout (unsigned int *secs*, unsigned int *usecs*, eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Register timeout.

**Parameters:**

***secs*** Number of seconds to the timeout

***usecs*** Number of microseconds to the timeout

***handler*** Callback function to be called when timeout occurs

***eloop_data*** Callback context data (eloop_ctx)

***user_data*** Callback context data (sock_ctx)

**Returns:**

0 on success, -1 on failure

Register a timeout that will cause the handler function to be called after given time.

Definition at line 227 of file eloop.c.

Here is the call graph for this function:

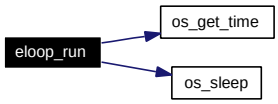

#### 6.101.4.12 void eloop_run (void)

Start the event loop.

Start the event loop and continue running as long as there are any registered event handlers. This function is run after event loop has been initialized with event_init() and one or more events have been registered.

Definition at line 414 of file eloop.c.

Here is the call graph for this function:




#### 6.101.4.13 void eloop_terminate (void)

Terminate event loop.

Terminate event loop even if there are registered events. This can be used to request the program to be terminated cleanly.

Definition at line 487 of file eloop.c.

#### 6.101.4.14 int eloop_terminated (void)

Check whether event loop has been terminated.

**Returns:**
1 = event loop terminate, 0 = event loop still running

This function can be used to check whether eloop_terminate() has been called to request termination of the event loop. This is normally used to abort operations that may still be queued to be run when eloop_terminate() was called.

Definition at line 510 of file eloop.c.

#### 6.101.4.15 void eloop_unregister_event (void * *event*, size_t *event_size*)

Unregister handler for a generic event.

**Parameters:**
***event*** Event to cancel (eloop implementation specific)

***event_size*** Size of event data

Unregister a generic event notifier that was previously registered with eloop_register_event().

Definition at line 220 of file eloop_win.c.

#### 6.101.4.16 void eloop_unregister_read_sock (int *sock*)

Unregister handler for read events.

**Parameters:**
***sock*** File descriptor number for the socket

Unregister a read socket notifier that was previously registered with eloop_register_read_sock().

Definition at line 185 of file eloop.c.

Here is the call graph for this function:


#### 6.101.4.17 void eloop_unregister_sock (int *sock*, eloop_event_type *type*)

Unregister handler for socket events.

**Parameters:**

***sock*** File descriptor number for the socket

***type*** Type of event for which sock was registered

Unregister a socket event notifier that was previously registered with eloop_register_sock().

Definition at line 218 of file eloop.c.

#### 6.101.4.18 void eloop_wait_for_read_sock (int *sock*)

Wait for a single reader.

**Parameters:**

***sock*** File descriptor number for the socket

Do a blocking wait for a single read socket.

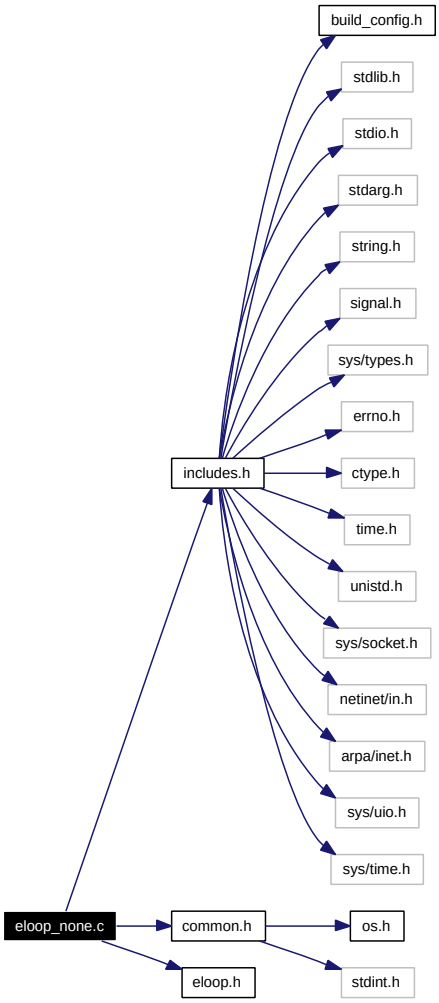
Definition at line 516 of file eloop.c.

## 6.102 eloop_none.c File Reference

Event loop - empty template (basic structure, but no OS specific operations).

```
#include "includes.h"
#include "common.h"
#include "eloop.h"
```

Include dependency graph for eloop_none.c:




### Data Structures

- struct **eloop_sock**
- struct **eloop_timeout**
- struct **eloop_signal**
- struct **eloop_data**

## Functions

- int eloop_init (void *user_data)
  *Initialize global event loop data.*
- int **eloop_register_read_sock** (int sock, void(*handler)(int sock, void *eloop_ctx, void *sock_ctx), void *eloop_data, void *user_data)
- void eloop_unregister_read_sock (int sock)
  *Unregister handler for read events.*
- int **eloop_register_timeout** (unsigned int secs, unsigned int usecs, void(*handler)(void *eloop_ctx, void *timeout_ctx), void *eloop_data, void *user_data)
- int **eloop_cancel_timeout** (void(*handler)(void *eloop_ctx, void *sock_ctx), void *eloop_data, void *user_data)
- int **eloop_register_signal** (int sig, void(*handler)(int sig, void *eloop_ctx, void *signal_ctx), void *user_data)
- int **eloop_register_signal_terminate** (void(*handler)(int sig, void *eloop_ctx, void *signal_ctx), void *user_data)
- int **eloop_register_signal_reconfig** (void(*handler)(int sig, void *eloop_ctx, void *signal_ctx), void *user_data)
- void eloop_run (void)
  *Start the event loop.*
- void eloop_terminate (void)
  *Terminate event loop.*
- void eloop_destroy (void)
  *Free any resources allocated for the event loop.*
- int eloop_terminated (void)
  *Check whether event loop has been terminated.*
- void eloop_wait_for_read_sock (int sock)
  *Wait for a single reader.*
- void * eloop_get_user_data (void)
  *Get global user data.*

### 6.102.1 Detailed Description

Event loop - empty template (basic structure, but no OS specific operations).



Definition in file eloop_none.c.

## 6.102.2 Function Documentation

### 6.102.2.1 void eloop_destroy (void)

Free any resources allocated for the event loop.

After calling eloop_destroy(), other eloop_∗ functions must not be called before re-running eloop_init().

Definition at line 358 of file eloop_none.c.

### 6.102.2.2 void∗ eloop_get_user_data (void)

Get global user data.

**Returns:**
user_data pointer that was registered with eloop_init()

Definition at line 388 of file eloop_none.c.

### 6.102.2.3 int eloop_init (void ∗ *user_data*)

Initialize global event loop data.

**Parameters:**
***user_data*** Pointer to global data passed as eloop_ctx to signal handlers

**Returns:**
0 on success, -1 on failure

This function must be called before any other eloop_∗ function. user_data can be used to configure a global (to the process) pointer that will be passed as eloop_ctx parameter to signal handlers.

Definition at line 64 of file eloop_none.c.

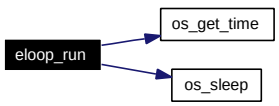
### 6.102.2.4 void eloop_run (void)

Start the event loop.

Start the event loop and continue running as long as there are any registered event handlers. This function is run after event loop has been initialized with event_init() and one or more events have been registered.

Definition at line 295 of file eloop_none.c.

Here is the call graph for this function:



#### 6.102.2.5 void eloop_terminate (void)

Terminate event loop.

Terminate event loop even if there are registered events. This can be used to request the program to be terminated cleanly.

Definition at line 352 of file eloop_none.c.

#### 6.102.2.6 int eloop_terminated (void)

Check whether event loop has been terminated.

**Returns:**
1 = event loop terminate, 0 = event loop still running

This function can be used to check whether eloop_terminate() has been called to request termination of the event loop. This is normally used to abort operations that may still be queued to be run when eloop_terminate() was called.

Definition at line 373 of file eloop_none.c.

#### 6.102.2.7 void eloop_unregister_read_sock (int *sock*)

Unregister handler for read events.

**Parameters:**
***sock*** File descriptor number for the socket

Unregister a read socket notifier that was previously registered with eloop_register_read_sock().

Definition at line 99 of file eloop_none.c.

#### 6.102.2.8 void eloop_wait_for_read_sock (int *sock*)

Wait for a single reader.

**Parameters:**
***sock*** File descriptor number for the socket

Do a blocking wait for a single read socket.

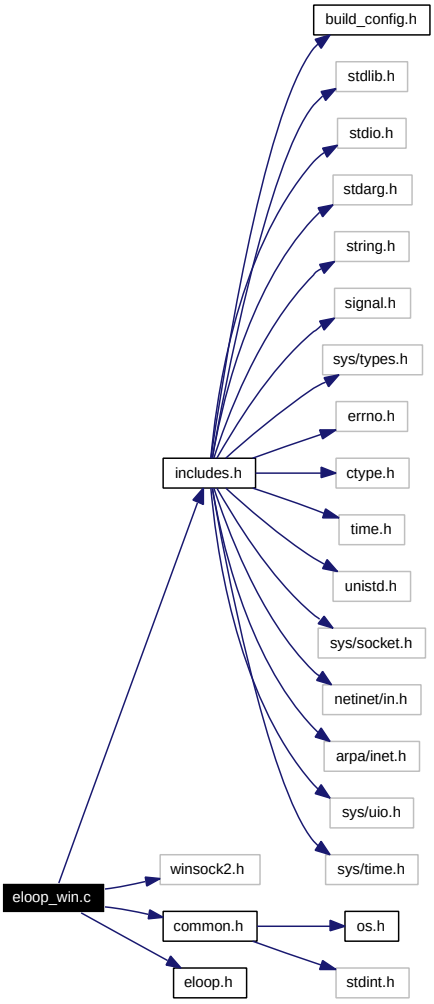
Definition at line 379 of file eloop_none.c.

## 6.103 eloop_win.c File Reference

Event loop based on Windows events and WaitForMultipleObjects.

```
#include "includes.h"
#include <winsock2.h>
#include "common.h"
#include "eloop.h"
```

Include dependency graph for eloop_win.c:




### Data Structures

- struct **eloop_sock**
- struct **eloop_timeout**
- struct **eloop_signal**
- struct **eloop_data**

## Functions

- int eloop_init (void *user_data)

  *Initialize global event loop data.*

- int eloop_register_read_sock (int sock, eloop_sock_handler handler, void *eloop_data, void *user_data)

  *Register handler for read events.*

- void eloop_unregister_read_sock (int sock)

  *Unregister handler for read events.*

- int eloop_register_event (void *event, size_t event_size, eloop_event_handler handler, void *eloop_data, void *user_data)

  *Register handler for generic events.*

- void eloop_unregister_event (void *event, size_t event_size)

  *Unregister handler for a generic event.*

- int eloop_register_timeout (unsigned int secs, unsigned int usecs, eloop_timeout_handler handler, void *eloop_data, void *user_data)

  *Register timeout.*

- int eloop_cancel_timeout (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  *Cancel timeouts.*

- int eloop_register_signal (int sig, eloop_signal_handler handler, void *user_data)

  *Register handler for signals.*

- int eloop_register_signal_terminate (eloop_signal_handler handler, void *user_data)

  *Register handler for terminate signals.*

- int eloop_register_signal_reconfig (eloop_signal_handler handler, void *user_data)

  *Register handler for reconfig signals.*

- void eloop_run (void)

  *Start the event loop.*

- void eloop_terminate (void)

  *Terminate event loop.*

- void eloop_destroy (void)

  *Free any resources allocated for the event loop.*

- int eloop_terminated (void)

  *Check whether event loop has been terminated.*

- void eloop_wait_for_read_sock (int sock)

  *Wait for a single reader.*

- void * eloop_get_user_data (void)

  *Get global user data.*

### 6.103.1 Detailed Description

Event loop based on Windows events and WaitForMultipleObjects.

**Copyright**
Copyright (c) 2002-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eloop_win.c.

### 6.103.2 Function Documentation

#### 6.103.2.1 int eloop_cancel_timeout (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Cancel timeouts.

**Parameters:**
***handler*** Matching callback function

***eloop_data*** Matching eloop_data or ELOOP_ALL_CTX to match all

***user_data*** Matching user_data or ELOOP_ALL_CTX to match all

**Returns:**
Number of cancelled timeouts

Cancel matching <handler,eloop_data,user_data> timeouts registered with eloop_register_timeout(). ELOOP_ALL_CTX can be used as a wildcard for cancelling all timeouts regardless of eloop_data/user_data.

Definition at line 292 of file eloop_win.c.

#### 6.103.2.2 void eloop_destroy (void)

Free any resources allocated for the event loop.

After calling eloop_destroy(), other eloop_* functions must not be called before re-running eloop_init().

Definition at line 553 of file eloop_win.c.

#### 6.103.2.3 void* eloop_get_user_data (void)

Get global user data.

**Returns:**
user_data pointer that was registered with eloop_init()

Definition at line 602 of file eloop_win.c.

#### 6.103.2.4 int eloop_init (void * *user_data*)

Initialize global event loop data.

**Parameters:**
***user_data*** Pointer to global data passed as eloop_ctx to signal handlers

**Returns:**
0 on success, -1 on failure

This function must be called before any other eloop_* function. user_data can be used to configure a global (to the process) pointer that will be passed as eloop_ctx parameter to signal handlers.

Definition at line 83 of file eloop_win.c.

#### 6.103.2.5 int eloop_register_event (void * *event*, size_t *event_size*, eloop_event_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for generic events.

**Parameters:**
***event*** Event to wait (eloop implementation specific)
***event_size*** Size of event data
***handler*** Callback function to be called when event is triggered
***eloop_data*** Callback context data (eloop_data)
***user_data*** Callback context data (user_data)

**Returns:**
0 on success, -1 on failure

Register an event handler for the given event. This function is used to register eloop implementation specific events which are mainly targetted for operating system specific code (driver interface and l2_packet) since the portable code will not be able to use such an OS-specific call. The handler function will be called whenever the event is triggered. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

In case of Windows implementation (eloop_win.c), event pointer is of HANDLE type, i.e., void*. The callers are likely to have 'HANDLE h' type variable, and they would call this function with eloop_register_event(h, sizeof(h), ...).

Definition at line 191 of file eloop_win.c.

#### 6.103.2.6 int eloop_register_read_sock (int *sock*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for read events.

**Parameters:**

***sock*** File descriptor number for the socket

***handler*** Callback function to be called when data is available for reading

***eloop_data*** Callback context data (eloop_ctx)

***user_data*** Callback context data (sock_ctx)

**Returns:**

0 on success, -1 on failure

Register a read socket notifier for the given file descriptor. The handler function will be called whenever data is available for reading from the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

Definition at line 121 of file eloop_win.c.

Here is the call graph for this function:


#### 6.103.2.7 int eloop_register_signal (int *sig*, eloop_signal_handler *handler*, void * *user_data*)

Register handler for signals.

**Parameters:**

***sig*** Signal number (e.g., SIGHUP)

***handler*** Callback function to be called when the signal is received

***user_data*** Callback context data (signal_ctx)

**Returns:**

0 on success, -1 on failure

Register a callback function that will be called when a signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

Definition at line 371 of file eloop_win.c.

#### 6.103.2.8 int eloop_register_signal_reconfig (eloop_signal_handler *handler*, void * *user_data*)

Register handler for reconfig signals.

**Parameters:**

***handler*** Callback function to be called when the signal is received

***user_data*** Callback context data (signal_ctx)

**Returns:**

0 on success, -1 on failure

Register a callback function that will be called when a reconfiguration / hangup signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers a handler for SIGHUP.

Definition at line 431 of file eloop_win.c.

Here is the call graph for this function:


#### 6.103.2.9 int eloop_register_signal_terminate (eloop_signal_handler *handler*, void * *user_data*)

Register handler for terminate signals.

**Parameters:**

***handler*** Callback function to be called when the signal is received

***user_data*** Callback context data (signal_ctx)

**Returns:**

0 on success, -1 on failure

Register a callback function that will be called when a process termination signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers handlers for SIGINT and SIGTERM.

Definition at line 412 of file eloop_win.c.

Here is the call graph for this function:


#### 6.103.2.10 int eloop_register_timeout (unsigned int *secs*, unsigned int *usecs*, eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Register timeout.

**Parameters:**

***secs*** Number of seconds to the timeout

***usecs*** Number of microseconds to the timeout

***handler*** Callback function to be called when timeout occurs

***eloop_data*** Callback context data (eloop_ctx)

***user_data*** Callback context data (sock_ctx)

**Returns:**
0 on success, -1 on failure

Register a timeout that will cause the handler function to be called after given time.

Definition at line 245 of file eloop_win.c.

Here is the call graph for this function:

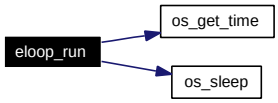

#### 6.103.2.11 void eloop_run (void)

Start the event loop.

Start the event loop and continue running as long as there are any registered event handlers. This function is run after event loop has been initialized with event_init() and one or more events have been registered.

Definition at line 439 of file eloop_win.c.

Here is the call graph for this function:




#### 6.103.2.12 void eloop_terminate (void)

Terminate event loop.

Terminate event loop even if there are registered events. This can be used to request the program to be terminated cleanly.

Definition at line 546 of file eloop_win.c.

#### 6.103.2.13 int eloop_terminated (void)

Check whether event loop has been terminated.

**Returns:**
1 = event loop terminate, 0 = event loop still running

This function can be used to check whether eloop_terminate() has been called to request termination of the event loop. This is normally used to abort operations that may still be queued to be run when eloop_terminate() was called.

Definition at line 574 of file eloop_win.c.

#### 6.103.2.14 void eloop_unregister_event (void * *event*, size_t *event_size*)

Unregister handler for a generic event.

**Parameters:**

***event*** Event to cancel (eloop implementation specific)

***event_size*** Size of event data

Unregister a generic event notifier that was previously registered with eloop_register_event().

Definition at line 220 of file eloop_win.c.

#### 6.103.2.15 void eloop_unregister_read_sock (int *sock*)

Unregister handler for read events.

**Parameters:**

***sock*** File descriptor number for the socket

Unregister a read socket notifier that was previously registered with eloop_register_read_sock().

Definition at line 164 of file eloop_win.c.

Here is the call graph for this function:


#### 6.103.2.16 void eloop_wait_for_read_sock (int *sock*)

Wait for a single reader.

**Parameters:**

***sock*** File descriptor number for the socket

Do a blocking wait for a single read socket.

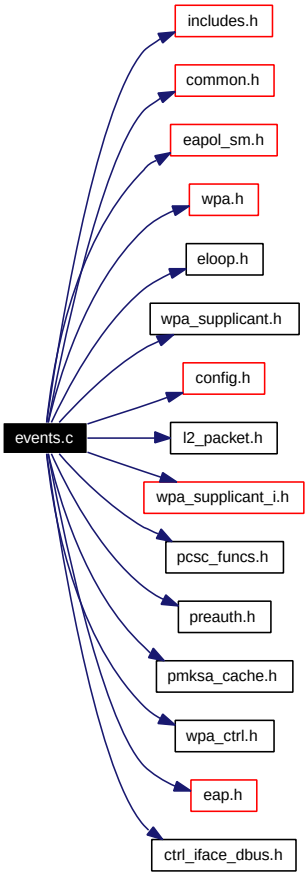
Definition at line 580 of file eloop_win.c.

## 6.104 events.c File Reference

WPA Supplicant - Driver event processing.

```
#include "includes.h"
#include "common.h"
#include "eapol_sm.h"
#include "wpa.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "config.h"
#include "l2_packet.h"
#include "wpa_supplicant_i.h"
#include "pcsc_funcs.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_ctrl.h"
#include "eap.h"
#include "ctrl_iface_dbus.h"
```

Include dependency graph for events.c:




## Functions

- int wpa_supplicant_scard_init (struct wpa_supplicant ∗wpa_s, struct wpa_ssid ∗ssid)

  *Initialize SIM/USIM access with PC/SC.*

- void wpa_supplicant_event (struct wpa_supplicant ∗wpa_s, wpa_event_type event, union wpa_-event_data ∗data)

  *Report a driver event for wpa_supplicant.*

### 6.104.1 Detailed Description

WPA Supplicant - Driver event processing.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file events.c.

## 6.104.2 Function Documentation

### 6.104.2.1 void wpa_supplicant_event (struct wpa_supplicant * *wpa_s*, wpa_event_type *event*, union wpa_event_data * *data*)

Report a driver event for wpa_supplicant.

**Parameters:**

***wpa_s*** pointer to wpa_supplicant data; this is the ctx variable registered with struct wpa_driver_ops::init()

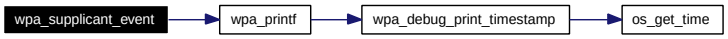
***event*** event type (defined above)

***data*** possible extra data for the event

Driver wrapper code should call this function whenever an event is received from the driver.

Definition at line 800 of file events.c.

Here is the call graph for this function:




### 6.104.2.2 int wpa_supplicant_scard_init (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Initialize SIM/USIM access with PC/SC.

**Parameters:**

***wpa_s*** pointer to wpa_supplicant data

***ssid*** Configuration data for the network

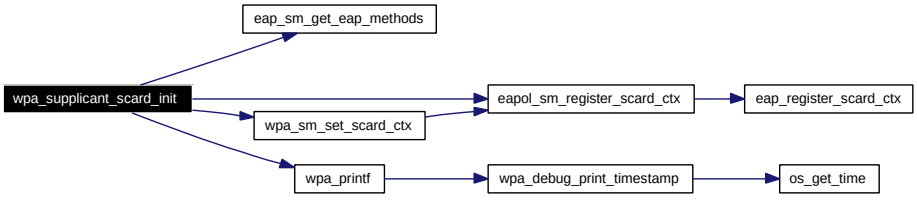
**Returns:**

0 on success, -1 on failure

This function is called when starting authentication with a network that is configured to use PC/SC for SIM/USIM access (EAP-SIM or EAP-AKA).

Definition at line 176 of file events.c.

Here is the call graph for this function:

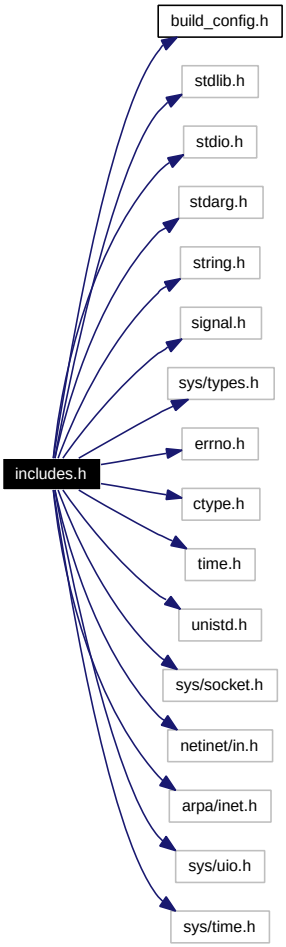


## 6.105 includes.h File Reference

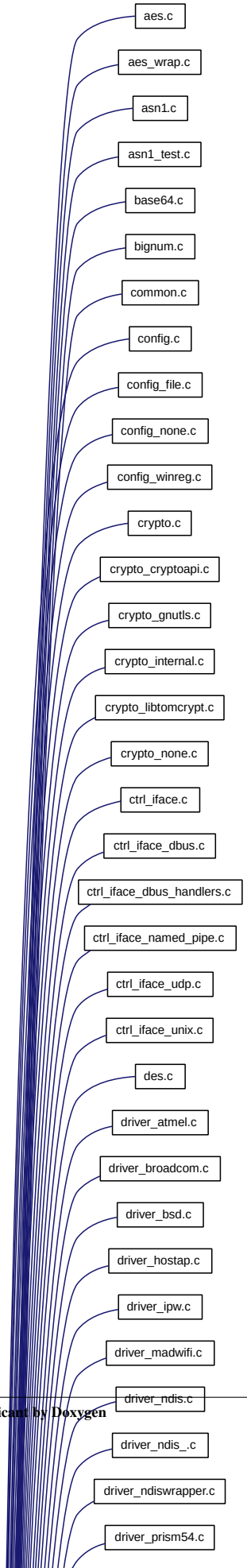
wpa_supplicant/hostapd - Default include files

```
#include "build_config.h"
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <signal.h>
#include <sys/types.h>
#include <errno.h>
#include <ctype.h>
#include <time.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/uio.h>
#include <sys/time.h>
```

Include dependency graph for includes.h:




This graph shows which files directly or indirectly include this file:


aes.c
aes_wrap.c
asn1.c
asn1_test.c
base64.c
bignum.c
common.c
config.c
config_file.c
config_none.c
config_winreg.c
crypto.c
crypto_cryptoapi.c
crypto_gnutls.c
crypto_internal.c
crypto_libtomcrypt.c
crypto_none.c
ctrl_iface.c
ctrl_iface_dbus.c
ctrl_iface_dbus_handlers.c
ctrl_iface_named_pipe.c
ctrl_iface_udp.c
ctrl_iface_unix.c
des.c
driver_atmel.c
driver_broadcom.c
driver_bsd.c
driver_hostap.c
driver_ipw.c
driver_madwifi.c
driver_ndis.c
driver_ndis_.c
driver_ndiswrapper.c
driver_prism54.c

### 6.105.1 Detailed Description

wpa_supplicant/hostapd - Default include files

**Copyright**

Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

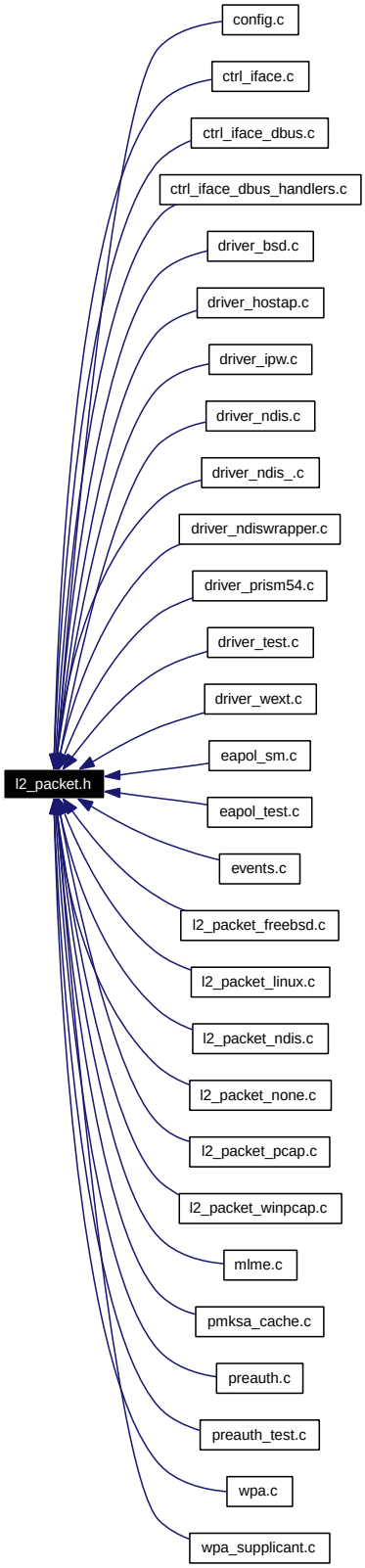
This header file is included into all C files so that commonly used header files can be selected with OS specific #ifdefs in one place instead of having to have OS/C library specific selection in many files.

Definition in file includes.h.

## 6.106 l2_packet.h File Reference

WPA Supplicant - Layer2 packet interface definition.

This graph shows which files directly or indirectly include this file:


config.c
ctrl_iface.c
ctrl_iface_dbus.c
ctrl_iface_dbus_handlers.c
driver_bsd.c
driver_hostap.c
driver_ipw.c
driver_ndis.c
driver_ndis_.c
driver_ndiswrapper.c
driver_prism54.c
driver_test.c
driver_wext.c
eapol_sm.c
l2_packet.h
eapol_test.c
events.c
l2_packet_freebsd.c
l2_packet_linux.c
l2_packet_ndis.c
l2_packet_none.c
l2_packet_pcap.c
l2_packet_winpcap.c
mlme.c
pmksa_cache.c
preauth.c
preauth_test.c
wpa.c
wpa_supplicant.c

## Defines

- #define **MAC2STR**(a) (a)[0], (a)[1], (a)[2], (a)[3], (a)[4], (a)[5]
- #define **MACSTR** "%02x:%02x:%02x:%02x:%02x:%02x"
- #define **ETH_P_EAPOL** 0x888e
- #define **ETH_P_RSN_PREAUTH** 0x88c7

## Functions

- l2_packet_data * l2_packet_init (const char *ifname, const u8 *own_addr, unsigned short protocol, void(*rx_callback)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len), void *rx_callback_ctx, int l2_hdr)

  *Initialize l2_packet interface.*

- void l2_packet_deinit (struct l2_packet_data *l2)

  *Deinitialize l2_packet interface.*

- int l2_packet_get_own_addr (struct l2_packet_data *l2, u8 *addr)

  *Get own layer 2 address.*

- int l2_packet_send (struct l2_packet_data *l2, const u8 *dst_addr, u16 proto, const u8 *buf, size_t len)

  *Send a packet.*

- int l2_packet_get_ip_addr (struct l2_packet_data *l2, char *buf, size_t len)

  *Get the current IP address from the interface.*

- void l2_packet_notify_auth_start (struct l2_packet_data *l2)

  *Notify l2_packet about start of authentication.*

## Variables

- l2_ethhdr **STRUCT_PACKED**

### 6.106.1 Detailed Description

WPA Supplicant - Layer2 packet interface definition.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file defines an interface for layer 2 (link layer) packet sending and receiving. l2_packet_linux.c is one implementation for such a layer 2 implementation using Linux packet sockets and l2_packet_pcap.c

another one using libpcap and libdnet. When porting wpa_supplicant to other operating systems, a new l2_packet implementation may need to be added.

Definition in file l2_packet.h.

## 6.106.2 Function Documentation

### 6.106.2.1 void l2_packet_deinit (struct l2_packet_data * *l2*)

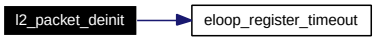
Deinitialize l2_packet interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

Definition at line 225 of file l2_packet_freebsd.c.

Here is the call graph for this function:




### 6.106.2.2 int l2_packet_get_ip_addr (struct l2_packet_data * *l2*, char * *buf*, size_t *len*)

Get the current IP address from the interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***buf*** Buffer for the IP address in text format

***len*** Maximum buffer length

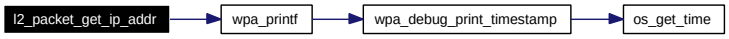
**Returns:**

0 on success, -1 on failure

This function can be used to get the current IP address from the interface bound to the l2_packet. This is mainly for status information and the IP address will be stored as an ASCII string. This function is not essential for wpa_supplicant operation, so full implementation is not required. l2_packet implementation will need to define the function, but it can return -1 if the IP address information is not available.

Definition at line 235 of file l2_packet_freebsd.c.

Here is the call graph for this function:




### 6.106.2.3 int l2_packet_get_own_addr (struct l2_packet_data * *l2*, u8 * *addr*)

Get own layer 2 address.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***addr*** Buffer for the own address (6 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 45 of file l2_packet_freebsd.c.

#### 6.106.2.4 struct l2_packet_data* l2_packet_init (const char * *ifname*, const u8 * *own_addr*, unsigned short *protocol*, void(*)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len) *rx_callback*, void * *rx_callback_ctx*, int *l2_hdr*)

Initialize l2_packet interface.

**Parameters:**

***ifname*** Interface name

***own_addr*** Optional own MAC address if available from driver interface or NULL if not available

***protocol*** Ethernet protocol number in host byte order

***rx_callback*** Callback function that will be called for each received packet

***rx_callback_ctx*** Callback data (ctx) for calls to rx_callback()

***l2_hdr*** 1 = include layer 2 header, 0 = do not include header

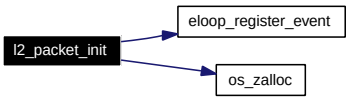
**Returns:**

Pointer to internal data or NULL on failure

rx_callback function will be called with src_addr pointing to the source address (MAC address) of the the packet. If l2_hdr is set to 0, buf points to len bytes of the payload after the layer 2 header and similarly, TX buffers start with payload. This behavior can be changed by setting l2_hdr=1 to include the layer 2 header in the data buffer.

Definition at line 193 of file l2_packet_freebsd.c.

Here is the call graph for this function:




#### 6.106.2.5 void l2_packet_notify_auth_start (struct l2_packet_data * *l2*)

Notify l2_packet about start of authentication.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

This function is called when authentication is expected to start, e.g., when association has been completed, in order to prepare l2_packet implementation for EAPOL frames. This function is used mainly if the l2_packet code needs to do polling in which case it can increasing polling frequency. This can also be an empty function if the l2_packet implementation does not benefit from knowing about the starting authentication.

Definition at line 271 of file l2_packet_freebsd.c.

#### 6.106.2.6 int l2_packet_send (struct l2_packet_data * *l2*, const u8 * *dst_addr*, u16 *proto*, const u8 * *buf*, size_t *len*)

Send a packet.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***dst_addr*** Destination address for the packet (only used if l2_hdr == 0)

***proto*** Protocol/ethertype for the packet in host byte order (only used if l2_hdr == 0)

***buf*** Packet contents to be sent; including layer 2 header if l2_hdr was set to 1 in l2_packet_init() call. Otherwise, only the payload of the packet is included.

***len*** Length of the buffer (including l2 header only if l2_hdr == 1)

**Returns:**

>=0 on success, <0 on failure

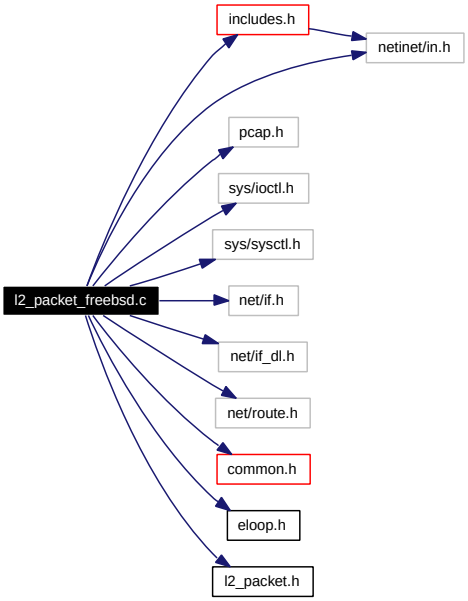
Definition at line 52 of file l2_packet_freebsd.c.

# 6.107 l2_packet_freebsd.c File Reference

WPA Supplicant - Layer2 packet handling with FreeBSD.

```
#include "includes.h"
#include <pcap.h>
#include <sys/ioctl.h>
#include <sys/sysctl.h>
#include <net/if.h>
#include <net/if_dl.h>
#include <net/route.h>
#include <netinet/in.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

Include dependency graph for l2_packet_freebsd.c:




## Functions

- int l2_packet_get_own_addr (struct l2_packet_data *l2, u8 *addr)

  *Get own layer 2 address.*

- int l2_packet_send (struct l2_packet_data *l2, const u8 *dst_addr, u16 proto, const u8 *buf, size_t len)

  *Send a packet.*

- l2_packet_data * l2_packet_init (const char *ifname, const u8 *own_addr, unsigned short protocol, void(*rx_callback)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len), void *rx_callback_ctx, int l2_hdr)

  *Initialize l2_packet interface.*

- void l2_packet_deinit (struct l2_packet_data *l2)

  *Deinitialize l2_packet interface.*

- int l2_packet_get_ip_addr (struct l2_packet_data *l2, char *buf, size_t len)

  *Get the current IP address from the interface.*

- void l2_packet_notify_auth_start (struct l2_packet_data *l2)

  *Notify l2_packet about start of authentication.*

### 6.107.1 Detailed Description

WPA Supplicant - Layer2 packet handling with FreeBSD.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi> Copyright (c) 2005, Sam Leffler <sam@errno.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file l2_packet_freebsd.c.

### 6.107.2 Function Documentation

#### 6.107.2.1 void l2_packet_deinit (struct l2_packet_data * *l2*)

Deinitialize l2_packet interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

Definition at line 225 of file l2_packet_freebsd.c.

#### 6.107.2.2 int l2_packet_get_ip_addr (struct l2_packet_data * *l2*, char * *buf*, size_t *len*)

Get the current IP address from the interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***buf*** Buffer for the IP address in text format

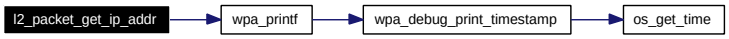
***len*** Maximum buffer length

**Returns:**
0 on success, -1 on failure

This function can be used to get the current IP address from the interface bound to the l2_packet. This is mainly for status information and the IP address will be stored as an ASCII string. This function is not essential for wpa_supplicant operation, so full implementation is not required. l2_packet implementation will need to define the function, but it can return -1 if the IP address information is not available.

Definition at line 235 of file l2_packet_freebsd.c.

Here is the call graph for this function:




#### 6.107.2.3 int l2_packet_get_own_addr (struct l2_packet_data * *l2*, u8 * *addr*)

Get own layer 2 address.

**Parameters:**
***l2*** Pointer to internal l2_packet data from l2_packet_init()

***addr*** Buffer for the own address (6 bytes)

**Returns:**
0 on success, -1 on failure

Definition at line 45 of file l2_packet_freebsd.c.

#### 6.107.2.4 struct l2_packet_data* l2_packet_init (const char * *ifname*, const u8 * *own_addr*, unsigned short *protocol*, void(*)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len) *rx_callback*, void * *rx_callback_ctx*, int *l2_hdr*)

Initialize l2_packet interface.

**Parameters:**
***ifname*** Interface name

***own_addr*** Optional own MAC address if available from driver interface or NULL if not available

***protocol*** Ethernet protocol number in host byte order

***rx_callback*** Callback function that will be called for each received packet

***rx_callback_ctx*** Callback data (ctx) for calls to rx_callback()

***l2_hdr*** 1 = include layer 2 header, 0 = do not include header

**Returns:**
Pointer to internal data or NULL on failure

rx_callback function will be called with src_addr pointing to the source address (MAC address) of the the packet. If l2_hdr is set to 0, buf points to len bytes of the payload after the layer 2 header and similarly, TX buffers start with payload. This behavior can be changed by setting l2_hdr=1 to include the layer 2 header in the data buffer.

Definition at line 193 of file l2_packet_freebsd.c.

Here is the call graph for this function:


#### 6.107.2.5 void l2_packet_notify_auth_start (struct l2_packet_data * *l2*)

Notify l2_packet about start of authentication.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

This function is called when authentication is expected to start, e.g., when association has been completed, in order to prepare l2_packet implementation for EAPOL frames. This function is used mainly if the l2_packet code needs to do polling in which case it can increasing polling frequency. This can also be an empty function if the l2_packet implementation does not benefit from knowing about the starting authentication.

Definition at line 271 of file l2_packet_freebsd.c.

#### 6.107.2.6 int l2_packet_send (struct l2_packet_data * *l2*, const u8 * *dst_addr*, u16 *proto*, const u8 * *buf*, size_t *len*)

Send a packet.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***dst_addr*** Destination address for the packet (only used if l2_hdr == 0)

***proto*** Protocol/ethertype for the packet in host byte order (only used if l2_hdr == 0)

***buf*** Packet contents to be sent; including layer 2 header if l2_hdr was set to 1 in l2_packet_init() call. Otherwise, only the payload of the packet is included.

***len*** Length of the buffer (including l2 header only if l2_hdr == 1)

**Returns:**

>=0 on success, <0 on failure

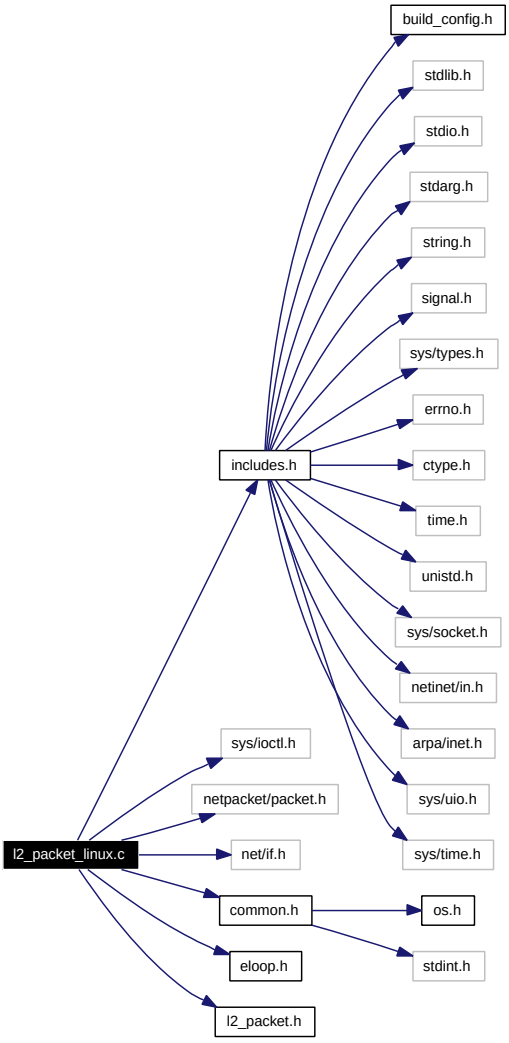
Definition at line 52 of file l2_packet_freebsd.c.

## 6.108 l2_packet_linux.c File Reference

WPA Supplicant - Layer2 packet handling with Linux packet sockets.

```
#include "includes.h"
#include <sys/ioctl.h>
#include <netpacket/packet.h>
#include <net/if.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

Include dependency graph for l2_packet_linux.c:



## Data Structures

- struct **l2_packet_data**

## Functions

- int l2_packet_get_own_addr (struct l2_packet_data *l2, u8 *addr)

  *Get own layer 2 address.*

- int l2_packet_send (struct l2_packet_data *l2, const u8 *dst_addr, u16 proto, const u8 *buf, size_t len)

  *Send a packet.*

- l2_packet_data * l2_packet_init (const char *ifname, const u8 *own_addr, unsigned short protocol, void(*rx_callback)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len), void *rx_callback_ctx, int l2_hdr)

  *Initialize l2_packet interface.*

- void l2_packet_deinit (struct l2_packet_data *l2)

  *Deinitialize l2_packet interface.*

- int l2_packet_get_ip_addr (struct l2_packet_data *l2, char *buf, size_t len)

  *Get the current IP address from the interface.*

- void l2_packet_notify_auth_start (struct l2_packet_data *l2)

  *Notify l2_packet about start of authentication.*

### 6.108.1 Detailed Description

WPA Supplicant - Layer2 packet handling with Linux packet sockets.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file l2_packet_linux.c.

### 6.108.2 Function Documentation

#### 6.108.2.1 void l2_packet_deinit (struct l2_packet_data * *l2*)

Deinitialize l2_packet interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

Definition at line 153 of file l2_packet_linux.c.

Here is the call graph for this function:


#### 6.108.2.2 int l2_packet_get_ip_addr (struct l2_packet_data * *l2*, char * *buf*, size_t *len*)

Get the current IP address from the interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***buf*** Buffer for the IP address in text format

***len*** Maximum buffer length

**Returns:**

0 on success, -1 on failure

This function can be used to get the current IP address from the interface bound to the l2_packet. This is mainly for status information and the IP address will be stored as an ASCII string. This function is not essential for wpa_supplicant operation, so full implementation is not required. l2_packet implementation will need to define the function, but it can return -1 if the IP address information is not available.

Definition at line 167 of file l2_packet_linux.c.

#### 6.108.2.3 int l2_packet_get_own_addr (struct l2_packet_data * *l2*, u8 * *addr*)

Get own layer 2 address.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***addr*** Buffer for the own address (6 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 39 of file l2_packet_linux.c.

#### 6.108.2.4 struct l2_packet_data* l2_packet_init (const char * *ifname*, const u8 * *own_addr*, unsigned short *protocol*, void(*)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len) *rx_callback*, void * *rx_callback_ctx*, int *l2_hdr*)

Initialize l2_packet interface.

**Parameters:**

***ifname*** Interface name

***own_addr*** Optional own MAC address if available from driver interface or NULL if not available

***protocol*** Ethernet protocol number in host byte order

***rx_callback*** Callback function that will be called for each received packet

***rx_callback_ctx*** Callback data (ctx) for calls to rx_callback()

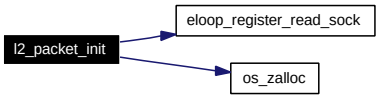
***l2_hdr*** 1 = include layer 2 header, 0 = do not include header

**Returns:**
Pointer to internal data or NULL on failure

rx_callback function will be called with src_addr pointing to the source address (MAC address) of the the packet. If l2_hdr is set to 0, buf points to len bytes of the payload after the layer 2 header and similarly, TX buffers start with payload. This behavior can be changed by setting l2_hdr=1 to include the layer 2 header in the data buffer.

Definition at line 94 of file l2_packet_linux.c.

Here is the call graph for this function:




#### 6.108.2.5 void l2_packet_notify_auth_start (struct l2_packet_data * *l2*)

Notify l2_packet about start of authentication.

**Parameters:**
***l2*** Pointer to internal l2_packet data from l2_packet_init()

This function is called when authentication is expected to start, e.g., when association has been completed, in order to prepare l2_packet implementation for EAPOL frames. This function is used mainly if the l2_-packet code needs to do polling in which case it can increasing polling frequency. This can also be an empty function if the l2_packet implementation does not benefit from knowing about the starting authentication.

Definition at line 194 of file l2_packet_linux.c.

#### 6.108.2.6 int l2_packet_send (struct l2_packet_data * *l2*, const u8 * *dst_addr*, u16 *proto*, const u8 * *buf*, size_t *len*)

Send a packet.

**Parameters:**
***l2*** Pointer to internal l2_packet data from l2_packet_init()

***dst_addr*** Destination address for the packet (only used if l2_hdr == 0)

***proto*** Protocol/ethertype for the packet in host byte order (only used if l2_hdr == 0)

***buf*** Packet contents to be sent; including layer 2 header if l2_hdr was set to 1 in l2_packet_init() call. Otherwise, only the payload of the packet is included.

***len*** Length of the buffer (including l2 header only if l2_hdr == 1)

**Returns:**
>=0 on success, <0 on failure

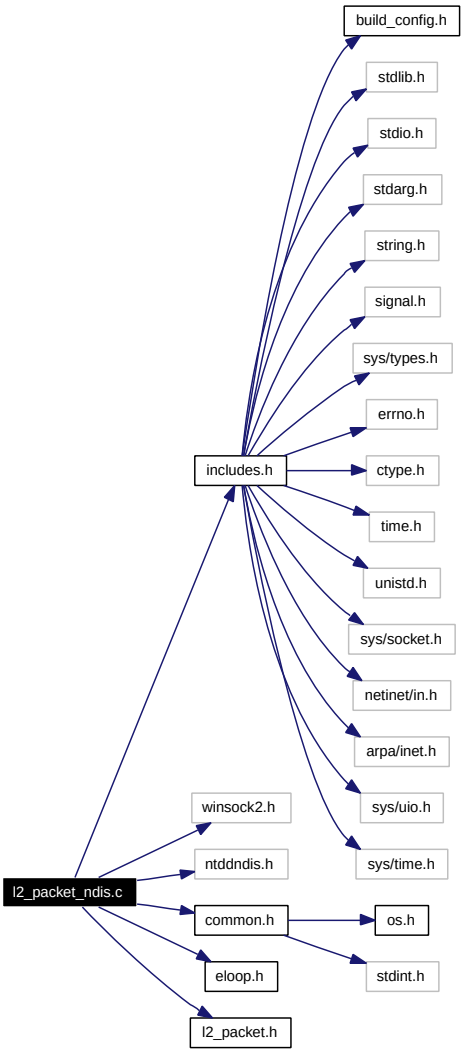
Definition at line 46 of file l2_packet_linux.c.

# 6.109 l2_packet_ndis.c File Reference

WPA Supplicant - Layer2 packet handling with Microsoft NDISUIO.

```
#include "includes.h"
#include <winsock2.h>
#include <ntddndis.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

Include dependency graph for l2_packet_ndis.c:




## Data Structures

- struct **l2_packet_data**

## Defines

- #define **FSCTL_NDISUIO_BASE** FILE_DEVICE_NETWORK
- #define **_NDISUIO_CTL_CODE**(_Function, _Method, _Access) CTL_CODE(FSCTL_-NDISUIO_BASE, _Function, _Method, _Access)
- #define **IOCTL_NDISUIO_SET_ETHER_TYPE**

## Functions

- HANDLE **driver_ndis_get_ndisuio_handle** (void)
- int l2_packet_get_own_addr (struct l2_packet_data *l2, u8 *addr)

  *Get own layer 2 address.*

- int l2_packet_send (struct l2_packet_data *l2, const u8 *dst_addr, u16 proto, const u8 *buf, size_t len)

  *Send a packet.*

- l2_packet_data * l2_packet_init (const char *ifname, const u8 *own_addr, unsigned short protocol, void(*rx_callback)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len), void *rx_callback_ctx, int l2_hdr)

  *Initialize l2_packet interface.*

- void l2_packet_deinit (struct l2_packet_data *l2)

  *Deinitialize l2_packet interface.*

- int l2_packet_get_ip_addr (struct l2_packet_data *l2, char *buf, size_t len)

  *Get the current IP address from the interface.*

- void l2_packet_notify_auth_start (struct l2_packet_data *l2)

  *Notify l2_packet about start of authentication.*

### 6.109.1 Detailed Description

WPA Supplicant - Layer2 packet handling with Microsoft NDISUIO.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>



This implementation requires Windows specific event loop implementation, i.e., eloop_win.c. In addition, the NDISUIO connection is shared with driver_ndis.c, so only that driver interface can be used and CONFIG_USE_NDISUIO must be defined.

WinXP version of the code uses overlapped I/O and a single threaded design with callback functions from I/O code. WinCE version uses a separate RX thread that blocks on ReadFile() whenever the media status is connected.

Definition in file l2_packet_ndis.c.

## 6.109.2 Define Documentation

### 6.109.2.1 #define IOCTL_NDISUIO_SET_ETHER_TYPE

**Value:**

```
_NDISUIO_CTL_CODE(0x202, METHOD_BUFFERED, \
                          FILE_READ_ACCESS | FILE_WRITE_ACCESS)
```

Definition at line 43 of file l2_packet_ndis.c.

## 6.109.3 Function Documentation

### 6.109.3.1 void l2_packet_deinit (struct l2_packet_data * *l2*)

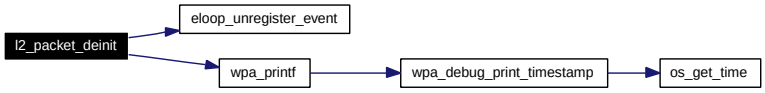
Deinitialize l2_packet interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

Definition at line 454 of file l2_packet_ndis.c.

Here is the call graph for this function:




### 6.109.3.2 int l2_packet_get_ip_addr (struct l2_packet_data * *l2*, char * *buf*, size_t *len*)

Get the current IP address from the interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***buf*** Buffer for the IP address in text format

***len*** Maximum buffer length

**Returns:**

0 on success, -1 on failure

This function can be used to get the current IP address from the interface bound to the l2_packet. This is mainly for status information and the IP address will be stored as an ASCII string. This function is not essential for wpa_supplicant operation, so full implementation is not required. l2_packet implementation will need to define the function, but it can return -1 if the IP address information is not available.

Definition at line 509 of file l2_packet_ndis.c.

#### 6.109.3.3 int l2_packet_get_own_addr (struct l2_packet_data * *l2*, u8 * *addr*)

Get own layer 2 address.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***addr*** Buffer for the own address (6 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 91 of file l2_packet_ndis.c.

#### 6.109.3.4 struct l2_packet_data* l2_packet_init (const char * *ifname*, const u8 * *own_addr*, unsigned short *protocol*, void(*)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len) *rx_callback*, void * *rx_callback_ctx*, int *l2_hdr*)

Initialize l2_packet interface.

**Parameters:**

***ifname*** Interface name

***own_addr*** Optional own MAC address if available from driver interface or NULL if not available

***protocol*** Ethernet protocol number in host byte order

***rx_callback*** Callback function that will be called for each received packet

***rx_callback_ctx*** Callback data (ctx) for calls to rx_callback()

***l2_hdr*** 1 = include layer 2 header, 0 = do not include header

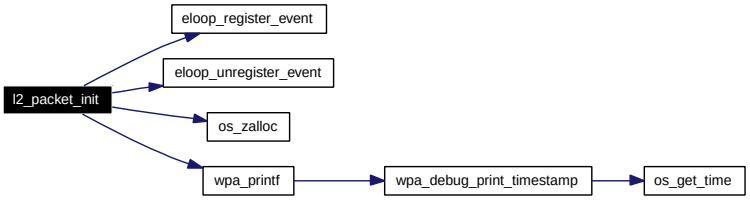
**Returns:**

Pointer to internal data or NULL on failure

rx_callback function will be called with src_addr pointing to the source address (MAC address) of the the packet. If l2_hdr is set to 0, buf points to len bytes of the payload after the layer 2 header and similarly, TX buffers start with payload. This behavior can be changed by setting l2_hdr=1 to include the layer 2 header in the data buffer.

Definition at line 349 of file l2_packet_ndis.c.

Here is the call graph for this function:



#### 6.109.3.5 void l2_packet_notify_auth_start (struct l2_packet_data * *l2*)

Notify l2_packet about start of authentication.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

This function is called when authentication is expected to start, e.g., when association has been completed, in order to prepare l2_packet implementation for EAPOL frames. This function is used mainly if the l2_packet code needs to do polling in which case it can increasing polling frequency. This can also be an empty function if the l2_packet implementation does not benefit from knowing about the starting authentication.

Definition at line 515 of file l2_packet_ndis.c.

#### 6.109.3.6 int l2_packet_send (struct l2_packet_data * *l2*, const u8 * *dst_addr*, u16 *proto*, const u8 * *buf*, size_t *len*)

Send a packet.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***dst_addr*** Destination address for the packet (only used if l2_hdr == 0)

***proto*** Protocol/ethertype for the packet in host byte order (only used if l2_hdr == 0)

***buf*** Packet contents to be sent; including layer 2 header if l2_hdr was set to 1 in l2_packet_init() call. Otherwise, only the payload of the packet is included.

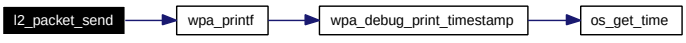
***len*** Length of the buffer (including l2 header only if l2_hdr == 1)

**Returns:**

>=0 on success, <0 on failure

Definition at line 98 of file l2_packet_ndis.c.

Here is the call graph for this function:

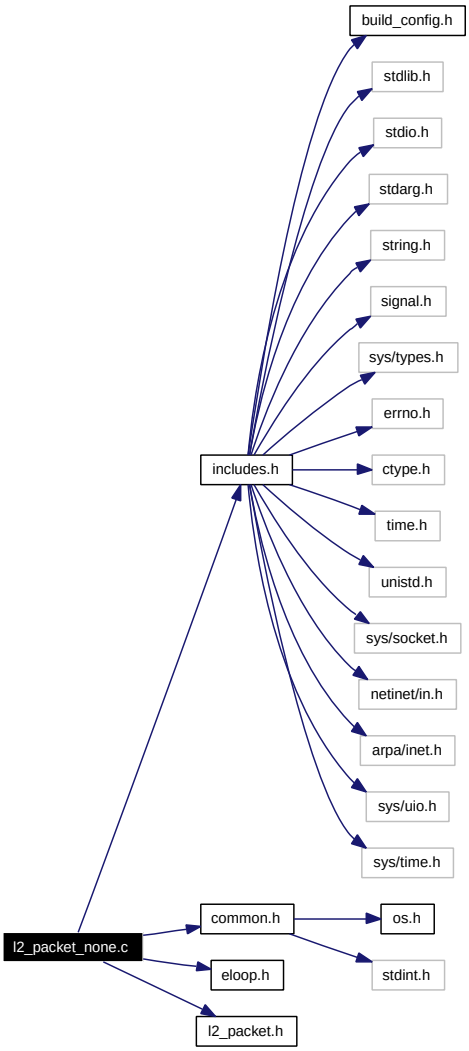


## 6.110 l2_packet_none.c File Reference

WPA Supplicant - Layer2 packet handling example with dummy functions.

```
#include "includes.h"
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

Include dependency graph for l2_packet_none.c:




### Data Structures

- struct **l2_packet_data**

## Functions

- int l2_packet_get_own_addr (struct l2_packet_data *l2, u8 *addr)

  *Get own layer 2 address.*

- int l2_packet_send (struct l2_packet_data *l2, const u8 *dst_addr, u16 proto, const u8 *buf, size_t len)

  *Send a packet.*

- l2_packet_data * l2_packet_init (const char *ifname, const u8 *own_addr, unsigned short protocol, void(*rx_callback)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len), void *rx_callback_ctx, int l2_hdr)

  *Initialize l2_packet interface.*

- void l2_packet_deinit (struct l2_packet_data *l2)

  *Deinitialize l2_packet interface.*

- int l2_packet_get_ip_addr (struct l2_packet_data *l2, char *buf, size_t len)

  *Get the current IP address from the interface.*

- void l2_packet_notify_auth_start (struct l2_packet_data *l2)

  *Notify l2_packet about start of authentication.*

### 6.110.1 Detailed Description

WPA Supplicant - Layer2 packet handling example with dummy functions.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file can be used as a starting point for layer2 packet implementation.

Definition in file l2_packet_none.c.

### 6.110.2 Function Documentation

#### 6.110.2.1 void l2_packet_deinit (struct l2_packet_data * *l2*)

Deinitialize l2_packet interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

Definition at line 100 of file l2_packet_none.c.

Here is the call graph for this function:


#### 6.110.2.2 int l2_packet_get_ip_addr (struct l2_packet_data * *l2*, char * *buf*, size_t *len*)

Get the current IP address from the interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***buf*** Buffer for the IP address in text format

***len*** Maximum buffer length

**Returns:**

0 on success, -1 on failure

This function can be used to get the current IP address from the interface bound to the l2_packet. This is mainly for status information and the IP address will be stored as an ASCII string. This function is not essential for wpa_supplicant operation, so full implementation is not required. l2_packet implementation will need to define the function, but it can return -1 if the IP address information is not available.

Definition at line 114 of file l2_packet_none.c.

#### 6.110.2.3 int l2_packet_get_own_addr (struct l2_packet_data * *l2*, u8 * *addr*)

Get own layer 2 address.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***addr*** Buffer for the own address (6 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 37 of file l2_packet_none.c.

#### 6.110.2.4 struct l2_packet_data* l2_packet_init (const char * *ifname*, const u8 * *own_addr*, unsigned short *protocol*, void(*)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len) *rx_callback*, void * *rx_callback_ctx*, int *l2_hdr*)

Initialize l2_packet interface.

**Parameters:**

***ifname*** Interface name

***own_addr*** Optional own MAC address if available from driver interface or NULL if not available

***protocol*** Ethernet protocol number in host byte order

***rx_callback*** Callback function that will be called for each received packet

***rx_callback_ctx*** Callback data (ctx) for calls to rx_callback()

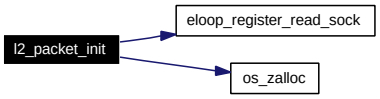
***l2_hdr*** 1 = include layer 2 header, 0 = do not include header

**Returns:**
Pointer to internal data or NULL on failure

rx_callback function will be called with src_addr pointing to the source address (MAC address) of the the packet. If l2_hdr is set to 0, buf points to len bytes of the payload after the layer 2 header and similarly, TX buffers start with payload. This behavior can be changed by setting l2_hdr=1 to include the layer 2 header in the data buffer.

Definition at line 74 of file l2_packet_none.c.

Here is the call graph for this function:




#### 6.110.2.5 void l2_packet_notify_auth_start (struct l2_packet_data * *l2*)

Notify l2_packet about start of authentication.

**Parameters:**
***l2*** Pointer to internal l2_packet data from l2_packet_init()

This function is called when authentication is expected to start, e.g., when association has been completed, in order to prepare l2_packet implementation for EAPOL frames. This function is used mainly if the l2_packet code needs to do polling in which case it can increasing polling frequency. This can also be an empty function if the l2_packet implementation does not benefit from knowing about the starting authentication.

Definition at line 121 of file l2_packet_none.c.

#### 6.110.2.6 int l2_packet_send (struct l2_packet_data * *l2*, const u8 * *dst_addr*, u16 *proto*, const u8 * *buf*, size_t *len*)

Send a packet.

**Parameters:**
***l2*** Pointer to internal l2_packet data from l2_packet_init()

***dst_addr*** Destination address for the packet (only used if l2_hdr == 0)

***proto*** Protocol/ethertype for the packet in host byte order (only used if l2_hdr == 0)

***buf*** Packet contents to be sent; including layer 2 header if l2_hdr was set to 1 in l2_packet_init() call. Otherwise, only the payload of the packet is included.

***len*** Length of the buffer (including l2 header only if l2_hdr == 1)

**Returns:**
>=0 on success, <0 on failure

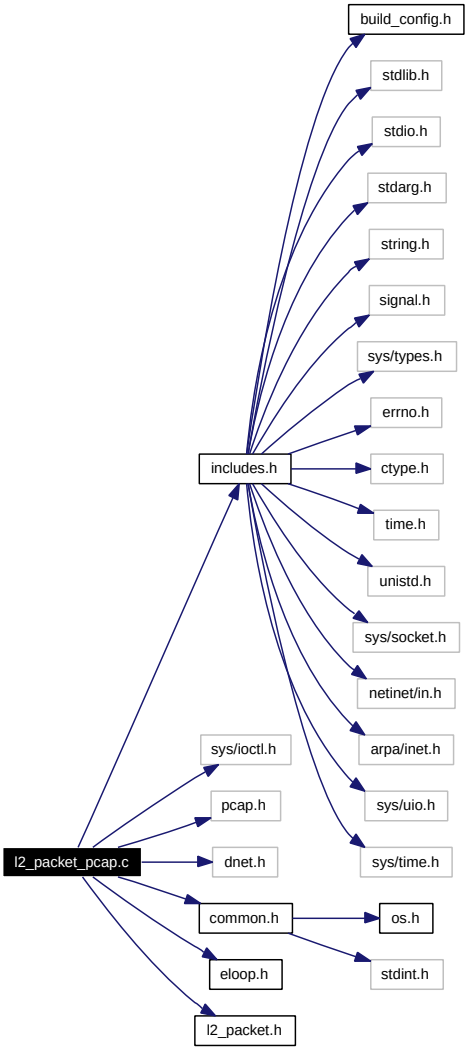
Definition at line 44 of file l2_packet_none.c.

## 6.111 l2_packet_pcap.c File Reference

WPA Supplicant - Layer2 packet handling with libpcap/libdnet and WinPcap.

```
#include "includes.h"
#include <sys/ioctl.h>
#include <pcap.h>
#include <dnet.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

Include dependency graph for l2_packet_pcap.c:



## Data Structures

- struct **l2_packet_data**

## Functions

- int l2_packet_get_own_addr (struct l2_packet_data *l2, u8 *addr)

  *Get own layer 2 address.*

- int l2_packet_send (struct l2_packet_data *l2, const u8 *dst_addr, u16 proto, const u8 *buf, size_t len)

  *Send a packet.*

- l2_packet_data * l2_packet_init (const char *ifname, const u8 *own_addr, unsigned short protocol, void(*rx_callback)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len), void *rx_callback_ctx, int l2_hdr)

  *Initialize l2_packet interface.*

- void l2_packet_deinit (struct l2_packet_data *l2)

  *Deinitialize l2_packet interface.*

- int l2_packet_get_ip_addr (struct l2_packet_data *l2, char *buf, size_t len)

  *Get the current IP address from the interface.*

- void l2_packet_notify_auth_start (struct l2_packet_data *l2)

  *Notify l2_packet about start of authentication.*

### 6.111.1 Detailed Description

WPA Supplicant - Layer2 packet handling with libpcap/libdnet and WinPcap.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file l2_packet_pcap.c.

### 6.111.2 Function Documentation

#### 6.111.2.1 void l2_packet_deinit (struct l2_packet_data * *l2*)

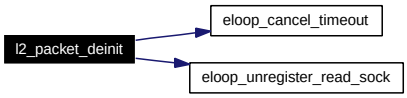
Deinitialize l2_packet interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

Definition at line 321 of file l2_packet_pcap.c.

Here is the call graph for this function:




#### 6.111.2.2 int l2_packet_get_ip_addr (struct l2_packet_data * *l2*, char * *buf*, size_t *len*)

Get the current IP address from the interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***buf*** Buffer for the IP address in text format

***len*** Maximum buffer length

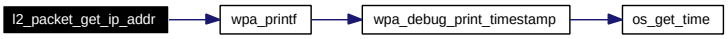
**Returns:**

0 on success, -1 on failure

This function can be used to get the current IP address from the interface bound to the l2_packet. This is mainly for status information and the IP address will be stored as an ASCII string. This function is not essential for wpa_supplicant operation, so full implementation is not required. l2_packet implementation will need to define the function, but it can return -1 if the IP address information is not available.

Definition at line 339 of file l2_packet_pcap.c.

Here is the call graph for this function:




#### 6.111.2.3 int l2_packet_get_own_addr (struct l2_packet_data * *l2*, u8 * *addr*)

Get own layer 2 address.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***addr*** Buffer for the own address (6 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 50 of file l2_packet_pcap.c.

#### 6.111.2.4 struct l2_packet_data* l2_packet_init (const char * *ifname*, const u8 * *own_addr*, unsigned short *protocol*, void(*)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len) *rx_callback*, void * *rx_callback_ctx*, int *l2_hdr*)

Initialize l2_packet interface.

**Parameters:**

***ifname*** Interface name

***own_addr*** Optional own MAC address if available from driver interface or NULL if not available

***protocol*** Ethernet protocol number in host byte order

***rx_callback*** Callback function that will be called for each received packet

***rx_callback_ctx*** Callback data (ctx) for calls to rx_callback()

***l2_hdr*** 1 = include layer 2 header, 0 = do not include header

**Returns:**

Pointer to internal data or NULL on failure

rx_callback function will be called with src_addr pointing to the source address (MAC address) of the the packet. If l2_hdr is set to 0, buf points to len bytes of the payload after the layer 2 header and similarly, TX buffers start with payload. This behavior can be changed by setting l2_hdr=1 to include the layer 2 header in the data buffer.

Definition at line 285 of file l2_packet_pcap.c.

Here is the call graph for this function:


#### 6.111.2.5 void l2_packet_notify_auth_start (struct l2_packet_data * *l2*)

Notify l2_packet about start of authentication.

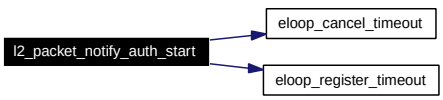
**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

This function is called when authentication is expected to start, e.g., when association has been completed, in order to prepare l2_packet implementation for EAPOL frames. This function is used mainly if the l2_packet code needs to do polling in which case it can increasing polling frequency. This can also be an empty function if the l2_packet implementation does not benefit from knowing about the starting authentication.

Definition at line 375 of file l2_packet_pcap.c.

Here is the call graph for this function:



#### 6.111.2.6 int l2_packet_send (struct l2_packet_data * *l2*, const u8 * *dst_addr*, u16 *proto*, const u8 * *buf*, size_t *len*)

Send a packet.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***dst_addr*** Destination address for the packet (only used if l2_hdr == 0)

***proto*** Protocol/ethertype for the packet in host byte order (only used if l2_hdr == 0)

***buf*** Packet contents to be sent; including layer 2 header if l2_hdr was set to 1 in l2_packet_init() call. Otherwise, only the payload of the packet is included.

***len*** Length of the buffer (including l2 header only if l2_hdr == 1)

**Returns:**

>=0 on success, <0 on failure

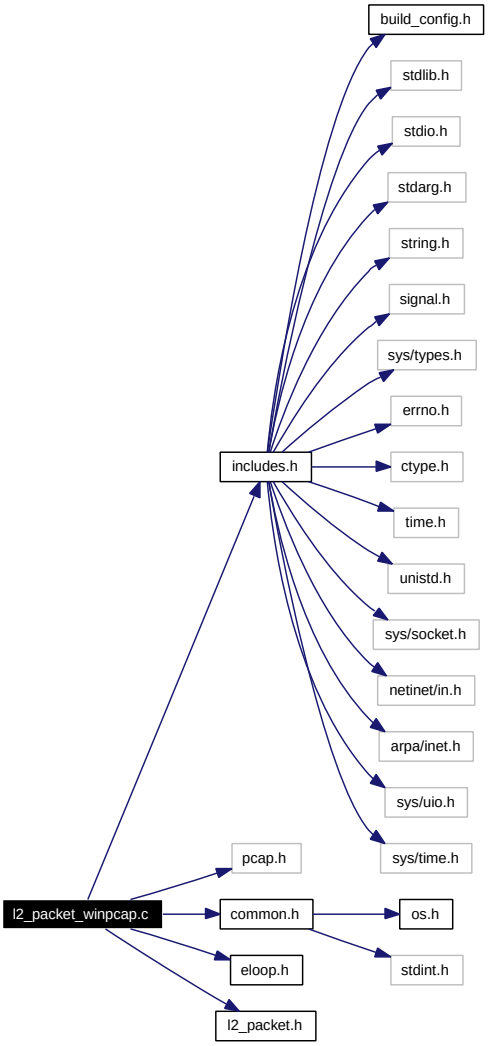
Definition at line 84 of file l2_packet_pcap.c.

# 6.112 l2_packet_winpcap.c File Reference

WPA Supplicant - Layer2 packet handling with WinPcap RX thread.

```
#include "includes.h"
#include <pcap.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

Include dependency graph for l2_packet_winpcap.c:




## Data Structures

- struct **l2_packet_data**

## Functions

- int l2_packet_get_own_addr (struct l2_packet_data *l2, u8 *addr)

  *Get own layer 2 address.*

- int l2_packet_send (struct l2_packet_data *l2, const u8 *dst_addr, u16 proto, const u8 *buf, size_t len)

  *Send a packet.*

- l2_packet_data * l2_packet_init (const char *ifname, const u8 *own_addr, unsigned short protocol, void(*rx_callback)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len), void *rx_callback_ctx, int l2_hdr)

  *Initialize l2_packet interface.*

- void l2_packet_deinit (struct l2_packet_data *l2)

  *Deinitialize l2_packet interface.*

- int l2_packet_get_ip_addr (struct l2_packet_data *l2, char *buf, size_t len)

  *Get the current IP address from the interface.*

- void l2_packet_notify_auth_start (struct l2_packet_data *l2)

  *Notify l2_packet about start of authentication.*

### 6.112.1 Detailed Description

WPA Supplicant - Layer2 packet handling with WinPcap RX thread.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This l2_packet implementation is explicitly for WinPcap and Windows events. l2_packet_pcap.c has support for WinPcap, but it requires polling to receive frames which means relatively long latency for EAPOL RX processing. The implementation here uses a separate thread to allow WinPcap to be receiving all the time to reduce latency for EAPOL receiving from about 100 ms to 3 ms when comparing l2_packet_pcap.c to l2_packet_winpcap.c. Extra sleep of 50 ms is added in to receive thread whenever no EAPOL frames has been received for a while. Whenever an EAPOL handshake is expected, this sleep is removed.

The RX thread receives a frame and signals main thread through Windows event about the availability of a new frame. Processing the received frame is synchronized with pair of Windows events so that no extra buffer or queuing mechanism is needed. This implementation requires Windows specific event loop implementation, i.e., eloop_win.c.

WinPcap has pcap_getevent() that could, in theory at least, be used to implement this kind of waiting with a simpler single-thread design. However, that event handle is not really signaled immediately when receiving each frame, so it does not really work for this kind of use.

Definition in file l2_packet_winpcap.c.

## 6.112.2 Function Documentation

### 6.112.2.1 void l2_packet_deinit (struct l2_packet_data * *l2*)

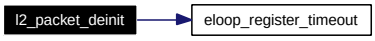
Deinitialize l2_packet interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

Definition at line 280 of file l2_packet_winpcap.c.

Here is the call graph for this function:




### 6.112.2.2 int l2_packet_get_ip_addr (struct l2_packet_data * *l2*, char * *buf*, size_t *len*)

Get the current IP address from the interface.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***buf*** Buffer for the IP address in text format

***len*** Maximum buffer length

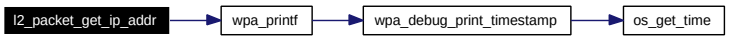
**Returns:**

0 on success, -1 on failure

This function can be used to get the current IP address from the interface bound to the l2_packet. This is mainly for status information and the IP address will be stored as an ASCII string. This function is not essential for wpa_supplicant operation, so full implementation is not required. l2_packet implementation will need to define the function, but it can return -1 if the IP address information is not available.

Definition at line 302 of file l2_packet_winpcap.c.

Here is the call graph for this function:




### 6.112.2.3 int l2_packet_get_own_addr (struct l2_packet_data * *l2*, u8 * *addr*)

Get own layer 2 address.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***addr*** Buffer for the own address (6 bytes)

**Returns:**

0 on success, -1 on failure

Definition at line 72 of file l2_packet_winpcap.c.

#### 6.112.2.4 struct l2_packet_data* l2_packet_init (const char * *ifname*, const u8 * *own_addr*, unsigned short *protocol*, void(*)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len) *rx_callback*, void * *rx_callback_ctx*, int *l2_hdr*)

Initialize l2_packet interface.

**Parameters:**

***ifname*** Interface name

***own_addr*** Optional own MAC address if available from driver interface or NULL if not available

***protocol*** Ethernet protocol number in host byte order

***rx_callback*** Callback function that will be called for each received packet

***rx_callback_ctx*** Callback data (ctx) for calls to rx_callback()

***l2_hdr*** 1 = include layer 2 header, 0 = do not include header

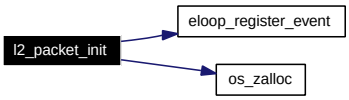
**Returns:**

Pointer to internal data or NULL on failure

rx_callback function will be called with src_addr pointing to the source address (MAC address) of the the packet. If l2_hdr is set to 0, buf points to len bytes of the payload after the layer 2 header and similarly, TX buffers start with payload. This behavior can be changed by setting l2_hdr=1 to include the layer 2 header in the data buffer.

Definition at line 205 of file l2_packet_winpcap.c.

Here is the call graph for this function:




#### 6.112.2.5 void l2_packet_notify_auth_start (struct l2_packet_data * *l2*)

Notify l2_packet about start of authentication.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

This function is called when authentication is expected to start, e.g., when association has been completed, in order to prepare l2_packet implementation for EAPOL frames. This function is used mainly if the l2_packet code needs to do polling in which case it can increasing polling frequency. This can also be an empty function if the l2_packet implementation does not benefit from knowing about the starting authentication.

Definition at line 338 of file l2_packet_winpcap.c.

#### 6.112.2.6 int l2_packet_send (struct l2_packet_data * *l2*, const u8 * *dst_addr*, u16 *proto*, const u8 * *buf*, size_t *len*)

Send a packet.

**Parameters:**

***l2*** Pointer to internal l2_packet data from l2_packet_init()

***dst_addr*** Destination address for the packet (only used if l2_hdr == 0)

***proto*** Protocol/ethertype for the packet in host byte order (only used if l2_hdr == 0)

***buf*** Packet contents to be sent; including layer 2 header if l2_hdr was set to 1 in l2_packet_init() call. Otherwise, only the payload of the packet is included.

***len*** Length of the buffer (including l2 header only if l2_hdr == 1)

**Returns:**

>=0 on success, <0 on failure

Definition at line 79 of file l2_packet_winpcap.c.

## 6.113 libtommath.c File Reference

Minimal code for RSA support from LibTomMath 0.3.9 http://math.libtomcrypt.com/ http://math.libtomcrypt.com/files/ltm-0.39.tar.bz2 This library was released in public domain by Tom St Denis.

### Defines

- #define **CHAR_BIT** 8
- #define **BN_MP_INVMOD_C**
- #define **BN_S_MP_EXPTMOD_C**
- #define **BN_S_MP_MUL_DIGS_C**
- #define **BN_MP_INVMOD_SLOW_C**
- #define **BN_S_MP_SQR_C**
- #define **BN_S_MP_MUL_HIGH_DIGS_C**
- #define **MIN**(x, y) ((x)<(y)?(x):(y))
- #define **MAX**(x, y) ((x)>(y)?(x):(y))
- #define **OPT_CAST**(x)
- #define **DIGIT_BIT** 28
- #define **MP_28BIT**
- #define **XMALLOC** os_malloc
- #define **XFREE** os_free
- #define **XREALLOC** os_realloc
- #define **MP_MASK** ((((mp_digit)1)<<((mp_digit)DIGIT_BIT))-((mp_digit)1))
- #define **MP_LT** -1
- #define **MP_EQ** 0
- #define **MP_GT** 1
- #define **MP_ZPOS** 0
- #define **MP_NEG** 1
- #define **MP_OKAY** 0
- #define **MP_MEM** -2
- #define **MP_VAL** -3
- #define **MP_YES** 1
- #define **MP_NO** 0
- #define **MP_LOW_MEM**
- #define **MP_PREC** 8
- #define **MP_WARRAY** (1 << (sizeof(mp_word) * CHAR_BIT - 2 * DIGIT_BIT + 1))
- #define **mp_iszero**(a) (((a) → used == 0) ? MP_YES : MP_NO)
- #define **mp_iseven**(a) (((a) → used > 0 && (((a) → dp[0] & 1) == 0)) ? MP_YES : MP_NO)
- #define **mp_isodd**(a) (((a) → used > 0 && (((a) → dp[0] & 1) == 1)) ? MP_YES : MP_NO)
- #define **s_mp_mul**(a, b, c) s_mp_mul_digs(a, b, c, (a) → used + (b) → used + 1)
- #define **TAB_SIZE** 32

### Typedefs

- typedef unsigned long **mp_digit**
- typedef u64 **mp_word**
- typedef int **mp_err**

### 6.113.1 Detailed Description

Minimal code for RSA support from LibTomMath 0.3.9 http://math.libtomcrypt.com/ http://math.libtomcrypt.com/files/ltm-0.39.tar.bz2 This library was released in public domain by Tom St Denis.

The combination in this file is not using many of the optimized algorithms (e.g., Montgomery reduction) and is considerable slower than the LibTomMath with its default of SC_RSA_1 settins. The main purpose of having this version here is to make it easier to build bignum.c wrapper without having to install and build an external library. However, it is likely worth the effort to use the full library with SC_RSA_1 instead of this minimized copy. Including the optimized algorithms may increase the size requirements by 15 kB or so (measured with x86 build).

If CONFIG_INTERNAL_LIBTOMMATH is defined, bignum.c includes this libtommath.c file instead of using the external LibTomMath library.

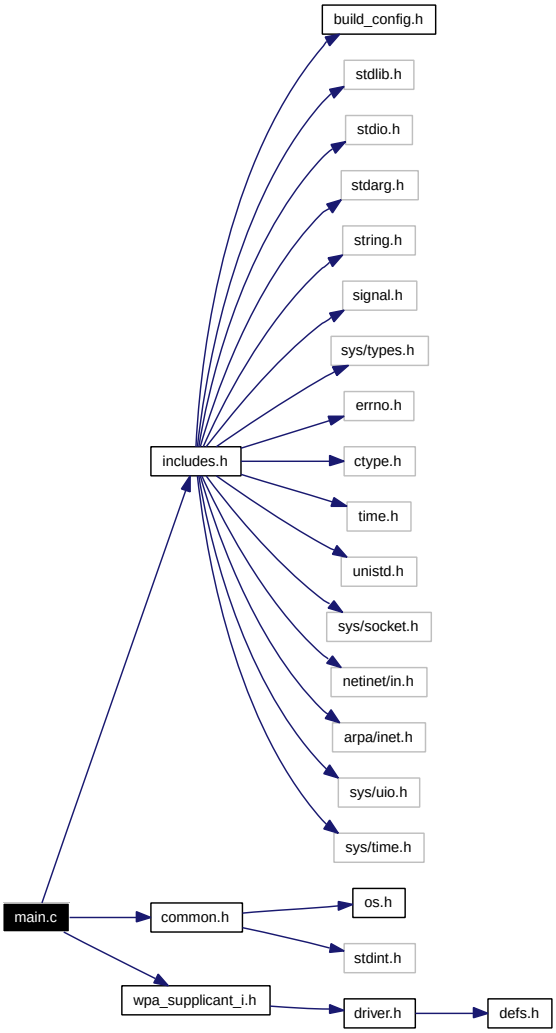
Definition in file libtommath.c.

## 6.114 main.c File Reference

WPA Supplicant / main() function for UNIX like OSes and MinGW.

```
#include "includes.h"
#include "common.h"
#include "wpa_supplicant_i.h"
```

Include dependency graph for main.c:




### Functions

- int **main** (int argc, char *argv[ ])

### Variables

- const char * **wpa_supplicant_version**

- const char * **wpa_supplicant_license**
- const char * **wpa_supplicant_full_license1**
- const char * **wpa_supplicant_full_license2**
- const char * **wpa_supplicant_full_license3**
- const char * **wpa_supplicant_full_license4**
- const char * **wpa_supplicant_full_license5**
- wpa_driver_ops * **wpa_supplicant_drivers** [ ]

### 6.114.1 Detailed Description

WPA Supplicant / main() function for UNIX like OSes and MinGW.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

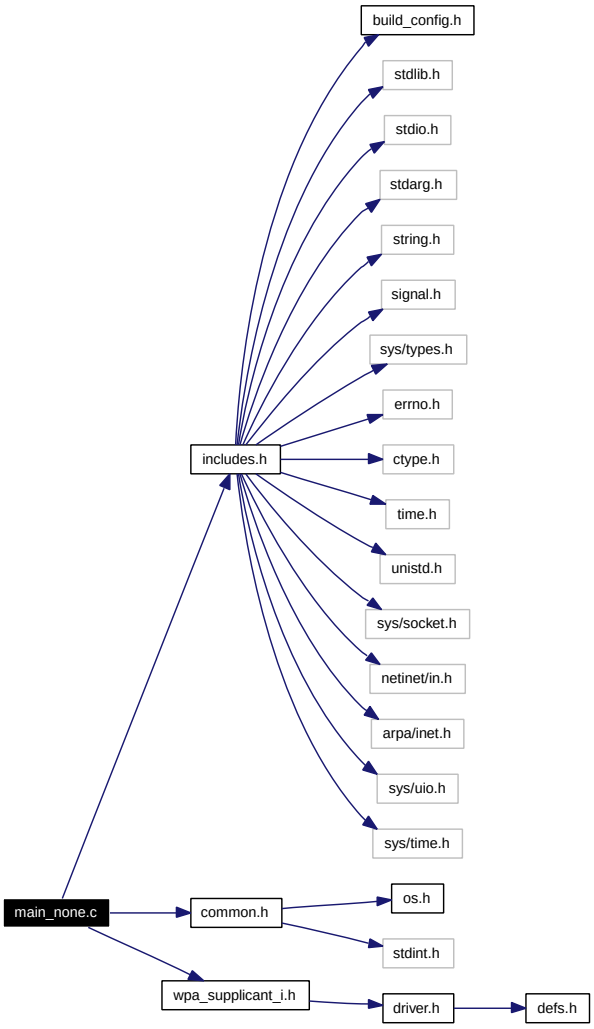
Definition in file main.c.

## 6.115 main_none.c File Reference

WPA Supplicant / Example program entrypoint.

```
#include "includes.h"
#include "common.h"
#include "wpa_supplicant_i.h"
```

Include dependency graph for main_none.c:




### Functions

- int **main** (int argc, char ∗argv[ ])

### 6.115.1 Detailed Description

WPA Supplicant / Example program entrypoint.

**Copyright**



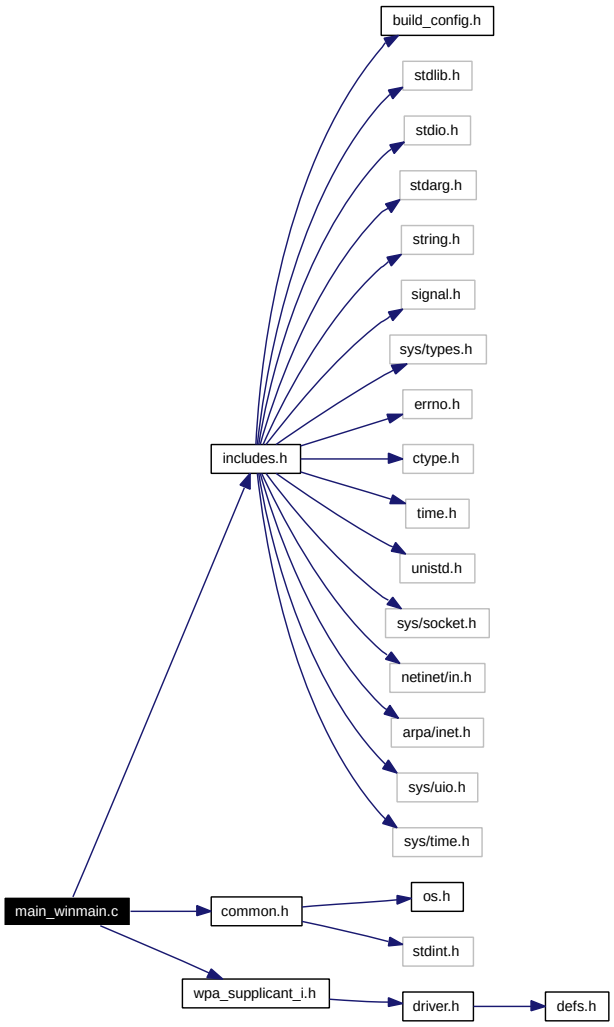
Definition in file main_none.c.

## 6.116 main_winmain.c File Reference

WPA Supplicant / WinMain() function for Windows-based applications.

```
#include "includes.h"
#include "common.h"
#include "wpa_supplicant_i.h"
```

Include dependency graph for main_winmain.c:




### Defines

- #define **CMDLINE** LPSTR

### Functions

- int WINAPI **WinMain** (HINSTANCE hInstance, HINSTANCE hPrevInstance, CMDLINE lpCmdLine, int nShowCmd)

### 6.116.1 Detailed Description

WPA Supplicant / WinMain() function for Windows-based applications.

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

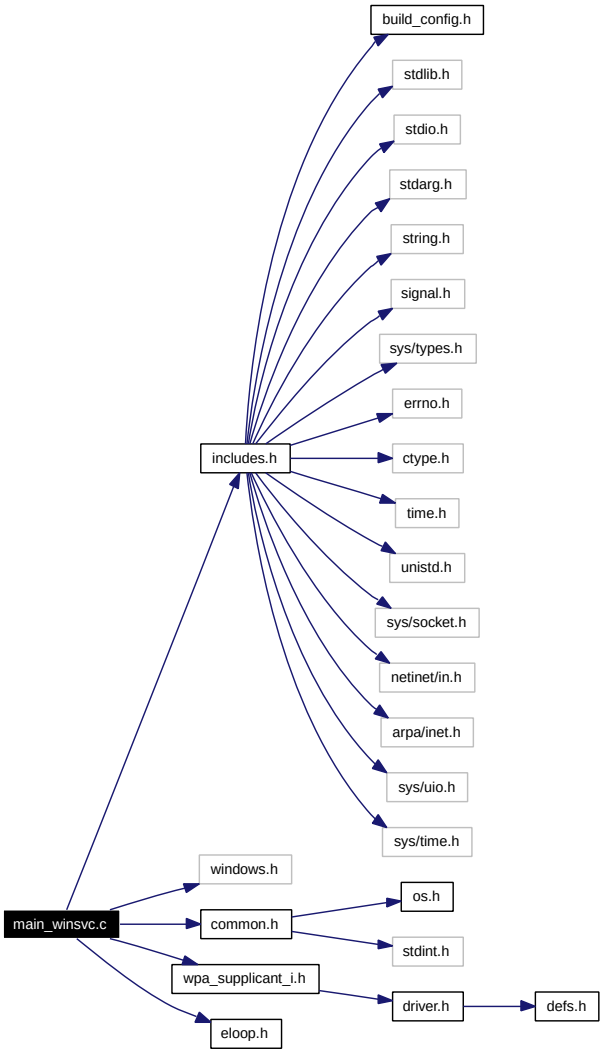
Definition in file main_winmain.c.

## 6.117 main_winsvc.c File Reference

WPA Supplicant / main() function for Win32 service.

```
#include "includes.h"
#include <windows.h>
#include "common.h"
#include "wpa_supplicant_i.h"
#include "eloop.h"
```

Include dependency graph for main_winsvc.c:




### Defines

- #define **WPASVC_NAME** TEXT("wpasvc")
- #define **WPASVC_DISPLAY_NAME** TEXT("wpa_supplicant service")

- #define **WPASVC_DESCRIPTION** TEXT("Provides IEEE 802.1X and WPA/WPA2 supplicant functionality")
- #define **WPA_KEY_ROOT** HKEY_LOCAL_MACHINE
- #define **WPA_KEY_PREFIX** TEXT("SOFTWARE\\wpa_supplicant")
- #define **TSTR** "%s"
- #define **TBUFLEN** 255

## Functions

- int **main** (int argc, char *argv[ ])

### 6.117.1 Detailed Description

WPA Supplicant / main() function for Win32 service.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

The root of wpa_supplicant configuration in registry is HKEY_LOCAL_MACHINE. This level includes global parameters and a 'interfaces' subkey with all the interface configuration (adapter to confname mapping). Each such mapping is a subkey that has 'adapter' and 'config' values.

This program can be run either as a normal command line application, e.g., for debugging, with 'wpasvc.exe app' or as a Windows service. Service need to be registered with 'wpasvc.exe reg <full path to wpasvc.exe>'. After this, it can be started like any other Windows service (e.g., 'net start wpasvc') or it can be configured to start automatically through the Services tool in administrative tasks. The service can be unregistered with 'wpasvc.exe unreg'.

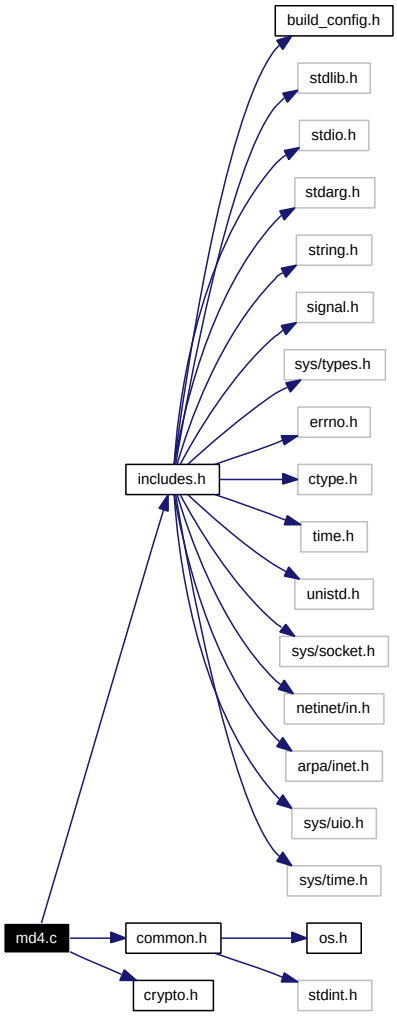
Definition in file main_winsvc.c.

## 6.118 md4.c File Reference

MD4 hash implementation.

```
#include "includes.h"
#include "common.h"
#include "crypto.h"
```

Include dependency graph for md4.c:




### 6.118.1 Detailed Description

MD4 hash implementation.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

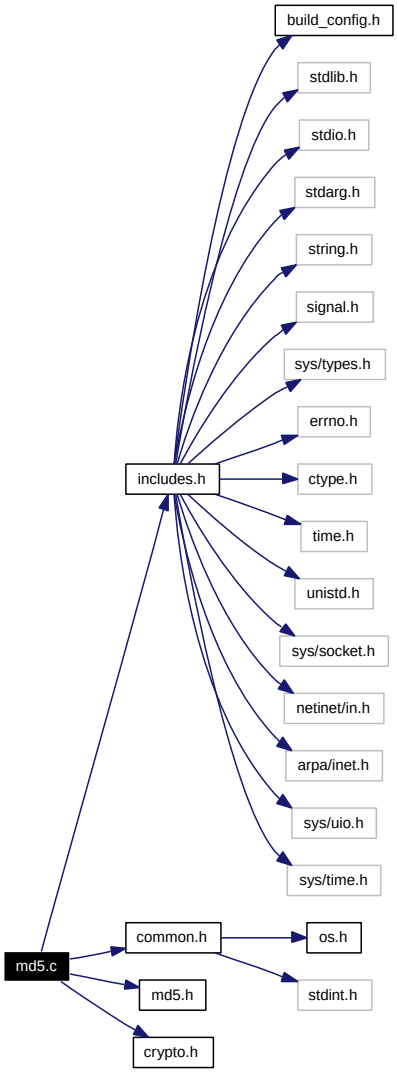
Definition in file md4.c.

## 6.119 md5.c File Reference

MD5 hash implementation and interface functions.

```
#include "includes.h"
#include "common.h"
#include "md5.h"
#include "crypto.h"
```

Include dependency graph for md5.c:




## Functions

- void hmac_md5_vector (const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *HMAC-MD5 over data vector (RFC 2104).*

- void hmac_md5 (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)
  *HMAC-MD5 over data buffer (RFC 2104).*

## 6.119.1 Detailed Description

MD5 hash implementation and interface functions.

**Copyright**
Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file md5.c.

## 6.119.2 Function Documentation

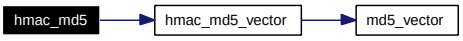
### 6.119.2.1 void hmac_md5 (const u8 * *key*, size_t *key_len*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

HMAC-MD5 over data buffer (RFC 2104).

**Parameters:**
***key*** Key for HMAC operations
***key_len*** Length of the key in bytes
***data*** Pointers to the data area
***data_len*** Length of the data area
***mac*** Buffer for the hash (16 bytes)

Definition at line 106 of file md5.c.

Here is the call graph for this function:




### 6.119.2.2 void hmac_md5_vector (const u8 * *key*, size_t *key_len*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

HMAC-MD5 over data vector (RFC 2104).

**Parameters:**
***key*** Key for HMAC operations
***key_len*** Length of the key in bytes

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash (16 bytes)

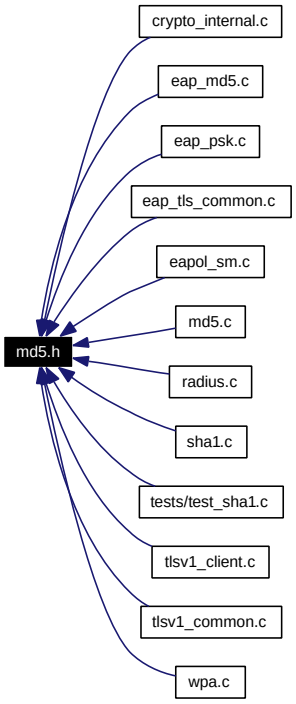
Definition at line 33 of file md5.c.

Here is the call graph for this function:

# 6.120 md5.h File Reference

MD5 hash implementation and interface functions.

This graph shows which files directly or indirectly include this file:




## Defines

- #define **MD5_MAC_LEN** 16

## Functions

- void hmac_md5_vector (const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *HMAC-MD5 over data vector (RFC 2104).*

- void hmac_md5 (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)

  *HMAC-MD5 over data buffer (RFC 2104).*

### 6.120.1 Detailed Description

MD5 hash implementation and interface functions.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file md5.h.

## 6.120.2 Function Documentation

### 6.120.2.1 void hmac_md5 (const u8 * *key*, size_t *key_len*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

HMAC-MD5 over data buffer (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

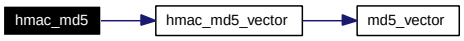
***data*** Pointers to the data area

***data_len*** Length of the data area

***mac*** Buffer for the hash (16 bytes)

Definition at line 106 of file md5.c.

Here is the call graph for this function:




### 6.120.2.2 void hmac_md5_vector (const u8 * *key*, size_t *key_len*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

HMAC-MD5 over data vector (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash (16 bytes)

Definition at line 33 of file md5.c.

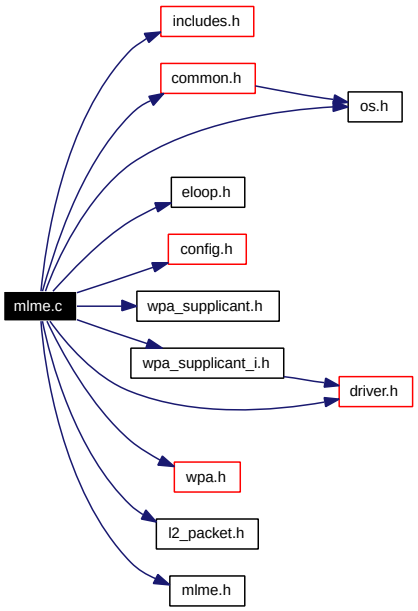
Here is the call graph for this function:

# 6.121 mlme.c File Reference

WPA Supplicant - Client mode MLME.

```
#include "includes.h"
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "wpa_supplicant.h"
#include "wpa_supplicant_i.h"
#include "wpa.h"
#include "os.h"
#include "l2_packet.h"
#include "driver.h"
#include "mlme.h"
```

Include dependency graph for mlme.c:




## Defines

- #define **IEEE80211_AUTH_TIMEOUT** (200)
- #define **IEEE80211_AUTH_MAX_TRIES** 3
- #define **IEEE80211_ASSOC_TIMEOUT** (200)
- #define **IEEE80211_ASSOC_MAX_TRIES** 3
- #define **IEEE80211_MONITORING_INTERVAL** (2000)
- #define **IEEE80211_PROBE_INTERVAL** (60000)
- #define **IEEE80211_RETRY_AUTH_INTERVAL** (1000)

- #define **IEEE80211_SCAN_INTERVAL** (2000)
- #define **IEEE80211_SCAN_INTERVAL_SLOW** (15000)
- #define **IEEE80211_IBSS_JOIN_TIMEOUT** (20000)
- #define **IEEE80211_PROBE_DELAY** (33)
- #define **IEEE80211_CHANNEL_TIME** (33)
- #define **IEEE80211_PASSIVE_CHANNEL_TIME** (200)
- #define **IEEE80211_SCAN_RESULT_EXPIRE** (10000)
- #define **IEEE80211_IBSS_MERGE_INTERVAL** (30000)
- #define **IEEE80211_IBSS_INACTIVITY_LIMIT** (60000)
- #define **IEEE80211_IBSS_MAX_STA_ENTRIES** 128
- #define **WLAN_EID_SSID** 0
- #define **WLAN_EID_SUPP_RATES** 1
- #define **WLAN_EID_FH_PARAMS** 2
- #define **WLAN_EID_DS_PARAMS** 3
- #define **WLAN_EID_CF_PARAMS** 4
- #define **WLAN_EID_TIM** 5
- #define **WLAN_EID_IBSS_PARAMS** 6
- #define **WLAN_EID_COUNTRY** 7
- #define **WLAN_EID_CHALLENGE** 16
- #define **WLAN_EID_PWR_CONSTRAINT** 32
- #define **WLAN_EID_PWR_CAPABILITY** 33
- #define **WLAN_EID_TPC_REQUEST** 34
- #define **WLAN_EID_TPC_REPORT** 35
- #define **WLAN_EID_SUPPORTED_CHANNELS** 36
- #define **WLAN_EID_CHANNEL_SWITCH** 37
- #define **WLAN_EID_MEASURE_REQUEST** 38
- #define **WLAN_EID_MEASURE_REPORT** 39
- #define **WLAN_EID_QUITE** 40
- #define **WLAN_EID_IBSS_DFS** 41
- #define **WLAN_EID_ERP_INFO** 42
- #define **WLAN_EID_RSN** 48
- #define **WLAN_EID_EXT_SUPP_RATES** 50
- #define **WLAN_EID_VENDOR_SPECIFIC** 221
- #define **WLAN_AUTH_OPEN** 0
- #define **WLAN_AUTH_SHARED_KEY** 1
- #define **WLAN_AUTH_LEAP** 128
- #define **WLAN_AUTH_CHALLENGE_LEN** 128
- #define **WLAN_CAPABILITY_ESS** BIT(0)
- #define **WLAN_CAPABILITY_IBSS** BIT(1)
- #define **WLAN_CAPABILITY_CF_POLLABLE** BIT(2)
- #define **WLAN_CAPABILITY_CF_POLL_REQUEST** BIT(3)
- #define **WLAN_CAPABILITY_PRIVACY** BIT(4)
- #define **WLAN_CAPABILITY_SHORT_PREAMBLE** BIT(5)
- #define **WLAN_CAPABILITY_PBCC** BIT(6)
- #define **WLAN_CAPABILITY_CHANNEL_AGILITY** BIT(7)
- #define **WLAN_CAPABILITY_SPECTRUM_MGMT** BIT(8)
- #define **WLAN_CAPABILITY_SHORT_SLOT_TIME** BIT(10)
- #define **WLAN_CAPABILITY_DSSS_OFDM** BIT(13)
- #define **WLAN_STATUS_SUCCESS** 0
- #define **WLAN_STATUS_UNSPECIFIED_FAILURE** 1

- #define **WLAN_STATUS_CAPS_UNSUPPORTED** 10
- #define **WLAN_STATUS_REASSOC_NO_ASSOC** 11
- #define **WLAN_STATUS_ASSOC_DENIED_UNSPEC** 12
- #define **WLAN_STATUS_NOT_SUPPORTED_AUTH_ALG** 13
- #define **WLAN_STATUS_UNKNOWN_AUTH_TRANSACTION** 14
- #define **WLAN_STATUS_CHALLENGE_FAIL** 15
- #define **WLAN_STATUS_AUTH_TIMEOUT** 16
- #define **WLAN_STATUS_AP_UNABLE_TO_HANDLE_NEW_STA** 17
- #define **WLAN_STATUS_ASSOC_DENIED_RATES** 18
- #define **WLAN_STATUS_ASSOC_DENIED_NOSHORT** 19
- #define **WLAN_STATUS_ASSOC_DENIED_NOPBCC** 20
- #define **WLAN_STATUS_ASSOC_DENIED_NOAGILITY** 21
- #define **WLAN_STATUS_SPEC_MGMT_REQUIRED** 22
- #define **WLAN_STATUS_PWR_CAPABILITY_NOT_VALID** 23
- #define **WLAN_STATUS_SUPPORTED_CHANNEL_NOT_VALID** 24
- #define **WLAN_STATUS_ASSOC_DENOED_NO_SHORT_SLOT_TIME** 25
- #define **WLAN_STATUS_ASSOC_DENOED_NO_ER_PBCC** 26
- #define **WLAN_STATUS_ASSOC_DENOED_NO_DSSS_OFDM** 27
- #define **WLAN_REASON_UNSPECIFIED** 1
- #define **WLAN_REASON_PREV_AUTH_NOT_VALID** 2
- #define **WLAN_REASON_DEAUTH_LEAVING** 3
- #define **WLAN_REASON_DISASSOC_DUE_TO_INACTIVITY** 4
- #define **WLAN_REASON_DISASSOC_AP_BUSY** 5
- #define **WLAN_REASON_CLASS2_FRAME_FROM_NONAUTH_STA** 6
- #define **WLAN_REASON_CLASS3_FRAME_FROM_NONASSOC_STA** 7
- #define **WLAN_REASON_DISASSOC_STA_HAS_LEFT** 8
- #define **WLAN_REASON_STA_REQ_ASSOC_WITHOUT_AUTH** 9
- #define **WLAN_REASON_PWR_CAPABILITY_NOT_VALID** 10
- #define **WLAN_REASON_SUPPORTED_CHANNEL_NOT_VALID** 11
- #define **WLAN_REASON_MIC_FAILURE** 14
- #define **WLAN_FC_PVER** 0x0003
- #define **WLAN_FC_TODS** 0x0100
- #define **WLAN_FC_FROMDS** 0x0200
- #define **WLAN_FC_MOREFRAG** 0x0400
- #define **WLAN_FC_RETRY** 0x0800
- #define **WLAN_FC_PWRMGT** 0x1000
- #define **WLAN_FC_MOREDATA** 0x2000
- #define **WLAN_FC_ISWEP** 0x4000
- #define **WLAN_FC_ORDER** 0x8000
- #define **WLAN_FC_GET_TYPE**(fc) (((fc) & 0x000c) >> 2)
- #define **WLAN_FC_GET_STYPE**(fc) (((fc) & 0x00f0) >> 4)
- #define **IEEE80211_FC**(type, stype) host_to_le16((type << 2) | (stype << 4))
- #define **WLAN_FC_TYPE_MGMT** 0
- #define **WLAN_FC_TYPE_CTRL** 1
- #define **WLAN_FC_TYPE_DATA** 2
- #define **WLAN_FC_STYPE_ASSOC_REQ** 0
- #define **WLAN_FC_STYPE_ASSOC_RESP** 1
- #define **WLAN_FC_STYPE_REASSOC_REQ** 2
- #define **WLAN_FC_STYPE_REASSOC_RESP** 3
- #define **WLAN_FC_STYPE_PROBE_REQ** 4

- #define **WLAN_FC_STYPE_PROBE_RESP** 5
- #define **WLAN_FC_STYPE_BEACON** 8
- #define **WLAN_FC_STYPE_ATIM** 9
- #define **WLAN_FC_STYPE_DISASSOC** 10
- #define **WLAN_FC_STYPE_AUTH** 11
- #define **WLAN_FC_STYPE_DEAUTH** 12
- #define **WLAN_FC_STYPE_ACTION** 13
- #define **ERP_INFO_USE_PROTECTION** BIT(1)
- #define **IEEE80211_MAX_SUPP_RATES** 32

## Enumerations

- enum **ParseRes** { **ParseOK** = 0, **ParseUnknown** = 1, **ParseFailed** = -1 }

## Functions

- int **ieee80211_sta_get_ssid** (struct wpa_supplicant *wpa_s, u8 *ssid, size_t *len)
- int **ieee80211_sta_associate** (struct wpa_supplicant *wpa_s, struct wpa_driver_associate_params *params)
- int **ieee80211_sta_req_scan** (struct wpa_supplicant *wpa_s, const u8 *ssid, size_t ssid_len)
- int **ieee80211_sta_get_scan_results** (struct wpa_supplicant *wpa_s, struct wpa_scan_result *results, size_t max_size)
- int **ieee80211_sta_deauthenticate** (struct wpa_supplicant *wpa_s, u16 reason)
- int **ieee80211_sta_disassociate** (struct wpa_supplicant *wpa_s, u16 reason)
- void **ieee80211_sta_rx** (struct wpa_supplicant *wpa_s, const u8 *buf, size_t len, struct ieee80211_rx_status *rx_status)
- void **ieee80211_sta_free_hw_features** (struct wpa_hw_modes *hw_features, size_t num_hw_features)
- int **ieee80211_sta_init** (struct wpa_supplicant *wpa_s)
- void **ieee80211_sta_deinit** (struct wpa_supplicant *wpa_s)

## Variables

- ieee80211_mgmt **STRUCT_PACKED**

### 6.121.1 Detailed Description

WPA Supplicant - Client mode MLME.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi> Copyright (c) 2004, Instant802 Networks, Inc. Copyright (c) 2005-2006, Devicescape Software, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

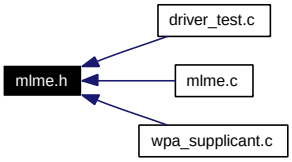
See README and COPYING for more details.

Definition in file mlme.c.

# 6.122 mlme.h File Reference

WPA Supplicant - Client mode MLME.

This graph shows which files directly or indirectly include this file:




## 6.122.1 Detailed Description

WPA Supplicant - Client mode MLME.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi> Copyright (c) 2004, Instant802 Networks, Inc. Copyright (c) 2005-2006, Devicescape Software, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

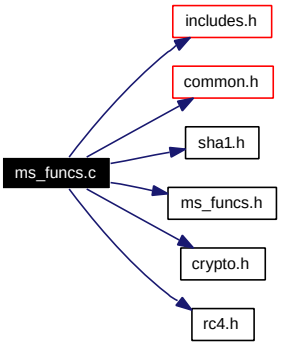
Definition in file mlme.h.

## 6.123 ms_funcs.c File Reference

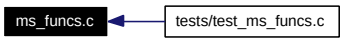
WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759.

```
#include "includes.h"
#include "common.h"
#include "sha1.h"
#include "ms_funcs.h"
#include "crypto.h"
#include "rc4.h"
```

Include dependency graph for ms_funcs.c:




This graph shows which files directly or indirectly include this file:




### Defines

- #define **PWBLOCK_LEN** 516

### Functions

- void nt_password_hash (const u8 *password, size_t password_len, u8 *password_hash)
  *NtPasswordHash() - RFC 2759, Sect. 8.3.*
- void hash_nt_password_hash (const u8 *password_hash, u8 *password_hash_hash)
  *HashNtPasswordHash() - RFC 2759, Sect. 8.4.*
- void challenge_response (const u8 *challenge, const u8 *password_hash, u8 *response)
  *ChallengeResponse() - RFC 2759, Sect. 8.5.*
- void generate_nt_response (const u8 *auth_challenge, const u8 *peer_challenge, const u8 *username, size_t username_len, const u8 *password, size_t password_len, u8 *response)
  *GenerateNTResponse() - RFC 2759, Sect. 8.1.*

- void generate_nt_response_pwhash (const u8 *auth_challenge, const u8 *peer_challenge, const u8 *username, size_t username_len, const u8 *password_hash, u8 *response)

  *GenerateNTResponse() - RFC 2759, Sect. 8.1.*

- void generate_authenticator_response_pwhash (const u8 *password_hash, const u8 *peer_challenge, const u8 *auth_challenge, const u8 *username, size_t username_len, const u8 *nt_response, u8 *response)

  *GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.*

- void generate_authenticator_response (const u8 *password, size_t password_len, const u8 *peer_challenge, const u8 *auth_challenge, const u8 *username, size_t username_len, const u8 *nt_response, u8 *response)

  *GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.*

- void nt_challenge_response (const u8 *challenge, const u8 *password, size_t password_len, u8 *response)

  *NtChallengeResponse() - RFC 2433, Sect. A.5.*

- void get_master_key (const u8 *password_hash_hash, const u8 *nt_response, u8 *master_key)

  *GetMasterKey() - RFC 3079, Sect. 3.4.*

- void get_asymetric_start_key (const u8 *master_key, u8 *session_key, size_t session_key_len, int is_send, int is_server)

  *GetAsymetricStartKey() - RFC 3079, Sect. 3.4.*

- void new_password_encrypted_with_old_nt_password_hash (const u8 *new_password, size_t new_password_len, const u8 *old_password, size_t old_password_len, u8 *encrypted_pw_block)

  *NewPasswordEncryptedWithOldNtPasswordHash() - RFC 2759, Sect. 8.9.*

- void old_nt_password_hash_encrypted_with_new_nt_password_hash (const u8 *new_password, size_t new_password_len, const u8 *old_password, size_t old_password_len, u8 *encrypted_password_hash)

  *OldNtPasswordHashEncryptedWithNewNtPasswordHash() - RFC 2759, Sect. 8.12.*

### 6.123.1 Detailed Description

WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ms_funcs.c.

## 6.123.2 Function Documentation

### 6.123.2.1 void challenge_response (const u8 * *challenge*, const u8 * *password_hash*, u8 * *response*)

ChallengeResponse() - RFC 2759, Sect. 8.5.

**Parameters:**

***challenge*** 8-octet Challenge (IN)

***password_hash*** 16-octet PasswordHash (IN)

***response*** 24-octet Response (OUT)

Definition at line 102 of file ms_funcs.c.

Here is the call graph for this function:


### 6.123.2.2 void generate_authenticator_response (const u8 * *password*, size_t *password_len*, const u8 * *peer_challenge*, const u8 * *auth_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *nt_response*, u8 * *response*)

GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

**Parameters:**

***password*** 0-to-256-unicode-char Password (IN; ASCII)

***password_len*** Length of password

***nt_response*** 24-octet NT-Response (IN)

***peer_challenge*** 16-octet PeerChallenge (IN)

***auth_challenge*** 16-octet AuthenticatorChallenge (IN)

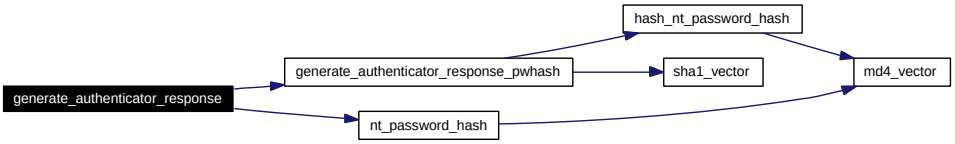
***username*** 0-to-256-char UserName (IN)

***username_len*** Length of username

***response*** 20-octet AuthenticatorResponse (OUT) (note: this value is usually encoded as a 42-octet ASCII string (S=<hexdump of="" response="">)

Definition at line 233 of file ms_funcs.c.

Here is the call graph for this function:



#### 6.123.2.3 void generate_authenticator_response_pwhash (const u8 * *password_hash*, const u8 * *peer_challenge*, const u8 * *auth_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *nt_response*, u8 * *response*)

GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

**Parameters:**

***password_hash*** 16-octet PasswordHash (IN)

***nt_response*** 24-octet NT-Response (IN)

***peer_challenge*** 16-octet PeerChallenge (IN)

***auth_challenge*** 16-octet AuthenticatorChallenge (IN)

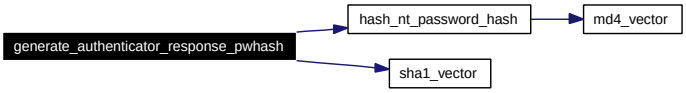
***username*** 0-to-256-char UserName (IN)

***username_len*** Length of username

***response*** 20-octet AuthenticatorResponse (OUT) (note: this value is usually encoded as a 42-octet ASCII string (S=<hexdump of="" response="">)

Definition at line 177 of file ms_funcs.c.

Here is the call graph for this function:




#### 6.123.2.4 void generate_nt_response (const u8 * *auth_challenge*, const u8 * *peer_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *password*, size_t *password_len*, u8 * *response*)

GenerateNTResponse() - RFC 2759, Sect. 8.1.

**Parameters:**

***auth_challenge*** 16-octet AuthenticatorChallenge (IN)

***peer_hallenge*** 16-octet PeerChallenge (IN)

***username*** 0-to-256-char UserName (IN)

***username_len*** Length of username

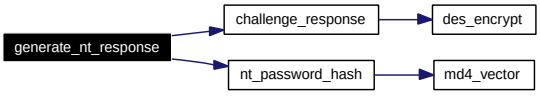
***password*** 0-to-256-unicode-char Password (IN; ASCII)

***password_len*** Length of password

***response*** 24-octet Response (OUT)

Definition at line 126 of file ms_funcs.c.

Here is the call graph for this function:



#### 6.123.2.5 void generate_nt_response_pwhash (const u8 * *auth_challenge*, const u8 * *peer_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *password_hash*, u8 * *response*)

GenerateNTResponse() - RFC 2759, Sect. 8.1.

**Parameters:**

***auth_challenge*** 16-octet AuthenticatorChallenge (IN)

***peer_hallenge*** 16-octet PeerChallenge (IN)

***username*** 0-to-256-char UserName (IN)

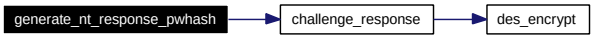
***username_len*** Length of username

***password_hash*** 16-octet PasswordHash (IN)

***response*** 24-octet Response (OUT)

Definition at line 151 of file ms_funcs.c.

Here is the call graph for this function:




#### 6.123.2.6 void get_asymetric_start_key (const u8 * *master_key*, u8 * *session_key*, size_t *session_key_len*, int *is_send*, int *is_server*)

GetAsymetricStartKey() - RFC 3079, Sect. 3.4.

**Parameters:**

***master_key*** 16-octet MasterKey (IN)

***session_key*** 8-to-16 octet SessionKey (OUT)

***session_key_len*** SessionKeyLength (Length of session_key) (IN)

***is_send*** IsSend (IN, BOOLEAN)

***is_server*** IsServer (IN, BOOLEAN)

Definition at line 302 of file ms_funcs.c.

Here is the call graph for this function:


#### 6.123.2.7 void get_master_key (const u8 * *password_hash_hash*, const u8 * *nt_response*, u8 * *master_key*)

GetMasterKey() - RFC 3079, Sect. 3.4.

**Parameters:**

***password_hash_hash*** 16-octet PasswordHashHash (IN)

***nt_response*** 24-octet NTResponse (IN)

***master_key*** 16-octet MasterKey (OUT)

Definition at line 272 of file ms_funcs.c.

Here is the call graph for this function:


#### 6.123.2.8 void hash_nt_password_hash (const u8 * *password_hash*, u8 * *password_hash_hash*)

HashNtPasswordHash() - RFC 2759, Sect. 8.4.

**Parameters:**

***password_hash*** 16-octet PasswordHash (IN)

***password_hash_hash*** 16-octet PasswordHashHash (OUT)

Definition at line 88 of file ms_funcs.c.

Here is the call graph for this function:


#### 6.123.2.9 void new_password_encrypted_with_old_nt_password_hash (const u8 * *new_password*, size_t *new_password_len*, const u8 * *old_password*, size_t *old_password_len*, u8 * *encrypted_pw_block*)

NewPasswordEncryptedWithOldNtPasswordHash() - RFC 2759, Sect. 8.9.

**Parameters:**

***new_password*** 0-to-256-unicode-char NewPassword (IN; ASCII)

***new_password_len*** Length of new_password

***old_password*** 0-to-256-unicode-char OldPassword (IN; ASCII)

***old_password_len*** Length of old_password

***encrypted_pw_block*** 516-octet EncryptedPwBlock (OUT)

Definition at line 406 of file ms_funcs.c.

Here is the call graph for this function:

#### 6.123.2.10 void nt_challenge_response (const u8 * *challenge*, const u8 * *password*, size_t *password_len*, u8 * *response*)

NtChallengeResponse() - RFC 2433, Sect. A.5.

**Parameters:**

***challenge*** 8-octet Challenge (IN)

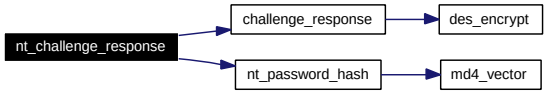
***password*** 0-to-256-unicode-char Password (IN; ASCII)

***password_len*** Length of password

***response*** 24-octet Response (OUT)

Definition at line 256 of file ms_funcs.c.

Here is the call graph for this function:




#### 6.123.2.11 void nt_password_hash (const u8 * *password*, size_t *password_len*, u8 * *password_hash*)

NtPasswordHash() - RFC 2759, Sect. 8.3.

**Parameters:**

***password*** 0-to-256-unicode-char Password (IN; ASCII)

***password_len*** Length of password

***password_hash*** 16-octet PasswordHash (OUT)

Definition at line 61 of file ms_funcs.c.

Here is the call graph for this function:


#### 6.123.2.12 void old_nt_password_hash_encrypted_with_new_nt_password_hash (const u8 * *new_password*, size_t *new_password_len*, const u8 * *old_password*, size_t *old_password_len*, u8 * *encrypted_password_hash*)

OldNtPasswordHashEncryptedWithNewNtPasswordHash() - RFC 2759, Sect. 8.12.

**Parameters:**

***new_password*** 0-to-256-unicode-char NewPassword (IN; ASCII)

***new_password_len*** Length of new_password

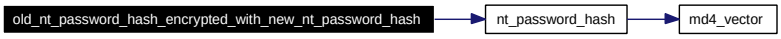
***old_password*** 0-to-256-unicode-char OldPassword (IN; ASCII)

***old_password_len*** Length of old_password

***encrypted_password_ash*** 16-octet EncryptedPasswordHash (OUT)

Definition at line 444 of file ms_funcs.c.

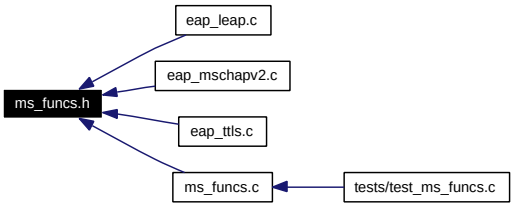
Here is the call graph for this function:



## 6.124 ms_funcs.h File Reference

WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759.

This graph shows which files directly or indirectly include this file:




### Functions

- void generate_nt_response (const u8 *auth_challenge, const u8 *peer_challenge, const u8 *username, size_t username_len, const u8 *password, size_t password_len, u8 *response)

  *GenerateNTResponse() - RFC 2759, Sect. 8.1.*

- void generate_nt_response_pwhash (const u8 *auth_challenge, const u8 *peer_challenge, const u8 *username, size_t username_len, const u8 *password_hash, u8 *response)

  *GenerateNTResponse() - RFC 2759, Sect. 8.1.*

- void generate_authenticator_response (const u8 *password, size_t password_len, const u8 *peer_-challenge, const u8 *auth_challenge, const u8 *username, size_t username_len, const u8 *nt_-response, u8 *response)

  *GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.*

- void generate_authenticator_response_pwhash (const u8 *password_hash, const u8 *peer_-challenge, const u8 *auth_challenge, const u8 *username, size_t username_len, const u8 *nt_-response, u8 *response)

  *GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.*

- void nt_challenge_response (const u8 *challenge, const u8 *password, size_t password_len, u8 *response)

  *NtChallengeResponse() - RFC 2433, Sect. A.5.*

- void challenge_response (const u8 *challenge, const u8 *password_hash, u8 *response)

  *ChallengeResponse() - RFC 2759, Sect. 8.5.*

- void nt_password_hash (const u8 *password, size_t password_len, u8 *password_hash)

  *NtPasswordHash() - RFC 2759, Sect. 8.3.*

- void hash_nt_password_hash (const u8 *password_hash, u8 *password_hash_hash)

  *HashNtPasswordHash() - RFC 2759, Sect. 8.4.*

- void get_master_key (const u8 *password_hash_hash, const u8 *nt_response, u8 *master_key)

  *GetMasterKey() - RFC 3079, Sect. 3.4.*

- void get_asymetric_start_key (const u8 *master_key, u8 *session_key, size_t session_key_len, int is_send, int is_server)

  *GetAsymetricStartKey() - RFC 3079, Sect. 3.4.*

- void new_password_encrypted_with_old_nt_password_hash (const u8 *new_password, size_t new_password_len, const u8 *old_password, size_t old_password_len, u8 *encrypted_pw_block)

  *NewPasswordEncryptedWithOldNtPasswordHash() - RFC 2759, Sect. 8.9.*

- void old_nt_password_hash_encrypted_with_new_nt_password_hash (const u8 *new_password, size_t new_password_len, const u8 *old_password, size_t old_password_len, u8 *encrypted_password_hash)

  *OldNtPasswordHashEncryptedWithNewNtPasswordHash() - RFC 2759, Sect. 8.12.*

## 6.124.1 Detailed Description

WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759.

**Copyright**
Copyright (c) 2004-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file ms_funcs.h.

## 6.124.2 Function Documentation

### 6.124.2.1 void challenge_response (const u8 * *challenge*, const u8 * *password_hash*, u8 * *response*)

ChallengeResponse() - RFC 2759, Sect. 8.5.

**Parameters:**
***challenge*** 8-octet Challenge (IN)

***password_hash*** 16-octet PasswordHash (IN)

***response*** 24-octet Response (OUT)

Definition at line 102 of file ms_funcs.c.

Here is the call graph for this function:

#### 6.124.2.2 void generate_authenticator_response (const u8 * *password*, size_t *password_len*, const u8 * *peer_challenge*, const u8 * *auth_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *nt_response*, u8 * *response*)

GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

**Parameters:**

***password*** 0-to-256-unicode-char Password (IN; ASCII)

***password_len*** Length of password

***nt_response*** 24-octet NT-Response (IN)

***peer_challenge*** 16-octet PeerChallenge (IN)

***auth_challenge*** 16-octet AuthenticatorChallenge (IN)

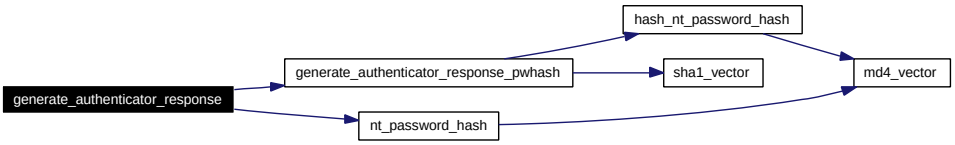
***username*** 0-to-256-char UserName (IN)

***username_len*** Length of username

***response*** 20-octet AuthenticatorResponse (OUT) (note: this value is usually encoded as a 42-octet ASCII string (S=<hexdump of="" response="">)

Definition at line 233 of file ms_funcs.c.

Here is the call graph for this function:




#### 6.124.2.3 void generate_authenticator_response_pwhash (const u8 * *password_hash*, const u8 * *peer_challenge*, const u8 * *auth_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *nt_response*, u8 * *response*)

GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

**Parameters:**

***password_hash*** 16-octet PasswordHash (IN)

***nt_response*** 24-octet NT-Response (IN)

***peer_challenge*** 16-octet PeerChallenge (IN)

***auth_challenge*** 16-octet AuthenticatorChallenge (IN)

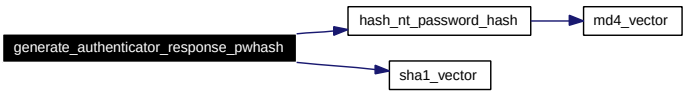
***username*** 0-to-256-char UserName (IN)

***username_len*** Length of username

***response*** 20-octet AuthenticatorResponse (OUT) (note: this value is usually encoded as a 42-octet ASCII string (S=<hexdump of="" response="">)

Definition at line 177 of file ms_funcs.c.

Here is the call graph for this function:




#### 6.124.2.4 void generate_nt_response (const u8 * *auth_challenge*, const u8 * *peer_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *password*, size_t *password_len*, u8 * *response*)

GenerateNTResponse() - RFC 2759, Sect. 8.1.

**Parameters:**

***auth_challenge*** 16-octet AuthenticatorChallenge (IN)

***peer_hallenge*** 16-octet PeerChallenge (IN)

***username*** 0-to-256-char UserName (IN)

***username_len*** Length of username

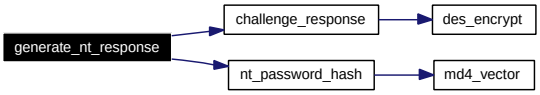
***password*** 0-to-256-unicode-char Password (IN; ASCII)

***password_len*** Length of password

***response*** 24-octet Response (OUT)

Definition at line 126 of file ms_funcs.c.

Here is the call graph for this function:




#### 6.124.2.5 void generate_nt_response_pwhash (const u8 * *auth_challenge*, const u8 * *peer_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *password_hash*, u8 * *response*)

GenerateNTResponse() - RFC 2759, Sect. 8.1.

**Parameters:**

***auth_challenge*** 16-octet AuthenticatorChallenge (IN)

***peer_hallenge*** 16-octet PeerChallenge (IN)

***username*** 0-to-256-char UserName (IN)

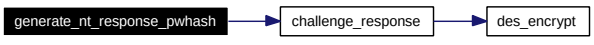
***username_len*** Length of username

***password_hash*** 16-octet PasswordHash (IN)

***response*** 24-octet Response (OUT)

Definition at line 151 of file ms_funcs.c.

Here is the call graph for this function:



#### 6.124.2.6 void get_asymetric_start_key (const u8 * *master_key*, u8 * *session_key*, size_t *session_key_len*, int *is_send*, int *is_server*)

GetAsymetricStartKey() - RFC 3079, Sect. 3.4.

**Parameters:**

***master_key*** 16-octet MasterKey (IN)

***session_key*** 8-to-16 octet SessionKey (OUT)

***session_key_len*** SessionKeyLength (Length of session_key) (IN)

***is_send*** IsSend (IN, BOOLEAN)

***is_server*** IsServer (IN, BOOLEAN)

Definition at line 302 of file ms_funcs.c.

Here is the call graph for this function:


#### 6.124.2.7 void get_master_key (const u8 * *password_hash_hash*, const u8 * *nt_response*, u8 * *master_key*)

GetMasterKey() - RFC 3079, Sect. 3.4.

**Parameters:**

***password_hash_hash*** 16-octet PasswordHashHash (IN)

***nt_response*** 24-octet NTResponse (IN)

***master_key*** 16-octet MasterKey (OUT)

Definition at line 272 of file ms_funcs.c.

Here is the call graph for this function:


#### 6.124.2.8 void hash_nt_password_hash (const u8 * *password_hash*, u8 * *password_hash_hash*)

HashNtPasswordHash() - RFC 2759, Sect. 8.4.

**Parameters:**

***password_hash*** 16-octet PasswordHash (IN)

***password_hash_hash*** 16-octet PasswordHashHash (OUT)

Definition at line 88 of file ms_funcs.c.

Here is the call graph for this function:

#### 6.124.2.9 void new_password_encrypted_with_old_nt_password_hash (const u8 * *new_password*, size_t *new_password_len*, const u8 * *old_password*, size_t *old_password_len*, u8 * *encrypted_pw_block*)

NewPasswordEncryptedWithOldNtPasswordHash() - RFC 2759, Sect. 8.9.

**Parameters:**

***new_password*** 0-to-256-unicode-char NewPassword (IN; ASCII)

***new_password_len*** Length of new_password

***old_password*** 0-to-256-unicode-char OldPassword (IN; ASCII)

***old_password_len*** Length of old_password

***encrypted_pw_block*** 516-octet EncryptedPwBlock (OUT)

Definition at line 406 of file ms_funcs.c.

Here is the call graph for this function:


#### 6.124.2.10 void nt_challenge_response (const u8 * *challenge*, const u8 * *password*, size_t *password_len*, u8 * *response*)

NtChallengeResponse() - RFC 2433, Sect. A.5.

**Parameters:**

***challenge*** 8-octet Challenge (IN)

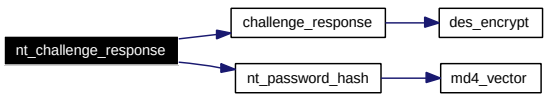
***password*** 0-to-256-unicode-char Password (IN; ASCII)

***password_len*** Length of password

***response*** 24-octet Response (OUT)

Definition at line 256 of file ms_funcs.c.

Here is the call graph for this function:




#### 6.124.2.11 void nt_password_hash (const u8 * *password*, size_t *password_len*, u8 * *password_hash*)

NtPasswordHash() - RFC 2759, Sect. 8.3.

**Parameters:**

***password*** 0-to-256-unicode-char Password (IN; ASCII)

***password_len*** Length of password

***password_hash*** 16-octet PasswordHash (OUT)

Definition at line 61 of file ms_funcs.c.

Here is the call graph for this function:


#### 6.124.2.12 void old_nt_password_hash_encrypted_with_new_nt_password_hash (const u8 * *new_password*, size_t *new_password_len*, const u8 * *old_password*, size_t *old_password_len*, u8 * *encrypted_password_hash*)

OldNtPasswordHashEncryptedWithNewNtPasswordHash() - RFC 2759, Sect. 8.12.

**Parameters:**

***new_password*** 0-to-256-unicode-char NewPassword (IN; ASCII)

***new_password_len*** Length of new_password

***old_password*** 0-to-256-unicode-char OldPassword (IN; ASCII)

***old_password_len*** Length of old_password

***encrypted_password_ash*** 16-octet EncryptedPasswordHash (OUT)

Definition at line 444 of file ms_funcs.c.

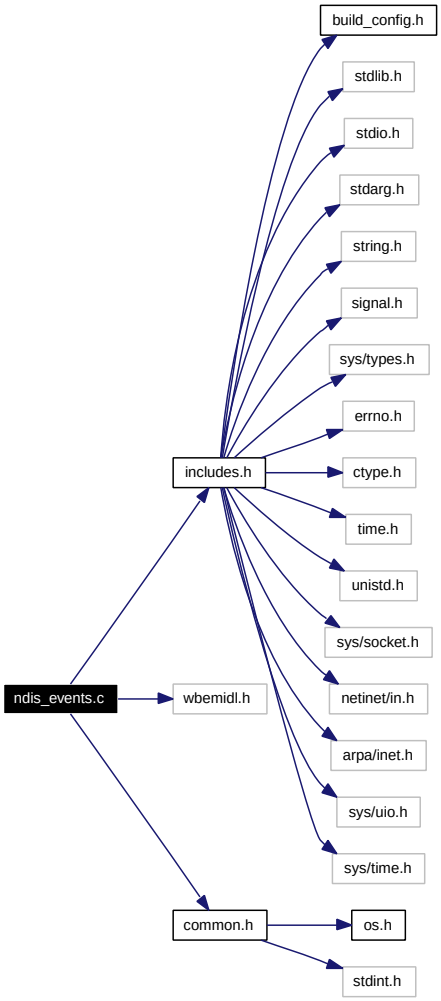
Here is the call graph for this function:

# 6.125 ndis_events.c File Reference

ndis_events - Receive NdisMIndicateStatus() events using WMI

```
#include "includes.h"
#include <wbemidl.h>
#include "common.h"
```

Include dependency graph for ndis_events.c:




## Defines

- #define **_WIN32_WINNT** 0x0400
- #define **MAX_QUERY_LEN** 256

## Enumerations

- enum **event_types** {

**EVENT_CONNECT**, **EVENT_DISCONNECT**, **EVENT_MEDIA_SPECIFIC**, **EVENT_-ADAPTER_ARRIVAL**,

**EVENT_ADAPTER_REMOVAL** }

## Functions

- void **ndis_events_deinit** (struct ndis_events_data *events)
- ndis_events_data * **ndis_events_init** (HANDLE *read_pipe, HANDLE *event_avail, const char *ifname, const char *desc)

### 6.125.1 Detailed Description

ndis_events - Receive NdisMIndicateStatus() events using WMI

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

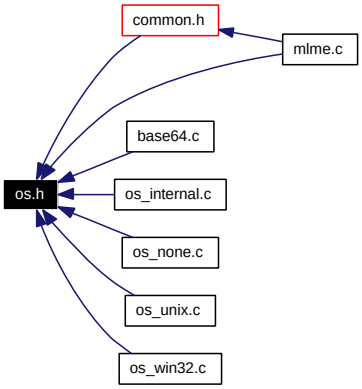
See README and COPYING for more details.

Definition in file ndis_events.c.

# 6.126 os.h File Reference

wpa_supplicant/hostapd / OS specific functions

This graph shows which files directly or indirectly include this file:




## Defines

- #define **os_time_before**(a, b)
- #define **os_time_sub**(a, b, res)
- #define **os_malloc**(s) malloc((s))
- #define **os_realloc**(p, s) realloc((p), (s))
- #define **os_free**(p) free((p))
- #define **os_memcpy**(d, s, n) memcpy((d), (s), (n))
- #define **os_memmove**(d, s, n) memmove((d), (s), (n))
- #define **os_memset**(s, c, n) memset(s, c, n)
- #define **os_memcmp**(s1, s2, n) memcmp((s1), (s2), (n))
- #define **os_strdup**(s) strdup(s)
- #define **os_strlen**(s) strlen(s)
- #define **os_strcasecmp**(s1, s2) strcasecmp((s1), (s2))
- #define **os_strncasecmp**(s1, s2, n) strncasecmp((s1), (s2), (n))
- #define **os_strchr**(s, c) strchr((s), (c))
- #define **os_strcmp**(s1, s2) strcmp((s1), (s2))
- #define **os_strncmp**(s1, s2, n) strncmp((s1), (s2), (n))
- #define **os_strncpy**(d, s, n) strncpy((d), (s), (n))
- #define **os_strrchr**(s, c) strrchr((s), (c))
- #define **os_strstr**(h, n) strstr((h), (n))
- #define **os_snprintf** snprintf

## Typedefs

- typedef long **os_time_t**

## Functions

- void os_sleep (os_time_t sec, os_time_t usec)

  *Sleep (sec, usec).*

- int os_get_time (struct os_time *t)

  *Get current time (sec, usec).*

- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)

  *Convert broken-down time into seconds since 1970-01-01.*

- int os_daemonize (const char *pid_file)

  *Run in the background (detach from the controlling terminal).*

- void os_daemonize_terminate (const char *pid_file)

  *Stop running in the background (remove pid file).*

- int os_get_random (unsigned char *buf, size_t len)

  *Get cryptographically strong pseudo random data.*

- unsigned long os_random (void)

  *Get pseudo random value (not necessarily very strong).*

- char * os_rel2abs_path (const char *rel_path)

  *Get an absolute path for a file.*

- int os_program_init (void)

  *Program initialization (called at start).*

- void os_program_deinit (void)

  *Program deinitialization (called just before exit).*

- int os_setenv (const char *name, const char *value, int overwrite)

  *Set environment variable.*

- int os_unsetenv (const char *name)

  *Delete environent variable.*

- char * os_readfile (const char *name, size_t *len)

  *Read a file to an allocated memory buffer.*

- void * os_zalloc (size_t size)

  *Allocate and zero memory.*

### 6.126.1 Detailed Description

wpa_supplicant/hostapd / OS specific functions

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file os.h.

## 6.126.2 Define Documentation

### 6.126.2.1 #define os_time_before(a, b)

**Value:**

```
((a)->sec < (b)->sec || \
         ((a)->sec == (b)->sec && (a)->usec < (b)->usec))
```

Definition at line 45 of file os.h.

### 6.126.2.2 #define os_time_sub(a, b, res)

**Value:**

```
do { \
        (res)->sec = (a)->sec - (b)->sec; \
        (res)->usec = (a)->usec - (b)->usec; \
        if ((res)->usec < 0) { \
                (res)->sec--; \
                (res)->usec += 1000000; \
        } \
} while (0)
```

Definition at line 49 of file os.h.

## 6.126.3 Function Documentation

### 6.126.3.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

**Returns:**
0 on success, -1 on failure

Definition at line 74 of file os_internal.c.

#### 6.126.3.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

Definition at line 93 of file os_internal.c.

#### 6.126.3.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

**Parameters:**
***buf*** Buffer for pseudo random data
***len*** Length of the buffer

**Returns:**
0 on success, -1 on failure

Definition at line 100 of file os_internal.c.

#### 6.126.3.4 int os_get_time (struct os_time * *t*)

Get current time (sec, usec).

**Parameters:**
***t*** Pointer to buffer for the time

**Returns:**
0 on success, -1 on failure

Definition at line 40 of file os_internal.c.

#### 6.126.3.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * *t*)

Convert broken-down time into seconds since 1970-01-01.

**Parameters:**
***year*** Four digit year
***month*** Month (1 .. 12)
***day*** Day of month (1 .. 31)
***hour*** Hour (0 .. 23)
***min*** Minute (0 .. 59)
***sec*** Second (0 .. 60)
***t*** Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

**Returns:**
0 on success, -1 on failure

Definition at line 51 of file os_internal.c.

#### 6.126.3.6 void os_program_deinit (void)

Program deinitialization (called just before exit).

This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

Definition at line 169 of file os_internal.c.

#### 6.126.3.7 int os_program_init (void)

Program initialization (called at start).

**Returns:**
0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

Definition at line 163 of file os_internal.c.

#### 6.126.3.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

**Returns:**
Pseudo random value

Definition at line 118 of file os_internal.c.

#### 6.126.3.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

**Parameters:**
***name*** Name of the file to read

***len*** For returning the length of the allocated buffer

**Returns:**
Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

Definition at line 191 of file os_internal.c.

#### 6.126.3.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

**Parameters:**
***rel_path*** Relative path to a file

**Returns:**
Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

Definition at line 124 of file os_internal.c.

#### 6.126.3.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

**Parameters:**
***name*** Name of the variable
***value*** Value to set to the variable
***overwrite*** Whether existing variable should be overwritten

**Returns:**
0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 174 of file os_internal.c.

#### 6.126.3.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

**Parameters:**
***sec*** Number of seconds to sleep
***usec*** Number of microseconds to sleep

Definition at line 31 of file os_internal.c.

#### 6.126.3.13 int os_unsetenv (const char * *name*)

Delete environent variable.

**Parameters:**
***name*** Name of the variable

**Returns:**
0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 180 of file os_internal.c.

#### 6.126.3.14 void* os_zalloc (size_t *size*)

Allocate and zero memory.

**Parameters:**
*size* Number of bytes to allocate

**Returns:**
Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

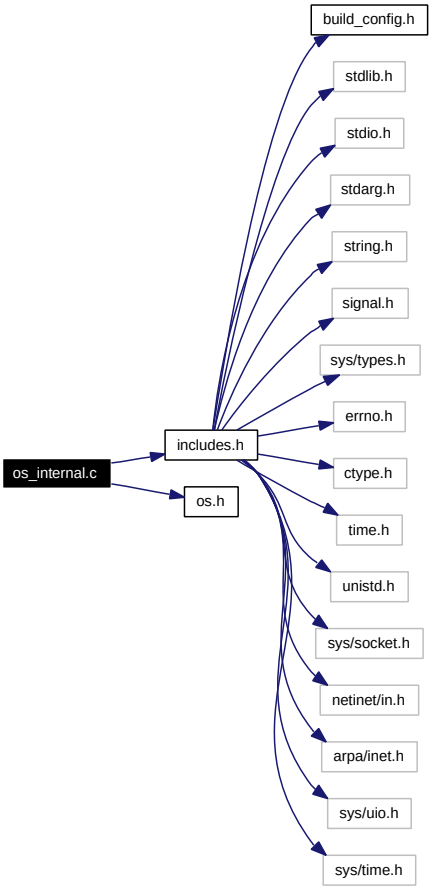
Definition at line 217 of file os_internal.c.

## 6.127 os_internal.c File Reference

wpa_supplicant/hostapd / Internal implementation of OS specific functions

```
#include "includes.h"
#include "os.h"
```

Include dependency graph for os_internal.c:




### Functions

- void os_sleep (os_time_t sec, os_time_t usec)

  *Sleep (sec, usec).*

- int os_get_time (struct os_time *t)

  *Get current time (sec, usec).*

- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)

  *Convert broken-down time into seconds since 1970-01-01.*

- int os_daemonize (const char *pid_file)

  *Run in the background (detach from the controlling terminal).*

- void os_daemonize_terminate (const char *pid_file)

  *Stop running in the background (remove pid file).*

- int os_get_random (unsigned char *buf, size_t len)

  *Get cryptographically strong pseudo random data.*

- unsigned long os_random (void)

  *Get pseudo random value (not necessarily very strong).*

- char * os_rel2abs_path (const char *rel_path)

  *Get an absolute path for a file.*

- int os_program_init (void)

  *Program initialization (called at start).*

- void os_program_deinit (void)

  *Program deinitialization (called just before exit).*

- int os_setenv (const char *name, const char *value, int overwrite)

  *Set environment variable.*

- int os_unsetenv (const char *name)

  *Delete environent variable.*

- char * os_readfile (const char *name, size_t *len)

  *Read a file to an allocated memory buffer.*

- void * os_zalloc (size_t size)

  *Allocate and zero memory.*

- void * **os_malloc** (size_t size)
- void * **os_realloc** (void *ptr, size_t size)
- void **os_free** (void *ptr)
- void * **os_memcpy** (void *dest, const void *src, size_t n)
- void * **os_memmove** (void *dest, const void *src, size_t n)
- void * **os_memset** (void *s, int c, size_t n)
- int **os_memcmp** (const void *s1, const void *s2, size_t n)
- char * **os_strdup** (const char *s)
- size_t **os_strlen** (const char *s)
- int **os_strcasecmp** (const char *s1, const char *s2)
- int **os_strncasecmp** (const char *s1, const char *s2, size_t n)
- char * **os_strchr** (const char *s, int c)
- char * **os_strrchr** (const char *s, int c)
- int **os_strcmp** (const char *s1, const char *s2)
- int **os_strncmp** (const char *s1, const char *s2, size_t n)
- char * **os_strncpy** (char *dest, const char *src, size_t n)
- char * **os_strstr** (const char *haystack, const char *needle)
- int **os_snprintf** (char *str, size_t size, const char *format,...)

### 6.127.1 Detailed Description

wpa_supplicant/hostapd / Internal implementation of OS specific functions

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file is an example of operating system specific wrapper functions. This version implements many of the functions internally, so it can be used to fill in missing functions from the target system C libraries.

Some of the functions are using standard C library calls in order to keep this file in working condition to allow the functions to be tested on a Linux target. Please note that OS_NO_C_LIB_DEFINES needs to be defined for this file to work correctly. Note that these implementations are only examples and are not optimized for speed.

Definition in file os_internal.c.

### 6.127.2 Function Documentation

#### 6.127.2.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

**Returns:**
0 on success, -1 on failure

Definition at line 74 of file os_internal.c.

#### 6.127.2.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

Definition at line 93 of file os_internal.c.

#### 6.127.2.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

**Parameters:**
***buf*** Buffer for pseudo random data

***len*** Length of the buffer

**Returns:**
0 on success, -1 on failure

Definition at line 100 of file os_internal.c.

#### 6.127.2.4 int os_get_time (struct os_time * *t*)

Get current time (sec, usec).

**Parameters:**
***t*** Pointer to buffer for the time

**Returns:**
0 on success, -1 on failure

Definition at line 40 of file os_internal.c.

#### 6.127.2.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * *t*)

Convert broken-down time into seconds since 1970-01-01.

**Parameters:**
***year*** Four digit year

***month*** Month (1 .. 12)

***day*** Day of month (1 .. 31)

***hour*** Hour (0 .. 23)

***min*** Minute (0 .. 59)

***sec*** Second (0 .. 60)

***t*** Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

**Returns:**
0 on success, -1 on failure

Definition at line 51 of file os_internal.c.

#### 6.127.2.6 void os_program_deinit (void)

Program deinitialization (called just before exit).

This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

Definition at line 169 of file os_internal.c.

#### 6.127.2.7 int os_program_init (void)

Program initialization (called at start).

**Returns:**
0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

Definition at line 163 of file os_internal.c.

#### 6.127.2.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

**Returns:**
Pseudo random value

Definition at line 118 of file os_internal.c.

#### 6.127.2.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

**Parameters:**
***name*** Name of the file to read
***len*** For returning the length of the allocated buffer

**Returns:**
Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

Definition at line 191 of file os_internal.c.

#### 6.127.2.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

**Parameters:**
***rel_path*** Relative path to a file

**Returns:**
Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

Definition at line 124 of file os_internal.c.

#### 6.127.2.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

**Parameters:**
***name*** Name of the variable
***value*** Value to set to the variable
***overwrite*** Whether existing variable should be overwritten

**Returns:**
0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 174 of file os_internal.c.

#### 6.127.2.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

**Parameters:**
***sec*** Number of seconds to sleep
***usec*** Number of microseconds to sleep

Definition at line 31 of file os_internal.c.

#### 6.127.2.13 int os_unsetenv (const char * *name*)

Delete environent variable.

**Parameters:**
***name*** Name of the variable

**Returns:**
0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 180 of file os_internal.c.

#### 6.127.2.14 void* os_zalloc (size_t *size*)

Allocate and zero memory.

**Parameters:**
***size*** Number of bytes to allocate

**Returns:**
Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

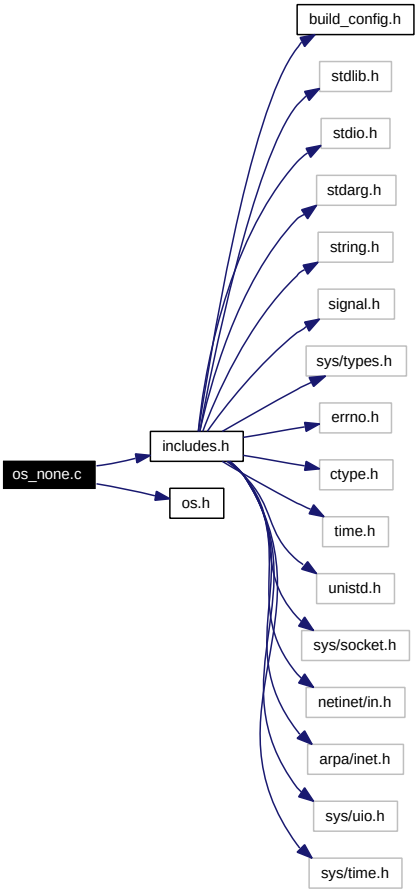
Definition at line 217 of file os_internal.c.

## 6.128 os_none.c File Reference

wpa_supplicant/hostapd / Empty OS specific functions

```
#include "includes.h"
#include "os.h"
```

Include dependency graph for os_none.c:




## Functions

- void os_sleep (os_time_t sec, os_time_t usec)

  *Sleep (sec, usec).*

- int os_get_time (struct os_time *t)

  *Get current time (sec, usec).*

- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)

  *Convert broken-down time into seconds since 1970-01-01.*

- int os_daemonize (const char *pid_file)

  *Run in the background (detach from the controlling terminal).*

- void os_daemonize_terminate (const char *pid_file)

  *Stop running in the background (remove pid file).*

- int os_get_random (unsigned char *buf, size_t len)

  *Get cryptographically strong pseudo random data.*

- unsigned long os_random (void)

  *Get pseudo random value (not necessarily very strong).*

- char * os_rel2abs_path (const char *rel_path)

  *Get an absolute path for a file.*

- int os_program_init (void)

  *Program initialization (called at start).*

- void os_program_deinit (void)

  *Program deinitialization (called just before exit).*

- int os_setenv (const char *name, const char *value, int overwrite)

  *Set environment variable.*

- int os_unsetenv (const char *name)

  *Delete environent variable.*

- char * os_readfile (const char *name, size_t *len)

  *Read a file to an allocated memory buffer.*

- void * os_zalloc (size_t size)

  *Allocate and zero memory.*

## 6.128.1 Detailed Description

wpa_supplicant/hostapd / Empty OS specific functions

**Copyright**

Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file can be used as a starting point when adding a new OS target. The functions here do not really work as-is since they are just empty or only return an error value. os_internal.c can be used as another starting point or reference since it has example implementation of many of these functions.

Definition in file os_none.c.

## 6.128.2 Function Documentation

### 6.128.2.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

**Returns:**
0 on success, -1 on failure

Definition at line 43 of file os_none.c.

### 6.128.2.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

Definition at line 49 of file os_none.c.

### 6.128.2.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

**Parameters:**
***buf*** Buffer for pseudo random data

***len*** Length of the buffer

**Returns:**
0 on success, -1 on failure

Definition at line 54 of file os_none.c.

### 6.128.2.4 int os_get_time (struct os_time * *t*)

Get current time (sec, usec).

**Parameters:**
***t*** Pointer to buffer for the time

**Returns:**
0 on success, -1 on failure

Definition at line 30 of file os_none.c.

#### 6.128.2.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * *t*)

Convert broken-down time into seconds since 1970-01-01.

**Parameters:**

***year*** Four digit year

***month*** Month (1 .. 12)

***day*** Day of month (1 .. 31)

***hour*** Hour (0 .. 23)

***min*** Minute (0 .. 59)

***sec*** Second (0 .. 60)

***t*** Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

**Returns:**

0 on success, -1 on failure

Definition at line 36 of file os_none.c.

#### 6.128.2.6 void os_program_deinit (void)

Program deinitialization (called just before exit).

This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

Definition at line 78 of file os_none.c.

#### 6.128.2.7 int os_program_init (void)

Program initialization (called at start).

**Returns:**

0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

Definition at line 72 of file os_none.c.

#### 6.128.2.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

**Returns:**

Pseudo random value

Definition at line 60 of file os_none.c.

#### 6.128.2.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

**Parameters:**

***name*** Name of the file to read

***len*** For returning the length of the allocated buffer

**Returns:**

Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

Definition at line 95 of file os_none.c.

#### 6.128.2.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

**Parameters:**

***rel_path*** Relative path to a file

**Returns:**

Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

Definition at line 66 of file os_none.c.

#### 6.128.2.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

**Parameters:**

***name*** Name of the variable

***value*** Value to set to the variable

***overwrite*** Whether existing variable should be overwritten

**Returns:**

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 83 of file os_none.c.

#### 6.128.2.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

**Parameters:**
***sec*** Number of seconds to sleep
***usec*** Number of microseconds to sleep

Definition at line 25 of file os_none.c.

#### 6.128.2.13 int os_unsetenv (const char * *name*)

Delete environent variable.

**Parameters:**
***name*** Name of the variable

**Returns:**
0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 89 of file os_none.c.

#### 6.128.2.14 void* os_zalloc (size_t *size*)

Allocate and zero memory.

**Parameters:**
***size*** Number of bytes to allocate

**Returns:**
Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

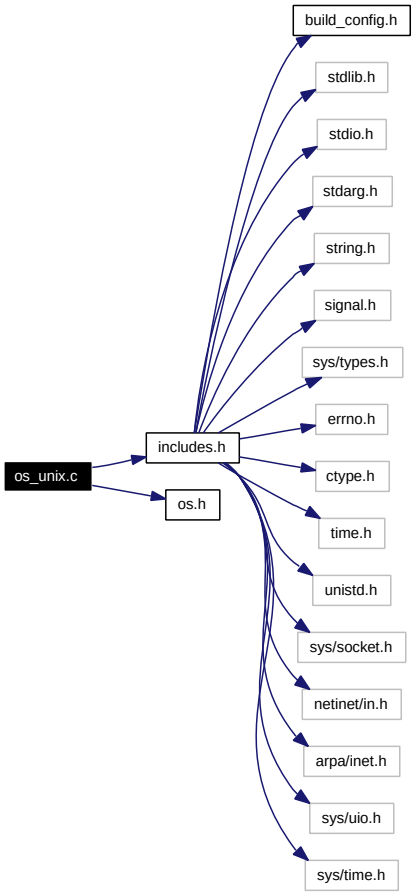
Definition at line 101 of file os_none.c.

## 6.129 os_unix.c File Reference

wpa_supplicant/hostapd / OS specific functions for UNIX/POSIX systems

```
#include "includes.h"
#include "os.h"
```

Include dependency graph for os_unix.c:




## Functions

- void os_sleep (os_time_t sec, os_time_t usec)

  *Sleep (sec, usec).*

- int os_get_time (struct os_time *t)

  *Get current time (sec, usec).*

- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)

  *Convert broken-down time into seconds since 1970-01-01.*

- int os_daemonize (const char *pid_file)

  *Run in the background (detach from the controlling terminal).*

- void os_daemonize_terminate (const char *pid_file)

  *Stop running in the background (remove pid file).*

- int os_get_random (unsigned char *buf, size_t len)

  *Get cryptographically strong pseudo random data.*

- unsigned long os_random (void)

  *Get pseudo random value (not necessarily very strong).*

- char * os_rel2abs_path (const char *rel_path)

  *Get an absolute path for a file.*

- int os_program_init (void)

  *Program initialization (called at start).*

- void os_program_deinit (void)

  *Program deinitialization (called just before exit).*

- int os_setenv (const char *name, const char *value, int overwrite)

  *Set environment variable.*

- int os_unsetenv (const char *name)

  *Delete environent variable.*

- char * os_readfile (const char *name, size_t *len)

  *Read a file to an allocated memory buffer.*

- void * os_zalloc (size_t size)

  *Allocate and zero memory.*

## 6.129.1 Detailed Description

wpa_supplicant/hostapd / OS specific functions for UNIX/POSIX systems

**Copyright**

Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file os_unix.c.

## 6.129.2 Function Documentation

### 6.129.2.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

**Returns:**
0 on success, -1 on failure

Definition at line 63 of file os_unix.c.

### 6.129.2.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

Definition at line 82 of file os_unix.c.

### 6.129.2.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

**Parameters:**
***buf*** Buffer for pseudo random data

***len*** Length of the buffer

**Returns:**
0 on success, -1 on failure

Definition at line 89 of file os_unix.c.

### 6.129.2.4 int os_get_time (struct os_time * *t*)

Get current time (sec, usec).

**Parameters:**
***t*** Pointer to buffer for the time

**Returns:**
0 on success, -1 on failure

Definition at line 29 of file os_unix.c.

#### 6.129.2.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * *t*)

Convert broken-down time into seconds since 1970-01-01.

**Parameters:**

***year*** Four digit year

***month*** Month (1 .. 12)

***day*** Day of month (1 .. 31)

***hour*** Hour (0 .. 23)

***min*** Minute (0 .. 59)

***sec*** Second (0 .. 60)

***t*** Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

**Returns:**

0 on success, -1 on failure

Definition at line 40 of file os_unix.c.

#### 6.129.2.6 void os_program_deinit (void)

Program deinitialization (called just before exit).

This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

Definition at line 162 of file os_unix.c.

#### 6.129.2.7 int os_program_init (void)

Program initialization (called at start).

**Returns:**

0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

Definition at line 156 of file os_unix.c.

#### 6.129.2.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

**Returns:**

Pseudo random value

Definition at line 107 of file os_unix.c.

#### 6.129.2.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

**Parameters:**

***name*** Name of the file to read

***len*** For returning the length of the allocated buffer

**Returns:**

Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

Definition at line 184 of file os_unix.c.

#### 6.129.2.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

**Parameters:**

***rel_path*** Relative path to a file

**Returns:**

Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

Definition at line 113 of file os_unix.c.

#### 6.129.2.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

**Parameters:**

***name*** Name of the variable

***value*** Value to set to the variable

***overwrite*** Whether existing variable should be overwritten

**Returns:**

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 167 of file os_unix.c.

#### 6.129.2.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

**Parameters:**
***sec*** Number of seconds to sleep
***usec*** Number of microseconds to sleep

Definition at line 20 of file os_unix.c.

#### 6.129.2.13 int os_unsetenv (const char * *name*)

Delete environent variable.

**Parameters:**
***name*** Name of the variable

**Returns:**
0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 173 of file os_unix.c.

#### 6.129.2.14 void* os_zalloc (size_t *size*)

Allocate and zero memory.

**Parameters:**
***size*** Number of bytes to allocate

**Returns:**
Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

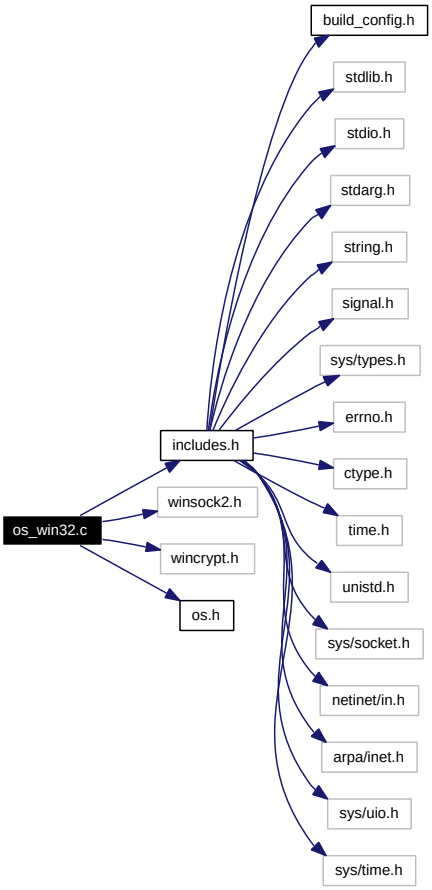
Definition at line 210 of file os_unix.c.

## 6.130 os_win32.c File Reference

wpa_supplicant/hostapd / OS specific functions for Win32 systems

```
#include "includes.h"
#include <winsock2.h>
#include <wincrypt.h>
#include "os.h"
```

Include dependency graph for os_win32.c:




### Defines

- #define **EPOCHFILETIME** (116444736000000000ULL)

### Functions

- void os_sleep (os_time_t sec, os_time_t usec)

  *Sleep (sec, usec).*

- int os_get_time (struct os_time *t)

*Get current time (sec, usec).*

- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)
  *Convert broken-down time into seconds since 1970-01-01.*
- int os_daemonize (const char *pid_file)
  *Run in the background (detach from the controlling terminal).*
- void os_daemonize_terminate (const char *pid_file)
  *Stop running in the background (remove pid file).*
- int os_get_random (unsigned char *buf, size_t len)
  *Get cryptographically strong pseudo random data.*
- unsigned long os_random (void)
  *Get pseudo random value (not necessarily very strong).*
- char * os_rel2abs_path (const char *rel_path)
  *Get an absolute path for a file.*
- int os_program_init (void)
  *Program initialization (called at start).*
- void os_program_deinit (void)
  *Program deinitialization (called just before exit).*
- int os_setenv (const char *name, const char *value, int overwrite)
  *Set environment variable.*
- int os_unsetenv (const char *name)
  *Delete environent variable.*
- char * os_readfile (const char *name, size_t *len)
  *Read a file to an allocated memory buffer.*
- void * os_zalloc (size_t size)
  *Allocate and zero memory.*

### 6.130.1 Detailed Description

wpa_supplicant/hostapd / OS specific functions for Win32 systems

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file os_win32.c.

## 6.130.2 Function Documentation

### 6.130.2.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

**Returns:**
0 on success, -1 on failure

Definition at line 79 of file os_win32.c.

### 6.130.2.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

**Parameters:**
***pid_file*** File name to write the process ID to or NULL to skip this

Definition at line 86 of file os_win32.c.

### 6.130.2.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

**Parameters:**
***buf*** Buffer for pseudo random data

***len*** Length of the buffer

**Returns:**
0 on success, -1 on failure

Definition at line 91 of file os_win32.c.

### 6.130.2.4 int os_get_time (struct os_time * *t*)

Get current time (sec, usec).

**Parameters:**
***t*** Pointer to buffer for the time

**Returns:**
0 on success, -1 on failure

Definition at line 31 of file os_win32.c.

#### 6.130.2.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * *t*)

Convert broken-down time into seconds since 1970-01-01.

**Parameters:**

***year*** Four digit year

***month*** Month (1 .. 12)

***day*** Day of month (1 .. 31)

***hour*** Hour (0 .. 23)

***min*** Minute (0 .. 59)

***sec*** Second (0 .. 60)

***t*** Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

**Returns:**

0 on success, -1 on failure

Definition at line 56 of file os_win32.c.

#### 6.130.2.6 void os_program_deinit (void)

Program deinitialization (called just before exit).

This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

Definition at line 132 of file os_win32.c.

#### 6.130.2.7 int os_program_init (void)

Program initialization (called at start).

**Returns:**

0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

Definition at line 119 of file os_win32.c.

#### 6.130.2.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

**Returns:**

Pseudo random value

Definition at line 107 of file os_win32.c.

#### 6.130.2.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

**Parameters:**

***name*** Name of the file to read

***len*** For returning the length of the allocated buffer

**Returns:**

Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

Definition at line 152 of file os_win32.c.

#### 6.130.2.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

**Parameters:**

***rel_path*** Relative path to a file

**Returns:**

Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

Definition at line 113 of file os_win32.c.

#### 6.130.2.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

**Parameters:**

***name*** Name of the variable

***value*** Value to set to the variable

***overwrite*** Whether existing variable should be overwritten

**Returns:**

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 140 of file os_win32.c.

#### 6.130.2.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

**Parameters:**
***sec*** Number of seconds to sleep
***usec*** Number of microseconds to sleep

Definition at line 22 of file os_win32.c.

#### 6.130.2.13 int os_unsetenv (const char * *name*)

Delete environent variable.

**Parameters:**
***name*** Name of the variable

**Returns:**
0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

Definition at line 146 of file os_win32.c.

#### 6.130.2.14 void* os_zalloc (size_t *size*)

Allocate and zero memory.

**Parameters:**
***size*** Number of bytes to allocate

**Returns:**
Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

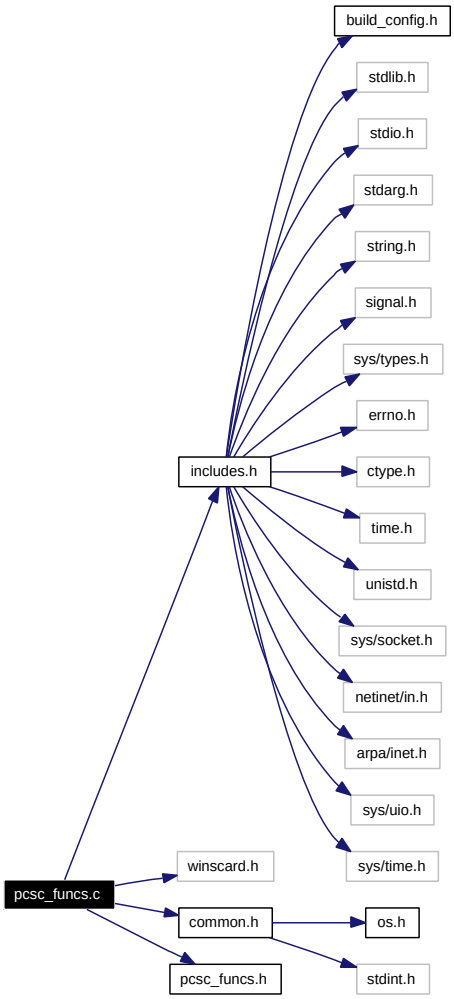
Definition at line 178 of file os_win32.c.

## 6.131 pcsc_funcs.c File Reference

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM.

```
#include "includes.h"
#include <winscard.h>
#include "common.h"
#include "pcsc_funcs.h"
```

Include dependency graph for pcsc_funcs.c:




### Defines

- #define **SIM_CMD_SELECT** 0xa0, 0xa4, 0x00, 0x00, 0x02
- #define **SIM_CMD_RUN_GSM_ALG** 0xa0, 0x88, 0x00, 0x00, 0x10
- #define **SIM_CMD_GET_RESPONSE** 0xa0, 0xc0, 0x00, 0x00
- #define **SIM_CMD_READ_BIN** 0xa0, 0xb0, 0x00, 0x00
- #define **SIM_CMD_READ_RECORD** 0xa0, 0xb2, 0x00, 0x00
- #define **SIM_CMD_VERIFY_CHV1** 0xa0, 0x20, 0x00, 0x01, 0x08

- #define **USIM_CLA** 0x00
- #define **USIM_CMD_RUN_UMTS_ALG** 0x00, 0x88, 0x00, 0x81, 0x22
- #define **USIM_CMD_GET_RESPONSE** 0x00, 0xc0, 0x00, 0x00
- #define **SIM_RECORD_MODE_ABSOLUTE** 0x04
- #define **USIM_FSP_TEMPL_TAG** 0x62
- #define **USIM_TLV_FILE_DESC** 0x82
- #define **USIM_TLV_FILE_ID** 0x83
- #define **USIM_TLV_DF_NAME** 0x84
- #define **USIM_TLV_PROPR_INFO** 0xA5
- #define **USIM_TLV_LIFE_CYCLE_STATUS** 0x8A
- #define **USIM_TLV_FILE_SIZE** 0x80
- #define **USIM_TLV_TOTAL_FILE_SIZE** 0x81
- #define **USIM_TLV_PIN_STATUS_TEMPLATE** 0xC6
- #define **USIM_TLV_SHORT_FILE_ID** 0x88
- #define **USIM_PS_DO_TAG** 0x90
- #define **AKA_RAND_LEN** 16
- #define **AKA_AUTN_LEN** 16
- #define **AKA_AUTS_LEN** 14
- #define **RES_MAX_LEN** 16
- #define **IK_LEN** 16
- #define **CK_LEN** 16
- #define **mingw_load_symbols**() 0
- #define **mingw_unload_symbols**() do { } while (0)

## Enumerations

- enum **sim_types** { **SCARD_GSM_SIM**, **SCARD_USIM** }

## Functions

- scard_data * scard_init (scard_sim_type sim_type)

  *Initialize SIM/USIM connection using PC/SC.*

- int scard_set_pin (struct scard_data *scard, const char *pin)

  *Set PIN (CHV1/PIN1) code for accessing SIM/USIM commands.*

- void scard_deinit (struct scard_data *scard)

  *Deinitialize SIM/USIM connection.*

- int scard_get_imsi (struct scard_data *scard, char *imsi, size_t *len)

  *Read IMSI from SIM/USIM card.*

- int scard_gsm_auth (struct scard_data *scard, const unsigned char *_rand, unsigned char *sres, unsigned char *kc)

  *Run GSM authentication command on SIM card.*

- int scard_umts_auth (struct scard_data *scard, const unsigned char *_rand, const unsigned char *autn, unsigned char *res, size_t *res_len, unsigned char *ik, unsigned char *ck, unsigned char *auts)

  *Run UMTS authentication command on USIM card.*

### 6.131.1 Detailed Description

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements wrapper functions for accessing GSM SIM and 3GPP USIM cards through PC/SC smartcard library. These functions are used to implement authentication routines for EAP-SIM and EAP-AKA.

Definition in file pcsc_funcs.c.

### 6.131.2 Function Documentation

#### 6.131.2.1 void scard_deinit (struct scard_data * *scard*)

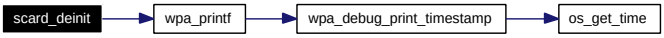
Deinitialize SIM/USIM connection.

**Parameters:**
***scard*** Pointer to private data from scard_init()

This function closes the SIM/USIM connect opened with scard_init().

Definition at line 650 of file pcsc_funcs.c.

Here is the call graph for this function:




#### 6.131.2.2 int scard_get_imsi (struct scard_data * *scard*, char * *imsi*, size_t * *len*)

Read IMSI from SIM/USIM card.

**Parameters:**
***scard*** Pointer to private data from scard_init()

***imsi*** Buffer for IMSI

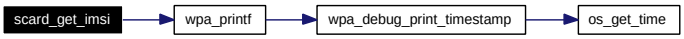
***len*** Length of imsi buffer; set to IMSI length on success

**Returns:**
0 on success, -1 if IMSI file cannot be selected, -2 if IMSI file selection returns invalid result code, -3 if parsing FSP template file fails (USIM only), -4 if IMSI does not fit in the provided imsi buffer (len is set to needed length), -5 if reading IMSI file fails.

This function can be used to read IMSI from the SIM/USIM card. If the IMSI file is PIN protected, scard_-set_pin() must have been used to set the correct PIN code before calling scard_get_imsi().

Definition at line 965 of file pcsc_funcs.c.

Here is the call graph for this function:




#### 6.131.2.3 int scard_gsm_auth (struct scard_data * *scard*, const unsigned char * *_rand*, unsigned char * *sres*, unsigned char * *kc*)

Run GSM authentication command on SIM card.

**Parameters:**

***scard*** Pointer to private data from scard_init()

***_rand*** 16-byte RAND value from HLR/AuC

***sres*** 4-byte buffer for SRES

***kc*** 8-byte buffer for Kc

**Returns:**

0 on success, -1 if SIM/USIM connection has not been initialized, -2 if authentication command execution fails, -3 if unknown response code for authentication command is received, -4 if reading of response fails, -5 if if response data is of unexpected length

This function performs GSM authentication using SIM/USIM card and the provided RAND value from HLR/AuC. If authentication command can be completed successfully, SRES and Kc values will be written into sres and kc buffers.

Definition at line 1045 of file pcsc_funcs.c.

Here is the call graph for this function:


#### 6.131.2.4 struct scard_data* scard_init (scard_sim_type *sim_type*)

Initialize SIM/USIM connection using PC/SC.

**Parameters:**

***sim_type*** Allowed SIM types (SIM, USIM, or both)

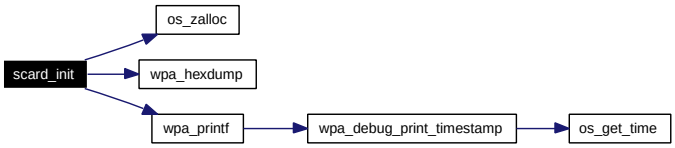
**Returns:**

Pointer to private data structure, or NULL on failure

This function is used to initialize SIM/USIM connection. PC/SC is used to open connection to the SIM/USIM card and the card is verified to support the selected sim_type. In addition, local flag is set if a PIN is needed to access some of the card functions. Once the connection is not needed anymore, scard_deinit() can be used to close it.

Definition at line 436 of file pcsc_funcs.c.

Here is the call graph for this function:




#### 6.131.2.5 int scard_set_pin (struct scard_data * *scard*, const char * *pin*)

Set PIN (CHV1/PIN1) code for accessing SIM/USIM commands.

**Parameters:**

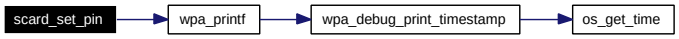
***scard*** Pointer to private data from scard_init() pin: PIN code as an ASCII string (e.g., "1234")

**Returns:**

0 on success, -1 on failure

Definition at line 620 of file pcsc_funcs.c.

Here is the call graph for this function:




#### 6.131.2.6 int scard_umts_auth (struct scard_data * *scard*, const unsigned char * *_rand*, const unsigned char * *autn*, unsigned char * *res*, size_t * *res_len*, unsigned char * *ik*, unsigned char * *ck*, unsigned char * *auts*)

Run UMTS authentication command on USIM card.

**Parameters:**

***scard*** Pointer to private data from scard_init()

***_rand*** 16-byte RAND value from HLR/AuC

***autn*** 16-byte AUTN value from HLR/AuC

***res*** 16-byte buffer for RES

***res_len*** Variable that will be set to RES length

***ik*** 16-byte buffer for IK

***ck*** 16-byte buffer for CK

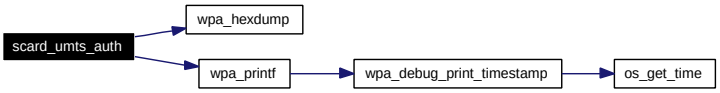
***auts*** 14-byte buffer for AUTS

**Returns:**
0 on success, -1 on failure, or -2 if USIM reports synchronization failure

This function performs AKA authentication using USIM card and the provided RAND and AUTN values from HLR/AuC. If authentication command can be completed successfully, RES, IK, and CK values will be written into provided buffers and res_len is set to length of received RES value. If USIM reports synchronization failure, the received AUTS value will be written into auts buffer. In this case, RES, IK, and CK are not valid.

Definition at line 1144 of file pcsc_funcs.c.

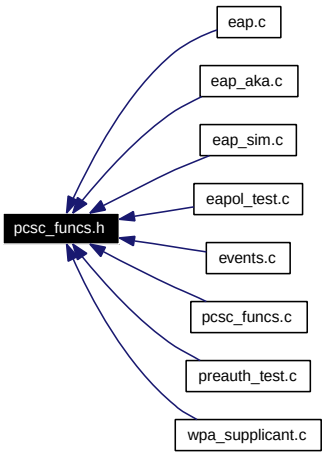
Here is the call graph for this function:



## 6.132 pcsc_funcs.h File Reference

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **SCARD_FILE_MF** 0x3F00
- #define **SCARD_FILE_GSM_DF** 0x7F20
- #define **SCARD_FILE_UMTS_DF** 0x7F50
- #define **SCARD_FILE_GSM_EF_IMSI** 0x6F07
- #define **SCARD_FILE_EF_DIR** 0x2F00
- #define **SCARD_FILE_EF_ICCID** 0x2FE2
- #define **SCARD_FILE_EF_CK** 0x6FE1
- #define **SCARD_FILE_EF_IK** 0x6FE2
- #define **SCARD_CHV1_OFFSET** 13
- #define **SCARD_CHV1_FLAG** 0x80
- #define **scard_init**(s) NULL
- #define **scard_deinit**(s) do { } while (0)
- #define **scard_set_pin**(s, p) -1
- #define **scard_get_imsi**(s, i, l) -1
- #define **scard_gsm_auth**(s, r, s2, k) -1
- #define **scard_umts_auth**(s, r, a, r2, rl, i, c, a2) -1

### Enumerations

- enum **scard_sim_type** { **SCARD_GSM_SIM_ONLY**, **SCARD_USIM_ONLY**, **SCARD_TRY_BOTH** }

### 6.132.1 Detailed Description

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM.

**Copyright**



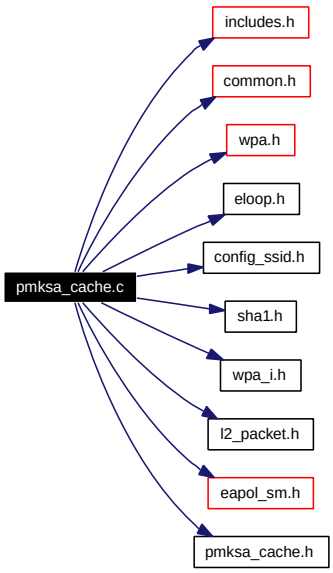
Definition in file pcsc_funcs.h.

## 6.133 pmksa_cache.c File Reference

WPA Supplicant - RSN PMKSA cache.

```
#include "includes.h"
#include "common.h"
#include "wpa.h"
#include "eloop.h"
#include "config_ssid.h"
#include "sha1.h"
#include "wpa_i.h"
#include "l2_packet.h"
#include "eapol_sm.h"
#include "pmksa_cache.h"
```

Include dependency graph for pmksa_cache.c:




### Functions

- rsn_pmksa_cache_entry * pmksa_cache_add (struct rsn_pmksa_cache *pmksa, const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, struct wpa_ssid *ssid)

  *Add a PMKSA cache entry.*

- void pmksa_cache_deinit (struct rsn_pmksa_cache *pmksa)

  *Free all entries in PMKSA cache.*

- rsn_pmksa_cache_entry * pmksa_cache_get (struct rsn_pmksa_cache *pmksa, const u8 *aa, const u8 *pmkid)

  *Fetch a PMKSA cache entry.*

- void pmksa_cache_notify_reconfig (struct rsn_pmksa_cache *pmksa)

  *Reconfiguration notification for PMKSA cache.*

- rsn_pmksa_cache_entry * pmksa_cache_get_opportunistic (struct rsn_pmksa_cache *pmksa, struct wpa_ssid *ssid, const u8 *aa)

  *Try to get an opportunistic PMKSA entry.*

- rsn_pmksa_cache_entry * pmksa_cache_get_current (struct wpa_sm *sm)

  *Get the current used PMKSA entry.*

- void pmksa_cache_clear_current (struct wpa_sm *sm)

  *Clear the current PMKSA entry selection.*

- int pmksa_cache_set_current (struct wpa_sm *sm, const u8 *pmkid, const u8 *bssid, struct wpa_ssid *ssid, int try_opportunistic)

  *Set the current PMKSA entry selection.*

- int pmksa_cache_list (struct wpa_sm *sm, char *buf, size_t len)

  *Dump text list of entries in PMKSA cache.*

- rsn_pmksa_cache * pmksa_cache_init (void(*free_cb)(struct rsn_pmksa_cache_entry *entry, void *ctx, int replace), void *ctx, struct wpa_sm *sm)

  *Initialize PMKSA cache.*

### 6.133.1 Detailed Description

WPA Supplicant - RSN PMKSA cache.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file pmksa_cache.c.

### 6.133.2 Function Documentation

#### 6.133.2.1 struct rsn_pmksa_cache_entry* pmksa_cache_add (struct rsn_pmksa_cache * *pmksa*, const u8 * *pmk*, size_t *pmk_len*, const u8 * *aa*, const u8 * *spa*, struct wpa_ssid * *ssid*)

Add a PMKSA cache entry.

**Parameters:**

***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

***pmk*** The new pairwise master key

***pmk_len*** PMK length in bytes, usually PMK_LEN (32)

***aa*** Authenticator address

***spa*** Supplicant address

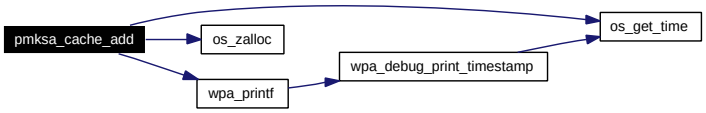
***ssid*** The network configuration for which this PMK is being added

**Returns:**
Pointer to the added PMKSA cache entry or NULL on error

This function create a PMKSA entry for a new PMK and adds it to the PMKSA cache. If an old entry is already in the cache for the same Authenticator, this entry will be replaced with the new entry. PMKID will be calculated based on the PMK and the driver interface is notified of the new PMKID.

Definition at line 161 of file pmksa_cache.c.

Here is the call graph for this function:




#### 6.133.2.2 void pmksa_cache_clear_current (struct wpa_sm * *sm*)

Clear the current PMKSA entry selection.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 401 of file pmksa_cache.c.

#### 6.133.2.3 void pmksa_cache_deinit (struct rsn_pmksa_cache * *pmksa*)

Free all entries in PMKSA cache.

**Parameters:**
***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

Definition at line 267 of file pmksa_cache.c.

#### 6.133.2.4 struct rsn_pmksa_cache_entry* pmksa_cache_get (struct rsn_pmksa_cache * *pmksa*, const u8 * *aa*, const u8 * *pmkid*)

Fetch a PMKSA cache entry.

**Parameters:**
***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

***aa*** Authenticator address or NULL to match any

***pmkid*** PMKID or NULL to match any

**Returns:**
Pointer to PMKSA cache entry or NULL if no match was found

Definition at line 294 of file pmksa_cache.c.

#### 6.133.2.5 struct rsn_pmksa_cache_entry* pmksa_cache_get_current (struct wpa_sm * *sm*)

Get the current used PMKSA entry.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

**Returns:**
Pointer to the current PMKSA cache entry or NULL if not available

Definition at line 388 of file pmksa_cache.c.

#### 6.133.2.6 struct rsn_pmksa_cache_entry* pmksa_cache_get_opportunistic (struct rsn_pmksa_cache * *pmksa*, struct wpa_ssid * *ssid*, const u8 * *aa*)

Try to get an opportunistic PMKSA entry.

**Parameters:**
***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

***ssid*** Pointer to the current network configuration

***aa*** Authenticator address for the new AP

**Returns:**
Pointer to a new PMKSA cache entry or NULL if not available

Try to create a new PMKSA cache entry opportunistically by guessing that the new AP is sharing the same PMK as another AP that has the same SSID and has already an entry in PMKSA cache.

Definition at line 359 of file pmksa_cache.c.

Here is the call graph for this function:


#### 6.133.2.7 struct rsn_pmksa_cache* pmksa_cache_init (void(*)(struct rsn_pmksa_cache_entry *entry, void *ctx, int replace) *free_cb*, void * *ctx*, struct wpa_sm * *sm*)

Initialize PMKSA cache.

**Parameters:**
***free_cb*** Callback function to be called when a PMKSA cache entry is freed

***ctx*** Context pointer for free_cb function

***sm*** Pointer to WPA state machine data from wpa_sm_init()

**Returns:**
Pointer to PMKSA cache data or NULL on failure

Definition at line 498 of file pmksa_cache.c.

Here is the call graph for this function:


#### 6.133.2.8 int pmksa_cache_list (struct wpa_sm * *sm*, char * *buf*, size_t *len*)

Dump text list of entries in PMKSA cache.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***buf*** Buffer for the list

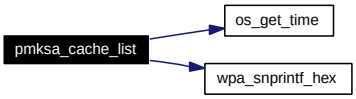
***len*** Length of the buffer

**Returns:**
number of bytes written to buffer

This function is used to generate a text format representation of the current PMKSA cache contents for the ctrl_iface PMKSA command.

Definition at line 452 of file pmksa_cache.c.

Here is the call graph for this function:




#### 6.133.2.9 void pmksa_cache_notify_reconfig (struct rsn_pmksa_cache * *pmksa*)

Reconfiguration notification for PMKSA cache.

**Parameters:**
***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

Clear references to old data structures when wpa_supplicant is reconfigured.

Definition at line 316 of file pmksa_cache.c.

#### 6.133.2.10 int pmksa_cache_set_current (struct wpa_sm * *sm*, const u8 * *pmkid*, const u8 * *bssid*, struct wpa_ssid * *ssid*, int *try_opportunistic*)

Set the current PMKSA entry selection.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***pmkid*** PMKID for selecting PMKSA or NULL if not used

***bssid*** BSSID for PMKSA or NULL if not used

***ssid*** The network configuration for the current network

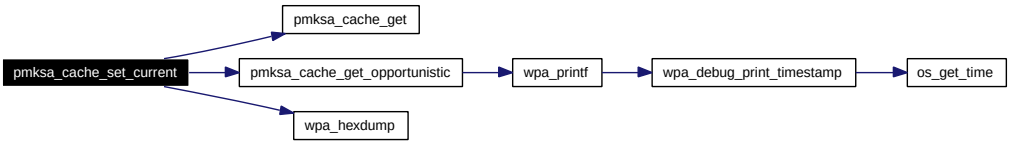
***try_opportunistic*** Whether to allow opportunistic PMKSA caching

**Returns:**

0 if PMKSA was found or -1 if no matching entry was found

Definition at line 419 of file pmksa_cache.c.

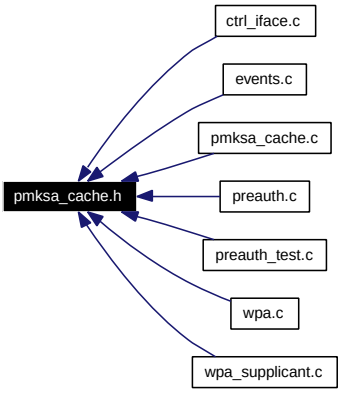
Here is the call graph for this function:



## 6.134 pmksa_cache.h File Reference

wpa_supplicant - WPA2/RSN PMKSA cache functions

This graph shows which files directly or indirectly include this file:




### Functions

- rsn_pmksa_cache * pmksa_cache_init (void(*free_cb)(struct rsn_pmksa_cache_entry *entry, void *ctx, int replace), void *ctx, struct wpa_sm *sm)

  *Initialize PMKSA cache.*

- void pmksa_cache_deinit (struct rsn_pmksa_cache *pmksa)

  *Free all entries in PMKSA cache.*

- rsn_pmksa_cache_entry * pmksa_cache_get (struct rsn_pmksa_cache *pmksa, const u8 *aa, const u8 *pmkid)

  *Fetch a PMKSA cache entry.*

- int pmksa_cache_list (struct wpa_sm *sm, char *buf, size_t len)

  *Dump text list of entries in PMKSA cache.*

- rsn_pmksa_cache_entry * pmksa_cache_add (struct rsn_pmksa_cache *pmksa, const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, struct wpa_ssid *ssid)

  *Add a PMKSA cache entry.*

- void pmksa_cache_notify_reconfig (struct rsn_pmksa_cache *pmksa)

  *Reconfiguration notification for PMKSA cache.*

- rsn_pmksa_cache_entry * pmksa_cache_get_current (struct wpa_sm *sm)

  *Get the current used PMKSA entry.*

- void pmksa_cache_clear_current (struct wpa_sm *sm)

  *Clear the current PMKSA entry selection.*

- int pmksa_cache_set_current (struct wpa_sm *sm, const u8 *pmkid, const u8 *bssid, struct wpa_ssid *ssid, int try_opportunistic)

*Set the current PMKSA entry selection.*

- rsn_pmksa_cache_entry * pmksa_cache_get_opportunistic (struct rsn_pmksa_cache *pmksa, struct wpa_ssid *ssid, const u8 *aa)

  *Try to get an opportunistic PMKSA entry.*

## 6.134.1 Detailed Description

wpa_supplicant - WPA2/RSN PMKSA cache functions

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file pmksa_cache.h.

## 6.134.2 Function Documentation

### 6.134.2.1 struct rsn_pmksa_cache_entry* pmksa_cache_add (struct rsn_pmksa_cache * *pmksa*, const u8 * *pmk*, size_t *pmk_len*, const u8 * *aa*, const u8 * *spa*, struct wpa_ssid * *ssid*)

Add a PMKSA cache entry.

**Parameters:**

***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

***pmk*** The new pairwise master key

***pmk_len*** PMK length in bytes, usually PMK_LEN (32)

***aa*** Authenticator address

***spa*** Supplicant address

***ssid*** The network configuration for which this PMK is being added

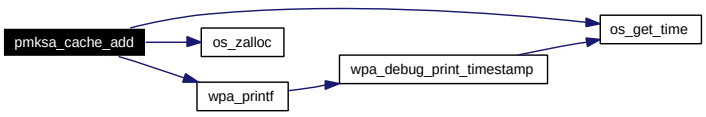
**Returns:**

Pointer to the added PMKSA cache entry or NULL on error

This function create a PMKSA entry for a new PMK and adds it to the PMKSA cache. If an old entry is already in the cache for the same Authenticator, this entry will be replaced with the new entry. PMKID will be calculated based on the PMK and the driver interface is notified of the new PMKID.

Definition at line 161 of file pmksa_cache.c.

Here is the call graph for this function:



#### 6.134.2.2 void pmksa_cache_clear_current (struct wpa_sm * *sm*)

Clear the current PMKSA entry selection.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 401 of file pmksa_cache.c.

#### 6.134.2.3 void pmksa_cache_deinit (struct rsn_pmksa_cache * *pmksa*)

Free all entries in PMKSA cache.

**Parameters:**
***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

Definition at line 267 of file pmksa_cache.c.

#### 6.134.2.4 struct rsn_pmksa_cache_entry* pmksa_cache_get (struct rsn_pmksa_cache * *pmksa*, const u8 * *aa*, const u8 * *pmkid*)

Fetch a PMKSA cache entry.

**Parameters:**
***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

***aa*** Authenticator address or NULL to match any

***pmkid*** PMKID or NULL to match any

**Returns:**
Pointer to PMKSA cache entry or NULL if no match was found

Definition at line 294 of file pmksa_cache.c.

#### 6.134.2.5 struct rsn_pmksa_cache_entry* pmksa_cache_get_current (struct wpa_sm * *sm*)

Get the current used PMKSA entry.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

**Returns:**
Pointer to the current PMKSA cache entry or NULL if not available

Definition at line 388 of file pmksa_cache.c.

#### 6.134.2.6 struct rsn_pmksa_cache_entry* pmksa_cache_get_opportunistic (struct rsn_pmksa_cache * *pmksa*, struct wpa_ssid * *ssid*, const u8 * *aa*)

Try to get an opportunistic PMKSA entry.

**Parameters:**

***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

***ssid*** Pointer to the current network configuration

***aa*** Authenticator address for the new AP

**Returns:**

Pointer to a new PMKSA cache entry or NULL if not available

Try to create a new PMKSA cache entry opportunistically by guessing that the new AP is sharing the same PMK as another AP that has the same SSID and has already an entry in PMKSA cache.

Definition at line 359 of file pmksa_cache.c.

Here is the call graph for this function:


#### 6.134.2.7 struct rsn_pmksa_cache* pmksa_cache_init (void(*)(struct rsn_pmksa_cache_entry *entry, void *ctx, int replace) *free_cb*, void * *ctx*, struct wpa_sm * *sm*)

Initialize PMKSA cache.

**Parameters:**

***free_cb*** Callback function to be called when a PMKSA cache entry is freed

***ctx*** Context pointer for free_cb function

***sm*** Pointer to WPA state machine data from wpa_sm_init()

**Returns:**

Pointer to PMKSA cache data or NULL on failure

Definition at line 498 of file pmksa_cache.c.

Here is the call graph for this function:


#### 6.134.2.8 int pmksa_cache_list (struct wpa_sm * *sm*, char * *buf*, size_t *len*)

Dump text list of entries in PMKSA cache.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***buf*** Buffer for the list

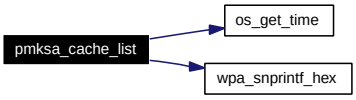
***len*** Length of the buffer

**Returns:**
number of bytes written to buffer

This function is used to generate a text format representation of the current PMKSA cache contents for the ctrl_iface PMKSA command.

Definition at line 452 of file pmksa_cache.c.

Here is the call graph for this function:




#### 6.134.2.9 void pmksa_cache_notify_reconfig (struct rsn_pmksa_cache * *pmksa*)

Reconfiguration notification for PMKSA cache.

**Parameters:**
***pmksa*** Pointer to PMKSA cache data from pmksa_cache_init()

Clear references to old data structures when wpa_supplicant is reconfigured.

Definition at line 316 of file pmksa_cache.c.

#### 6.134.2.10 int pmksa_cache_set_current (struct wpa_sm * *sm*, const u8 * *pmkid*, const u8 * *bssid*, struct wpa_ssid * *ssid*, int *try_opportunistic*)

Set the current PMKSA entry selection.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***pmkid*** PMKID for selecting PMKSA or NULL if not used

***bssid*** BSSID for PMKSA or NULL if not used

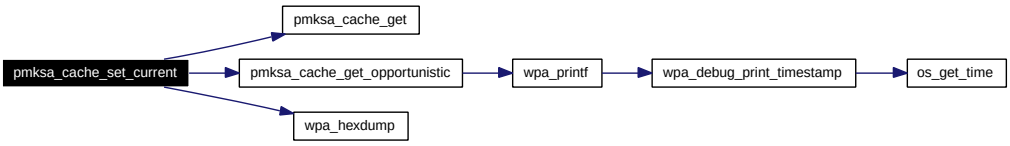
***ssid*** The network configuration for the current network

***try_opportunistic*** Whether to allow opportunistic PMKSA caching

**Returns:**
0 if PMKSA was found or -1 if no matching entry was found

Definition at line 419 of file pmksa_cache.c.

Here is the call graph for this function:

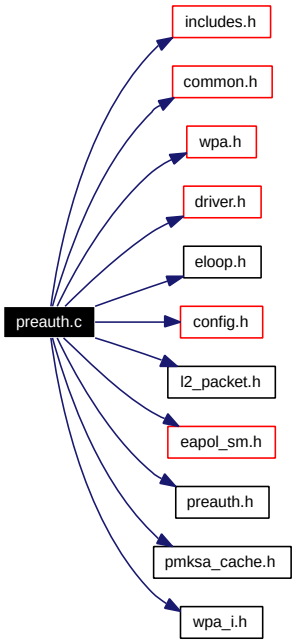


# 6.135 preauth.c File Reference

WPA Supplicant - RSN pre-authentication.

```
#include "includes.h"
#include "common.h"
#include "wpa.h"
#include "driver.h"
#include "eloop.h"
#include "config.h"
#include "l2_packet.h"
#include "eapol_sm.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_i.h"
```

Include dependency graph for preauth.c:




## Defines

- #define **PMKID_CANDIDATE_PRIO_SCAN** 1000

## Functions

- void pmksa_candidate_free (struct wpa_sm *sm)

  *Free all entries in PMKSA candidate list.*

- int rsn_preauth_init (struct wpa_sm *sm, const u8 *dst, struct wpa_ssid *config)
  *Start new RSN pre-authentication.*
- void rsn_preauth_deinit (struct wpa_sm *sm)
  *Abort RSN pre-authentication.*
- void rsn_preauth_candidate_process (struct wpa_sm *sm)
  *Process PMKSA candidates.*
- void pmksa_candidate_add (struct wpa_sm *sm, const u8 *bssid, int prio, int preauth)
  *Add a new PMKSA candidate.*
- void rsn_preauth_scan_results (struct wpa_sm *sm, struct wpa_scan_result *results, int count)
  *Process scan results to find PMKSA candidates.*
- int rsn_preauth_get_status (struct wpa_sm *sm, char *buf, size_t buflen, int verbose)
  *Get pre-authentication status.*
- int rsn_preauth_in_progress (struct wpa_sm *sm)
  *Verify whether pre-authentication is in progress.*

## 6.135.1 Detailed Description

WPA Supplicant - RSN pre-authentication.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file preauth.c.

## 6.135.2 Function Documentation

### 6.135.2.1 void pmksa_candidate_add (struct wpa_sm * *sm*, const u8 * *bssid*, int *prio*, int *preauth*)

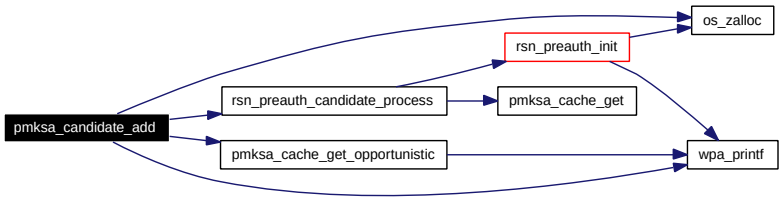
Add a new PMKSA candidate.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()
***bssid*** BSSID (authenticator address) of the candidate
***prio*** Priority (the smaller number, the higher priority)
***preauth*** Whether the candidate AP advertises support for pre-authentication

This function is used to add PMKSA candidates for RSN pre-authentication. It is called from scan result processing and from driver events for PMKSA candidates, i.e., EVENT_PMKID_CANDIDATE events to wpa_supplicant_event().

Definition at line 363 of file preauth.c.

Here is the call graph for this function:




#### 6.135.2.2 void pmksa_candidate_free (struct wpa_sm * *sm*)

Free all entries in PMKSA candidate list.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 45 of file preauth.c.

#### 6.135.2.3 void rsn_preauth_candidate_process (struct wpa_sm * *sm*)

Process PMKSA candidates.

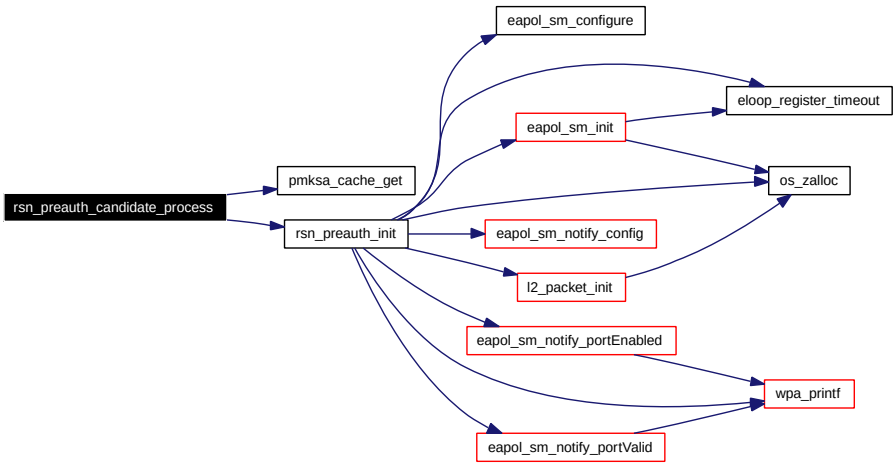
**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

Go through the PMKSA candidates and start pre-authentication if a candidate without an existing PMKSA cache entry is found. Processed candidates will be removed from the list.

Definition at line 299 of file preauth.c.

Here is the call graph for this function:




#### 6.135.2.4 void rsn_preauth_deinit (struct wpa_sm * *sm*)

Abort RSN pre-authentication.

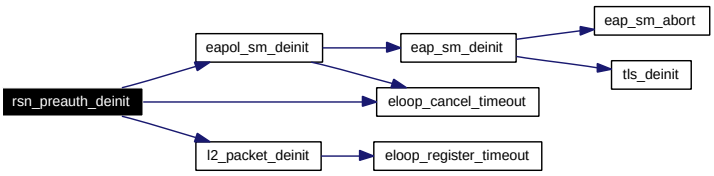
**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

This function aborts the current RSN pre-authentication (if one is started) and frees resources allocated for it.

Definition at line 271 of file preauth.c.

Here is the call graph for this function:




#### 6.135.2.5 int rsn_preauth_get_status (struct wpa_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get pre-authentication status.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

**Returns:**

Number of bytes written to buf.

Query WPA2 pre-authentication for status information. This function fills in a text area with current status information. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

Definition at line 500 of file preauth.c.

Here is the call graph for this function:


#### 6.135.2.6 int rsn_preauth_in_progress (struct wpa_sm * *sm*)

Verify whether pre-authentication is in progress.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 528 of file preauth.c.

#### 6.135.2.7 int rsn_preauth_init (struct wpa_sm * *sm*, const u8 * *dst*, struct wpa_ssid * *config*)

Start new RSN pre-authentication.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***dst*** Authenticator address (BSSID) with which to preauthenticate

***config*** Current network configuration

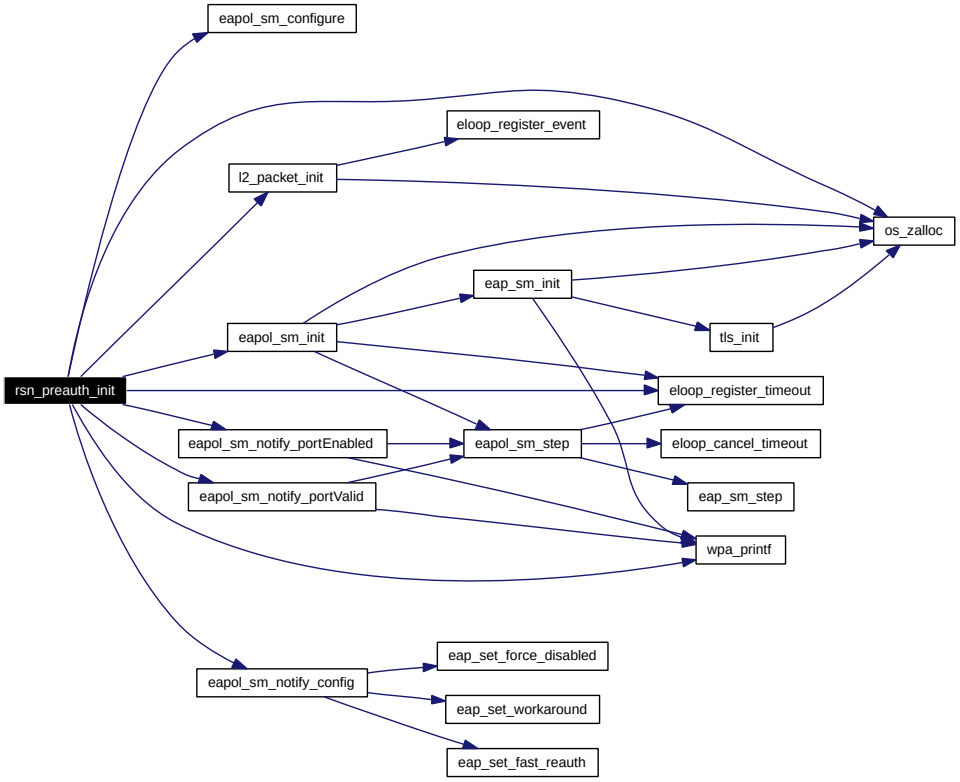
**Returns:**

0 on success, -1 on another pre-authentication is in progress, -2 on layer 2 packet initialization failure, -3 on EAPOL state machine initialization failure, -4 on memory allocation failure

This function request an RSN pre-authentication with a given destination address. This is usually called for PMKSA candidates found from scan results or from driver reports. In addition, ctrl_iface PREAUTH command can trigger pre-authentication.

Definition at line 180 of file preauth.c.

Here is the call graph for this function:




#### 6.135.2.8 void rsn_preauth_scan_results (struct wpa_sm * *sm*, struct wpa_scan_result * *results*, int *count*)

Process scan results to find PMKSA candidates.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

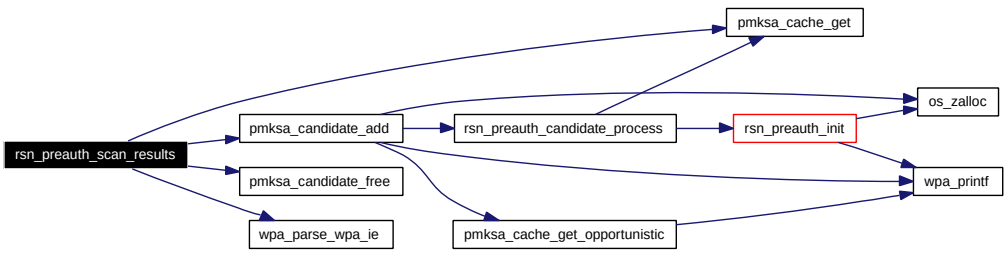
***results*** Scan results

***count*** Number of BSSes in scan results

This functions goes through the scan results and adds all suitable APs (Authenticators) into PMKSA candidate list.

Definition at line 438 of file preauth.c.

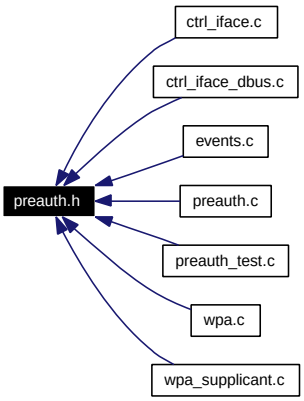
Here is the call graph for this function:



# 6.136 preauth.h File Reference

wpa_supplicant - WPA2/RSN pre-authentication functions

This graph shows which files directly or indirectly include this file:




## Functions

- void pmksa_candidate_free (struct wpa_sm *sm)
  *Free all entries in PMKSA candidate list.*
- int rsn_preauth_init (struct wpa_sm *sm, const u8 *dst, struct wpa_ssid *config)
  *Start new RSN pre-authentication.*
- void rsn_preauth_deinit (struct wpa_sm *sm)
  *Abort RSN pre-authentication.*
- void rsn_preauth_scan_results (struct wpa_sm *sm, struct wpa_scan_result *results, int count)
  *Process scan results to find PMKSA candidates.*
- void pmksa_candidate_add (struct wpa_sm *sm, const u8 *bssid, int prio, int preauth)
  *Add a new PMKSA candidate.*
- void rsn_preauth_candidate_process (struct wpa_sm *sm)
  *Process PMKSA candidates.*
- int rsn_preauth_get_status (struct wpa_sm *sm, char *buf, size_t buflen, int verbose)
  *Get pre-authentication status.*
- int rsn_preauth_in_progress (struct wpa_sm *sm)
  *Verify whether pre-authentication is in progress.*

### 6.136.1 Detailed Description

wpa_supplicant - WPA2/RSN pre-authentication functions

**Copyright**
Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file preauth.h.

## 6.136.2 Function Documentation

### 6.136.2.1 void pmksa_candidate_add (struct wpa_sm * *sm*, const u8 * *bssid*, int *prio*, int *preauth*)

Add a new PMKSA candidate.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***bssid*** BSSID (authenticator address) of the candidate

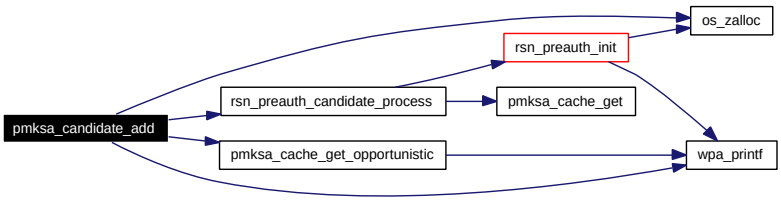
***prio*** Priority (the smaller number, the higher priority)

***preauth*** Whether the candidate AP advertises support for pre-authentication

This function is used to add PMKSA candidates for RSN pre-authentication. It is called from scan result processing and from driver events for PMKSA candidates, i.e., EVENT_PMKID_CANDIDATE events to wpa_supplicant_event().

Definition at line 363 of file preauth.c.

Here is the call graph for this function:




### 6.136.2.2 void pmksa_candidate_free (struct wpa_sm * *sm*)

Free all entries in PMKSA candidate list.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 45 of file preauth.c.

#### 6.136.2.3 void rsn_preauth_candidate_process (struct wpa_sm * *sm*)

Process PMKSA candidates.

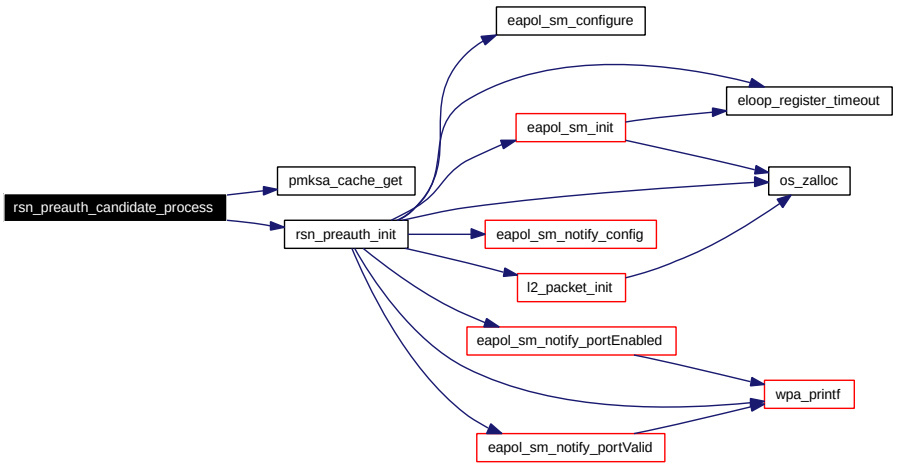
**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

Go through the PMKSA candidates and start pre-authentication if a candidate without an existing PMKSA cache entry is found. Processed candidates will be removed from the list.

Definition at line 299 of file preauth.c.

Here is the call graph for this function:




#### 6.136.2.4 void rsn_preauth_deinit (struct wpa_sm * *sm*)

Abort RSN pre-authentication.

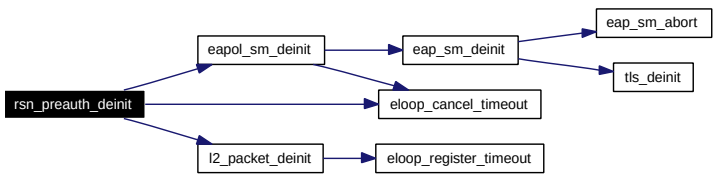
**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

This function aborts the current RSN pre-authentication (if one is started) and frees resources allocated for it.

Definition at line 271 of file preauth.c.

Here is the call graph for this function:



#### 6.136.2.5 int rsn_preauth_get_status (struct wpa_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get pre-authentication status.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

**Returns:**

Number of bytes written to buf.

Query WPA2 pre-authentication for status information. This function fills in a text area with current status information. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

Definition at line 500 of file preauth.c.

Here is the call graph for this function:


#### 6.136.2.6 int rsn_preauth_in_progress (struct wpa_sm * *sm*)

Verify whether pre-authentication is in progress.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 528 of file preauth.c.

#### 6.136.2.7 int rsn_preauth_init (struct wpa_sm * *sm*, const u8 * *dst*, struct wpa_ssid * *config*)

Start new RSN pre-authentication.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***dst*** Authenticator address (BSSID) with which to preauthenticate

***config*** Current network configuration

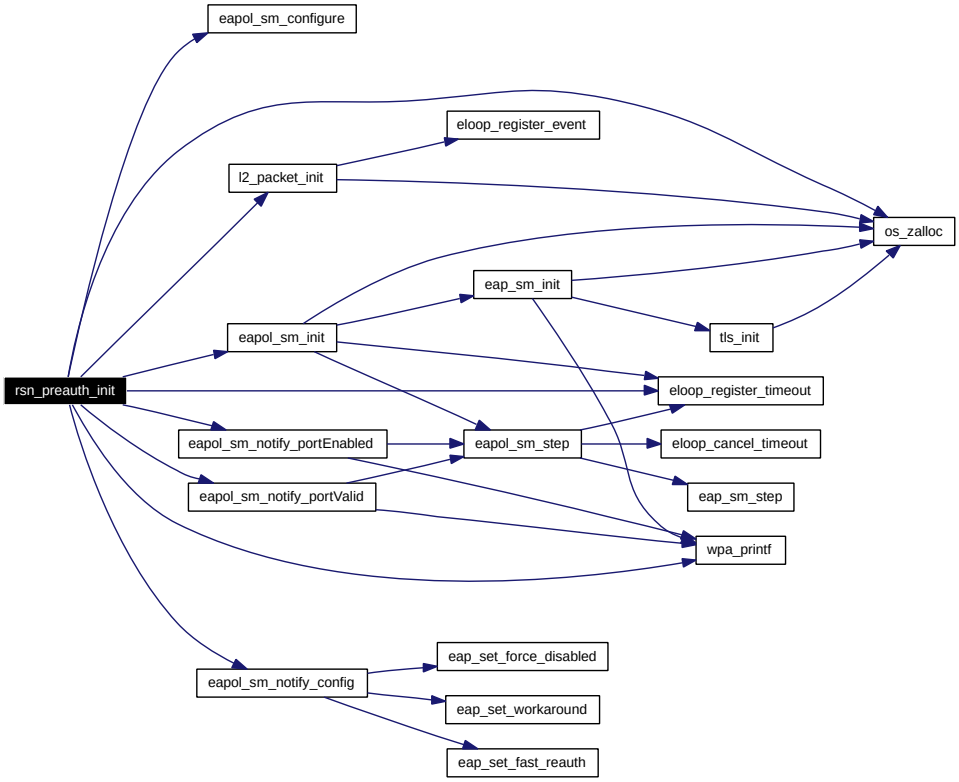
**Returns:**

0 on success, -1 on another pre-authentication is in progress, -2 on layer 2 packet initialization failure, -3 on EAPOL state machine initialization failure, -4 on memory allocation failure

This function request an RSN pre-authentication with a given destination address. This is usually called for PMKSA candidates found from scan results or from driver reports. In addition, ctrl_iface PREAUTH command can trigger pre-authentication.

Definition at line 180 of file preauth.c.

Here is the call graph for this function:




### 6.136.2.8 void rsn_preauth_scan_results (struct wpa_sm * *sm*, struct wpa_scan_result * *results*, int *count*)

Process scan results to find PMKSA candidates.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

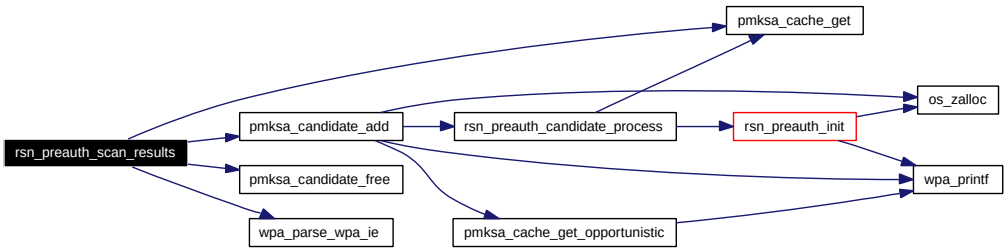
***results*** Scan results

***count*** Number of BSSes in scan results

This functions goes through the scan results and adds all suitable APs (Authenticators) into PMKSA candidate list.

Definition at line 438 of file preauth.c.

Here is the call graph for this function:

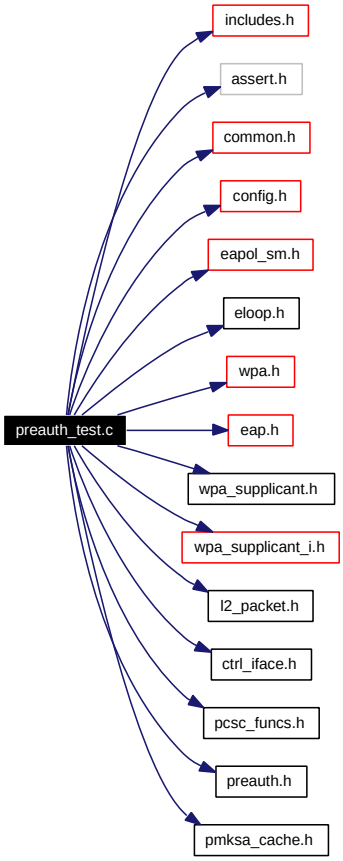

pmksa_cache_get
os_zalloc
pmksa_candidate_add
rsn_preauth_candidate_process
rsn_preauth_init
rsn_preauth_scan_results
wpa_printf
pmksa_candidate_free
wpa_parse_wpa_ie
pmksa_cache_get_opportunistic

## 6.137 preauth_test.c File Reference

WPA Supplicant - test code for pre-authentication.

```
#include "includes.h"
#include <assert.h>
#include "common.h"
#include "config.h"
#include "eapol_sm.h"
#include "eloop.h"
#include "wpa.h"
#include "eap.h"
#include "wpa_supplicant.h"
#include "wpa_supplicant_i.h"
#include "l2_packet.h"
#include "ctrl_iface.h"
#include "pcsc_funcs.h"
#include "preauth.h"
#include "pmksa_cache.h"
```

Include dependency graph for preauth_test.c:




## Functions

- void **wpa_supplicant_scan** (void *eloop_ctx, void *timeout_ctx)
- int **main** (int argc, char *argv[ ])

## Variables

- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- wpa_driver_ops * **wpa_supplicant_drivers** [ ] = { NULL }

### 6.137.1 Detailed Description

WPA Supplicant - test code for pre-authentication.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

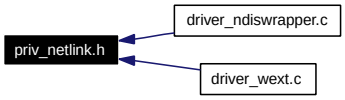
IEEE 802.1X Supplicant test code (to be used in place of wpa_supplicant.c. Not used in production version.

Definition in file preauth_test.c.

# 6.138 priv_netlink.h File Reference

wpa_supplicant - Private copy of Linux netlink/rtnetlink definitions.

This graph shows which files directly or indirectly include this file:




## Defines

- #define **IFF_LOWER_UP** 0x10000
- #define **IFF_DORMANT** 0x20000
- #define **IFLA_IFNAME** 3
- #define **IFLA_WIRELESS** 11
- #define **IFLA_OPERSTATE** 16
- #define **IFLA_LINKMODE** 17
- #define **IF_OPER_DORMANT** 5
- #define **IF_OPER_UP** 6
- #define **NLM_F_REQUEST** 1
- #define **NETLINK_ROUTE** 0
- #define **RTMGRP_LINK** 1
- #define **RTM_BASE** 0x10
- #define **RTM_NEWLINK** (RTM_BASE + 0)
- #define **RTM_DELLINK** (RTM_BASE + 1)
- #define **RTM_SETLINK** (RTM_BASE + 3)
- #define **NLMSG_ALIGNTO** 4
- #define **NLMSG_ALIGN**(len) (((len) + NLMSG_ALIGNTO - 1) & ~(NLMSG_ALIGNTO - 1))
- #define **NLMSG_LENGTH**(len) ((len) + NLMSG_ALIGN(sizeof(struct nlmsghdr)))
- #define **NLMSG_DATA**(nlh) ((void*) (((char*) nlh) + NLMSG_LENGTH(0)))
- #define **RTA_ALIGNTO** 4
- #define **RTA_ALIGN**(len) (((len) + RTA_ALIGNTO - 1) & ~(RTA_ALIGNTO - 1))
- #define **RTA_OK**(rta, len)
- #define **RTA_NEXT**(rta, attrlen)
- #define **RTA_LENGTH**(len) (RTA_ALIGN(sizeof(struct rtattr)) + (len))
- #define **RTA_DATA**(rta) ((void *) (((char *) (rta)) + RTA_LENGTH(0)))

### 6.138.1 Detailed Description

wpa_supplicant - Private copy of Linux netlink/rtnetlink definitions.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file priv_netlink.h.

## 6.138.2 Define Documentation

### 6.138.2.1 #define RTA_NEXT(rta, attrlen)

**Value:**

```
((attrlen) -= RTA_ALIGN((rta)->rta_len), \
(struct rtattr *) (((char *)(rta)) + RTA_ALIGN((rta)->rta_len)))
```

Definition at line 65 of file priv_netlink.h.

### 6.138.2.2 #define RTA_OK(rta, len)

**Value:**

```
((len) > 0 && (rta)->rta_len >= sizeof(struct rtattr) && \
(rta)->rta_len <= (len))
```

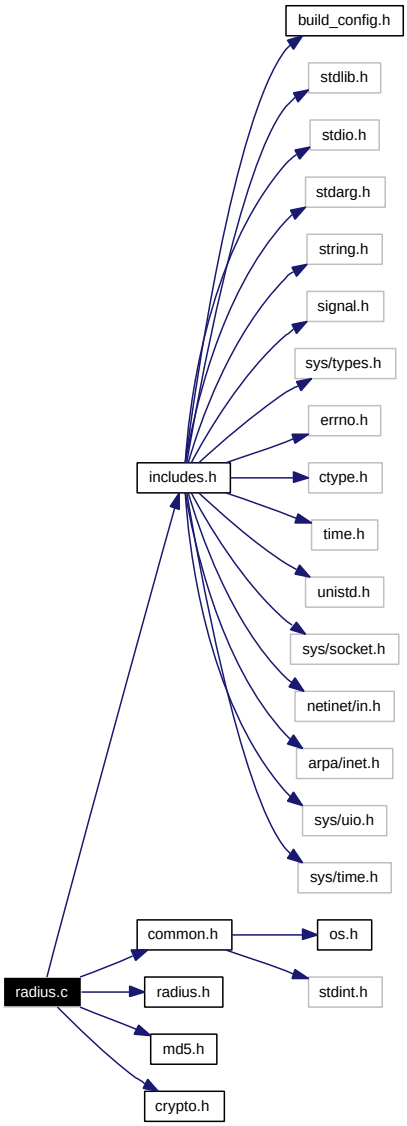
Definition at line 62 of file priv_netlink.h.

## 6.139 radius.c File Reference

hostapd / RADIUS message processing

```
#include "includes.h"
#include "common.h"
#include "radius.h"
#include "md5.h"
#include "crypto.h"
```

Include dependency graph for radius.c:



## Defines

- #define **RADIUS_ATTRS** (sizeof(radius_attrs) / sizeof(radius_attrs[0]))

## Functions

- radius_msg * **radius_msg_new** (u8 code, u8 identifier)
- int **radius_msg_initialize** (struct radius_msg *msg, size_t init_len)
- void **radius_msg_set_hdr** (struct radius_msg *msg, u8 code, u8 identifier)
- void **radius_msg_free** (struct radius_msg *msg)
- void **radius_msg_dump** (struct radius_msg *msg)
- int **radius_msg_finish** (struct radius_msg *msg, u8 *secret, size_t secret_len)
- int **radius_msg_finish_srv** (struct radius_msg *msg, const u8 *secret, size_t secret_len, const u8 *req_authenticator)
- void **radius_msg_finish_acct** (struct radius_msg *msg, u8 *secret, size_t secret_len)
- radius_attr_hdr * **radius_msg_add_attr** (struct radius_msg *msg, u8 type, const u8 *data, size_t data_len)
- radius_msg * **radius_msg_parse** (const u8 *data, size_t len)
- int **radius_msg_add_eap** (struct radius_msg *msg, const u8 *data, size_t data_len)
- u8 * **radius_msg_get_eap** (struct radius_msg *msg, size_t *eap_len)
- int **radius_msg_verify_msg_auth** (struct radius_msg *msg, const u8 *secret, size_t secret_len, const u8 *req_auth)
- int **radius_msg_verify** (struct radius_msg *msg, const u8 *secret, size_t secret_len, struct radius_msg *sent_msg, int auth)
- int **radius_msg_copy_attr** (struct radius_msg *dst, struct radius_msg *src, u8 type)
- void **radius_msg_make_authenticator** (struct radius_msg *msg, const u8 *data, size_t len)
- radius_ms_mppe_keys * **radius_msg_get_ms_keys** (struct radius_msg *msg, struct radius_msg *sent_msg, u8 *secret, size_t secret_len)
- radius_ms_mppe_keys * **radius_msg_get_cisco_keys** (struct radius_msg *msg, struct radius_msg *sent_msg, u8 *secret, size_t secret_len)
- int **radius_msg_add_mppe_keys** (struct radius_msg *msg, const u8 *req_authenticator, const u8 *secret, size_t secret_len, const u8 *send_key, size_t send_key_len, const u8 *recv_key, size_t recv_key_len)
- radius_attr_hdr * **radius_msg_add_attr_user_password** (struct radius_msg *msg, u8 *data, size_t data_len, u8 *secret, size_t secret_len)
- int **radius_msg_get_attr** (struct radius_msg *msg, u8 type, u8 *buf, size_t len)
- int **radius_msg_get_attr_ptr** (struct radius_msg *msg, u8 type, u8 **buf, size_t *len, const u8 *start)
- int **radius_msg_count_attr** (struct radius_msg *msg, u8 type, int min_len)
- int radius_msg_get_vlanid (struct radius_msg *msg)

  *Parse RADIUS attributes for VLAN tunnel information.*

### 6.139.1 Detailed Description

hostapd / RADIUS message processing

**Copyright**

Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file radius.c.

## 6.139.2 Function Documentation

### 6.139.2.1 int radius_msg_get_vlanid (struct radius_msg * *msg*)

Parse RADIUS attributes for VLAN tunnel information.

**Parameters:**
*msg* RADIUS message

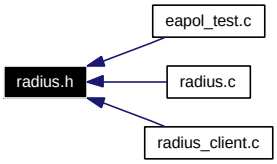
**Returns:**
VLAN ID for the first tunnel configuration of -1 if none is found

Definition at line 1170 of file radius.c.

# 6.140 radius.h File Reference

hostapd / RADIUS message processing

This graph shows which files directly or indirectly include this file:




## Defines

- #define **RADIUS_MAX_ATTR_LEN** (255 - sizeof(struct radius_attr_hdr))
- #define **RADIUS_TERMINATION_ACTION_DEFAULT** 0
- #define **RADIUS_TERMINATION_ACTION_RADIUS_REQUEST** 1
- #define **RADIUS_NAS_PORT_TYPE_IEEE_802_11** 19
- #define **RADIUS_ACCT_STATUS_TYPE_START** 1
- #define **RADIUS_ACCT_STATUS_TYPE_STOP** 2
- #define **RADIUS_ACCT_STATUS_TYPE_INTERIM_UPDATE** 3
- #define **RADIUS_ACCT_STATUS_TYPE_ACCOUNTING_ON** 7
- #define **RADIUS_ACCT_STATUS_TYPE_ACCOUNTING_OFF** 8
- #define **RADIUS_ACCT_AUTHENTIC_RADIUS** 1
- #define **RADIUS_ACCT_AUTHENTIC_LOCAL** 2
- #define **RADIUS_ACCT_AUTHENTIC_REMOTE** 3
- #define **RADIUS_ACCT_TERMINATE_CAUSE_USER_REQUEST** 1
- #define **RADIUS_ACCT_TERMINATE_CAUSE_LOST_CARRIER** 2
- #define **RADIUS_ACCT_TERMINATE_CAUSE_LOST_SERVICE** 3
- #define **RADIUS_ACCT_TERMINATE_CAUSE_IDLE_TIMEOUT** 4
- #define **RADIUS_ACCT_TERMINATE_CAUSE_SESSION_TIMEOUT** 5
- #define **RADIUS_ACCT_TERMINATE_CAUSE_ADMIN_RESET** 6
- #define **RADIUS_ACCT_TERMINATE_CAUSE_ADMIN_REBOOT** 7
- #define **RADIUS_ACCT_TERMINATE_CAUSE_PORT_ERROR** 8
- #define **RADIUS_ACCT_TERMINATE_CAUSE_NAS_ERROR** 9
- #define **RADIUS_ACCT_TERMINATE_CAUSE_NAS_REQUEST** 10
- #define **RADIUS_ACCT_TERMINATE_CAUSE_NAS_REBOOT** 11
- #define **RADIUS_ACCT_TERMINATE_CAUSE_PORT_UNNEEDED** 12
- #define **RADIUS_ACCT_TERMINATE_CAUSE_PORT_PREEMPTED** 13
- #define **RADIUS_ACCT_TERMINATE_CAUSE_PORT_SUSPENDED** 14
- #define **RADIUS_ACCT_TERMINATE_CAUSE_SERVICE_UNAVAILABLE** 15
- #define **RADIUS_ACCT_TERMINATE_CAUSE_CALLBACK** 16
- #define **RADIUS_ACCT_TERMINATE_CAUSE_USER_ERROR** 17
- #define **RADIUS_ACCT_TERMINATE_CAUSE_HOST_REQUEST** 18
- #define **RADIUS_TUNNEL_TAGS** 32
- #define **RADIUS_TUNNEL_TYPE_PPTP** 1
- #define **RADIUS_TUNNEL_TYPE_L2TP** 3
- #define **RADIUS_TUNNEL_TYPE_IPIP** 7
- #define **RADIUS_TUNNEL_TYPE_GRE** 10

- #define **RADIUS_TUNNEL_TYPE_VLAN** 13
- #define **RADIUS_TUNNEL_MEDIUM_TYPE_IPV4** 1
- #define **RADIUS_TUNNEL_MEDIUM_TYPE_IPV6** 2
- #define **RADIUS_TUNNEL_MEDIUM_TYPE_802** 6
- #define **RADIUS_VENDOR_ID_CISCO** 9
- #define **RADIUS_CISCO_AV_PAIR** 1
- #define **RADIUS_VENDOR_ID_MICROSOFT** 311
- #define **RADIUS_DEFAULT_MSG_SIZE** 1024
- #define **RADIUS_DEFAULT_ATTR_COUNT** 16
- #define **RADIUS_802_1X_ADDR_FORMAT** "%02X-%02X-%02X-%02X-%02X-%02X"
- #define **RADIUS_ADDR_FORMAT** "%02x%02x%02x%02x%02x%02x"

## Enumerations

- enum {

  **RADIUS_CODE_ACCESS_REQUEST** = 1, **RADIUS_CODE_ACCESS_ACCEPT** = 2, **RADIUS_CODE_ACCESS_REJECT** = 3, **RADIUS_CODE_ACCOUNTING_REQUEST** = 4,

  **RADIUS_CODE_ACCOUNTING_RESPONSE** = 5, **RADIUS_CODE_ACCESS_CHALLENGE** = 11, **RADIUS_CODE_STATUS_SERVER** = 12, **RADIUS_CODE_STATUS_CLIENT** = 13,

  **RADIUS_CODE_RESERVED** = 255 }
- enum {

  **RADIUS_ATTR_USER_NAME** = 1, **RADIUS_ATTR_USER_PASSWORD** = 2, **RADIUS_ATTR_NAS_IP_ADDRESS** = 4, **RADIUS_ATTR_NAS_PORT** = 5,

  **RADIUS_ATTR_FRAMED_MTU** = 12, **RADIUS_ATTR_STATE** = 24, **RADIUS_ATTR_CLASS** = 25, **RADIUS_ATTR_VENDOR_SPECIFIC** = 26,

  **RADIUS_ATTR_SESSION_TIMEOUT** = 27, **RADIUS_ATTR_IDLE_TIMEOUT** = 28, **RADIUS_ATTR_TERMINATION_ACTION** = 29, **RADIUS_ATTR_CALLED_STATION_ID** = 30,

  **RADIUS_ATTR_CALLING_STATION_ID** = 31, **RADIUS_ATTR_NAS_IDENTIFIER** = 32, **RADIUS_ATTR_ACCT_STATUS_TYPE** = 40, **RADIUS_ATTR_ACCT_DELAY_TIME** = 41,

  **RADIUS_ATTR_ACCT_INPUT_OCTETS** = 42, **RADIUS_ATTR_ACCT_OUTPUT_OCTETS** = 43, **RADIUS_ATTR_ACCT_SESSION_ID** = 44, **RADIUS_ATTR_ACCT_AUTHENTIC** = 45,

  **RADIUS_ATTR_ACCT_SESSION_TIME** = 46, **RADIUS_ATTR_ACCT_INPUT_PACKETS** = 47, **RADIUS_ATTR_ACCT_OUTPUT_PACKETS** = 48, **RADIUS_ATTR_ACCT_TERMINATE_CAUSE** = 49,

  **RADIUS_ATTR_ACCT_MULTI_SESSION_ID** = 50, **RADIUS_ATTR_ACCT_LINK_COUNT** = 51, **RADIUS_ATTR_ACCT_INPUT_GIGAWORDS** = 52, **RADIUS_ATTR_ACCT_OUTPUT_GIGAWORDS** = 53,

  **RADIUS_ATTR_EVENT_TIMESTAMP** = 55, **RADIUS_ATTR_NAS_PORT_TYPE** = 61, **RADIUS_ATTR_TUNNEL_TYPE** = 64, **RADIUS_ATTR_TUNNEL_MEDIUM_TYPE** = 65,

  **RADIUS_ATTR_CONNECT_INFO** = 77, **RADIUS_ATTR_EAP_MESSAGE** = 79, **RADIUS_ATTR_MESSAGE_AUTHENTICATOR** = 80, **RADIUS_ATTR_TUNNEL_PRIVATE_GROUP_ID** = 81,

  **RADIUS_ATTR_ACCT_INTERIM_INTERVAL** = 85, **RADIUS_ATTR_NAS_IPV6_ADDRESS** = 95 }
- enum { **RADIUS_VENDOR_ATTR_MS_MPPE_SEND_KEY** = 16, **RADIUS_VENDOR_ATTR_MS_MPPE_RECV_KEY** = 17 }

## Functions

- radius_msg * **radius_msg_new** (u8 code, u8 identifier)
- int **radius_msg_initialize** (struct radius_msg *msg, size_t init_len)
- void **radius_msg_set_hdr** (struct radius_msg *msg, u8 code, u8 identifier)
- void **radius_msg_free** (struct radius_msg *msg)
- void **radius_msg_dump** (struct radius_msg *msg)
- int **radius_msg_finish** (struct radius_msg *msg, u8 *secret, size_t secret_len)
- int **radius_msg_finish_srv** (struct radius_msg *msg, const u8 *secret, size_t secret_len, const u8 *req_authenticator)
- void **radius_msg_finish_acct** (struct radius_msg *msg, u8 *secret, size_t secret_len)
- radius_attr_hdr * **radius_msg_add_attr** (struct radius_msg *msg, u8 type, const u8 *data, size_t data_len)
- radius_msg * **radius_msg_parse** (const u8 *data, size_t len)
- int **radius_msg_add_eap** (struct radius_msg *msg, const u8 *data, size_t data_len)
- u8 * **radius_msg_get_eap** (struct radius_msg *msg, size_t *len)
- int **radius_msg_verify** (struct radius_msg *msg, const u8 *secret, size_t secret_len, struct radius_msg *sent_msg, int auth)
- int **radius_msg_verify_msg_auth** (struct radius_msg *msg, const u8 *secret, size_t secret_len, const u8 *req_auth)
- int **radius_msg_copy_attr** (struct radius_msg *dst, struct radius_msg *src, u8 type)
- void **radius_msg_make_authenticator** (struct radius_msg *msg, const u8 *data, size_t len)
- radius_ms_mppe_keys * **radius_msg_get_ms_keys** (struct radius_msg *msg, struct radius_msg *sent_msg, u8 *secret, size_t secret_len)
- radius_ms_mppe_keys * **radius_msg_get_cisco_keys** (struct radius_msg *msg, struct radius_msg *sent_msg, u8 *secret, size_t secret_len)
- int **radius_msg_add_mppe_keys** (struct radius_msg *msg, const u8 *req_authenticator, const u8 *secret, size_t secret_len, const u8 *send_key, size_t send_key_len, const u8 *recv_key, size_t recv_key_len)
- radius_attr_hdr * **radius_msg_add_attr_user_password** (struct radius_msg *msg, u8 *data, size_t data_len, u8 *secret, size_t secret_len)
- int **radius_msg_get_attr** (struct radius_msg *msg, u8 type, u8 *buf, size_t len)
- int radius_msg_get_vlanid (struct radius_msg *msg)

  *Parse RADIUS attributes for VLAN tunnel information.*

- int **radius_msg_get_attr_ptr** (struct radius_msg *msg, u8 type, u8 **buf, size_t *len, const u8 *start)
- int **radius_msg_count_attr** (struct radius_msg *msg, u8 type, int min_len)

## Variables

- radius_hdr **STRUCT_PACKED**

### 6.140.1 Detailed Description

hostapd / RADIUS message processing

**Copyright**

Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file radius.h.

## 6.140.2 Function Documentation

### 6.140.2.1 int radius_msg_get_vlanid (struct radius_msg * *msg*)

Parse RADIUS attributes for VLAN tunnel information.

**Parameters:**
*msg* RADIUS message

**Returns:**
VLAN ID for the first tunnel configuration of -1 if none is found

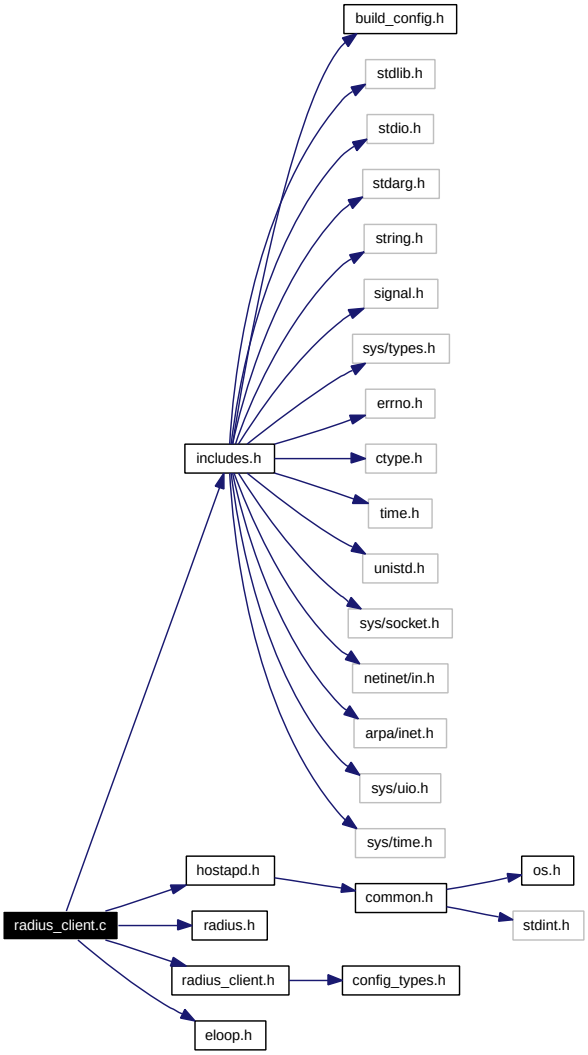
Definition at line 1170 of file radius.c.

# 6.141 radius_client.c File Reference

hostapd / RADIUS client

```
#include "includes.h"
#include "hostapd.h"
#include "radius.h"
#include "radius_client.h"
#include "eloop.h"
```

Include dependency graph for radius_client.c:




## Defines

- #define **RADIUS_CLIENT_FIRST_WAIT** 3
- #define **RADIUS_CLIENT_MAX_WAIT** 120
- #define **RADIUS_CLIENT_MAX_RETRIES** 10

- #define **RADIUS_CLIENT_MAX_ENTRIES** 30
- #define **RADIUS_CLIENT_NUM_FAILOVER** 4

## Functions

- int **radius_client_register** (struct radius_client_data *radius, RadiusType msg_type, RadiusRxResult(*handler)(struct radius_msg *msg, struct radius_msg *req, u8 *shared_secret, size_t shared_secret_len, void *data), void *data)
- int **radius_client_send** (struct radius_client_data *radius, struct radius_msg *msg, RadiusType msg_type, const u8 *addr)
- u8 **radius_client_get_id** (struct radius_client_data *radius)
- void **radius_client_flush** (struct radius_client_data *radius, int only_auth)
- void **radius_client_update_acct_msgs** (struct radius_client_data *radius, u8 *shared_secret, size_t shared_secret_len)
- radius_client_data * **radius_client_init** (void *ctx, struct hostapd_radius_servers *conf)
- void **radius_client_deinit** (struct radius_client_data *radius)
- void **radius_client_flush_auth** (struct radius_client_data *radius, u8 *addr)
- int **radius_client_get_mib** (struct radius_client_data *radius, char *buf, size_t buflen)
- radius_client_data * **radius_client_reconfig** (struct radius_client_data *old, void *ctx, struct hostapd_radius_servers *oldconf, struct hostapd_radius_servers *newconf)

### 6.141.1 Detailed Description

hostapd / RADIUS client

**Copyright**

Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>



Definition in file radius_client.c.

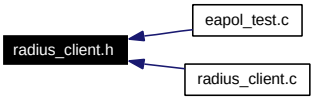
## 6.142 radius_client.h File Reference

hostapd / RADIUS client

```
#include "config_types.h"
```

Include dependency graph for radius_client.h:


This graph shows which files directly or indirectly include this file:




### Enumerations

- enum **RadiusType** { **RADIUS_AUTH**, **RADIUS_ACCT**, **RADIUS_ACCT_INTERIM** }
- enum **RadiusRxResult** { **RADIUS_RX_PROCESSED**, **RADIUS_RX_QUEUED**, **RADIUS_RX_UNKNOWN**, **RADIUS_RX_INVALID_AUTHENTICATOR** }

### Functions

- int **radius_client_register** (struct radius_client_data *radius, RadiusType msg_type, RadiusRxResult(*handler)(struct radius_msg *msg, struct radius_msg *req, u8 *shared_secret, size_t shared_secret_len, void *data), void *data)
- int **radius_client_send** (struct radius_client_data *radius, struct radius_msg *msg, RadiusType msg_type, const u8 *addr)
- u8 **radius_client_get_id** (struct radius_client_data *radius)
- void **radius_client_flush** (struct radius_client_data *radius, int only_auth)
- radius_client_data * **radius_client_init** (void *ctx, struct hostapd_radius_servers *conf)
- void **radius_client_deinit** (struct radius_client_data *radius)
- void **radius_client_flush_auth** (struct radius_client_data *radius, u8 *addr)
- int **radius_client_get_mib** (struct radius_client_data *radius, char *buf, size_t buflen)
- radius_client_data * **radius_client_reconfig** (struct radius_client_data *old, void *ctx, struct hostapd_radius_servers *oldconf, struct hostapd_radius_servers *newconf)

### 6.142.1 Detailed Description

hostapd / RADIUS client

**Copyright**

Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

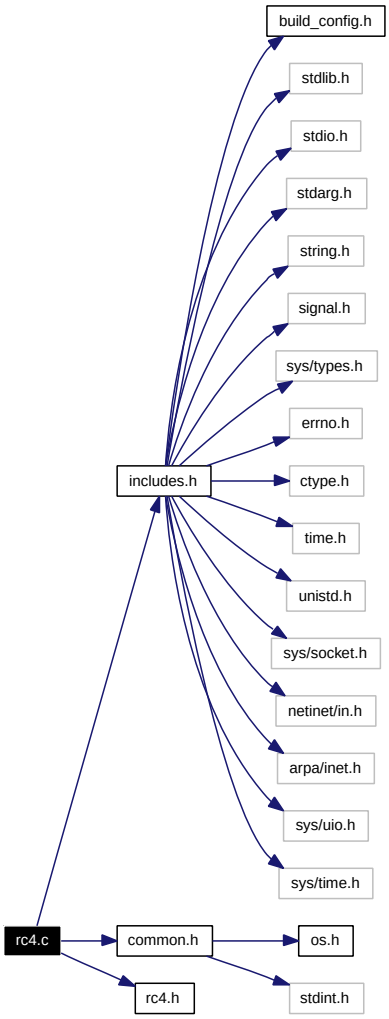
Definition in file radius_client.h.

# 6.143 rc4.c File Reference

RC4 stream cipher.

```
#include "includes.h"
#include "common.h"
#include "rc4.h"
```

Include dependency graph for rc4.c:




## Defines

- #define **S_SWAP**(a, b) do { u8 t = S[a]; S[a] = S[b]; S[b] = t; } while(0)

## Functions

- void rc4_skip (const u8 *key, size_t keylen, size_t skip, u8 *data, size_t data_len)

  *XOR RC4 stream to given data with skip-stream-start.*

- void rc4 (u8 *buf, size_t len, const u8 *key, size_t key_len)
  *XOR RC4 stream to given data.*

### 6.143.1 Detailed Description

RC4 stream cipher.

**Copyright**
Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file rc4.c.

### 6.143.2 Function Documentation

#### 6.143.2.1 void rc4 (u8 * *buf*, size_t *len*, const u8 * *key*, size_t *key_len*)

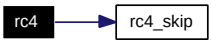
XOR RC4 stream to given data.

**Parameters:**
***buf*** data to be XOR'ed with RC4 stream
***len*** buf length
***key*** RC4 key
***key_len*** RC4 key length

Generate RC4 pseudo random stream for the given key and XOR this with the data buffer to perform RC4 encryption/decryption.

Definition at line 86 of file rc4.c.

Here is the call graph for this function:




#### 6.143.2.2 void rc4_skip (const u8 * *key*, size_t *keylen*, size_t *skip*, u8 * *data*, size_t *data_len*)

XOR RC4 stream to given data with skip-stream-start.

**Parameters:**
***key*** RC4 key
***keylen*** RC4 key length

***skip*** number of bytes to skip from the beginning of the RC4 stream

***data*** data to be XOR'ed with RC4 stream

***data_len*** buf length

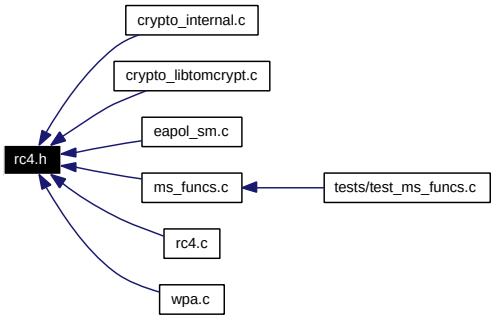
Generate RC4 pseudo random stream for the given key, skip beginning of the stream, and XOR the end result with the data buffer to perform RC4 encryption/decryption.

Definition at line 36 of file rc4.c.

## 6.144 rc4.h File Reference

RC4 stream cipher.

This graph shows which files directly or indirectly include this file:




### Functions

- void rc4_skip (const u8 *key, size_t keylen, size_t skip, u8 *data, size_t data_len)
  *XOR RC4 stream to given data with skip-stream-start.*
- void rc4 (u8 *buf, size_t len, const u8 *key, size_t key_len)
  *XOR RC4 stream to given data.*

### 6.144.1 Detailed Description

RC4 stream cipher.

**Copyright**
Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file rc4.h.

### 6.144.2 Function Documentation

#### 6.144.2.1 void rc4 (u8 * *buf*, size_t *len*, const u8 * *key*, size_t *key_len*)

XOR RC4 stream to given data.

**Parameters:**
***buf*** data to be XOR'ed with RC4 stream
***len*** buf length

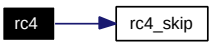
***key*** RC4 key

***key_len*** RC4 key length

Generate RC4 pseudo random stream for the given key and XOR this with the data buffer to perform RC4 encryption/decryption.

Definition at line 86 of file rc4.c.

Here is the call graph for this function:




#### 6.144.2.2 void rc4_skip (const u8 * *key*, size_t *keylen*, size_t *skip*, u8 * *data*, size_t *data_len*)

XOR RC4 stream to given data with skip-stream-start.

**Parameters:**

***key*** RC4 key

***keylen*** RC4 key length

***skip*** number of bytes to skip from the beginning of the RC4 stream

***data*** data to be XOR'ed with RC4 stream

***data_len*** buf length

Generate RC4 pseudo random stream for the given key, skip beginning of the stream, and XOR the end result with the data buffer to perform RC4 encryption/decryption.

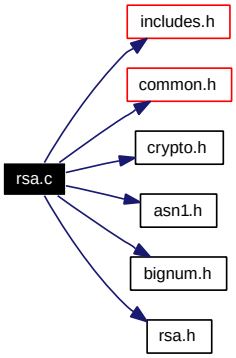
Definition at line 36 of file rc4.c.

## 6.145 rsa.c File Reference

RSA.

```
#include "includes.h"
#include "common.h"
#include "crypto.h"
#include "asn1.h"
#include "bignum.h"
#include "rsa.h"
```

Include dependency graph for rsa.c:




### Functions

- crypto_rsa_key * crypto_rsa_import_public_key (const u8 *buf, size_t len)

  *Import an RSA public key.*

- crypto_rsa_key * crypto_rsa_import_private_key (const u8 *buf, size_t len)

  *Import an RSA private key.*

- size_t crypto_rsa_get_modulus_len (struct crypto_rsa_key *key)

  *Get the modulus length of the RSA key.*

- int crypto_rsa_exptmod (const u8 *in, size_t inlen, u8 *out, size_t *outlen, struct crypto_rsa_key *key, int use_private)

  *RSA modular exponentiation.*

- void crypto_rsa_free (struct crypto_rsa_key *key)

  *Free RSA key.*

### 6.145.1 Detailed Description

RSA.

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file rsa.c.

## 6.145.2 Function Documentation

### 6.145.2.1 int crypto_rsa_exptmod (const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*, struct crypto_rsa_key * *key*, int *use_private*)

RSA modular exponentiation.

**Parameters:**

***in*** Input data

***inlen*** Input data length

***out*** Buffer for output data

***outlen*** Maximum size of the output buffer and used size on success

***key*** RSA key

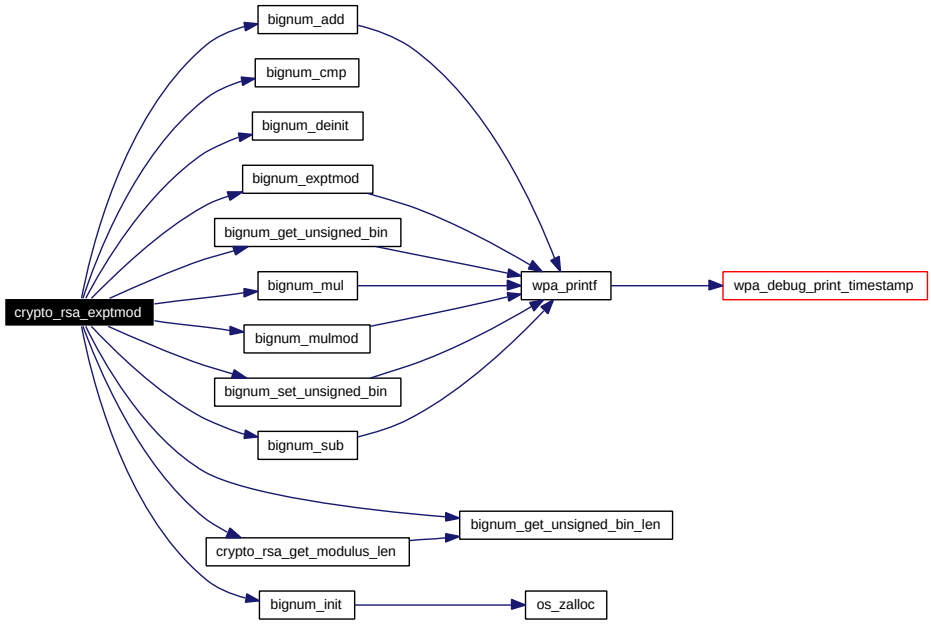
***use_private*** 1 = Use RSA private key, 0 = Use RSA public key

**Returns:**

0 on success, -1 on failure

Definition at line 254 of file rsa.c.

Here is the call graph for this function:



#### 6.145.2.2 void crypto_rsa_free (struct crypto_rsa_key * *key*)

Free RSA key.

**Parameters:**
***key*** RSA key to be freed

This function frees an RSA key imported with either crypto_rsa_import_public_key() or crypto_rsa_import_private_key().

Definition at line 352 of file rsa.c.

Here is the call graph for this function:


#### 6.145.2.3 size_t crypto_rsa_get_modulus_len (struct crypto_rsa_key * *key*)

Get the modulus length of the RSA key.

**Parameters:**
***key*** RSA key

**Returns:**
Modulus length of the key

Definition at line 237 of file rsa.c.

Here is the call graph for this function:


#### 6.145.2.4 struct crypto_rsa_key* crypto_rsa_import_private_key (const u8 * *buf*, size_t *len*)

Import an RSA private key.

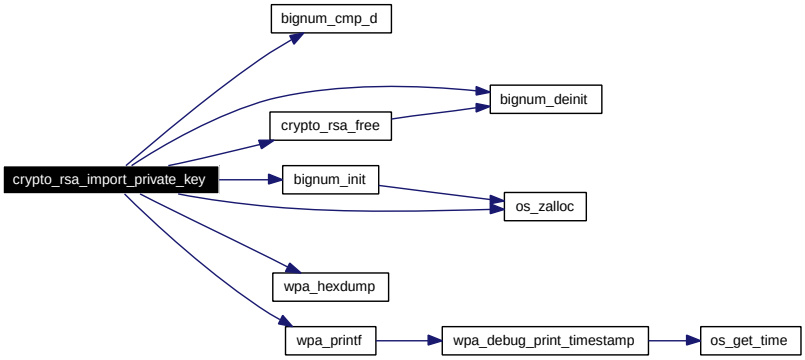
**Parameters:**
***buf*** Key buffer (DER encoded RSA private key)

***len*** Key buffer length in bytes

**Returns:**
Pointer to the private key or NULL on failure

Definition at line 136 of file rsa.c.

Here is the call graph for this function:




#### 6.145.2.5 struct crypto_rsa_key* crypto_rsa_import_public_key (const u8 * *buf*, size_t *len*)

Import an RSA public key.

**Parameters:**

***buf*** Key buffer (DER encoded RSA public key)

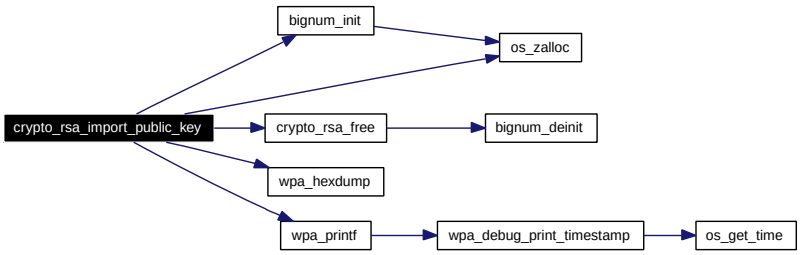
***len*** Key buffer length in bytes

**Returns:**

Pointer to the public key or NULL on failure

Definition at line 71 of file rsa.c.

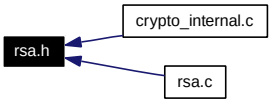
Here is the call graph for this function:



## 6.146 rsa.h File Reference

RSA.

This graph shows which files directly or indirectly include this file:




### Functions

- crypto_rsa_key * crypto_rsa_import_public_key (const u8 *buf, size_t len)
  *Import an RSA public key.*
- crypto_rsa_key * crypto_rsa_import_private_key (const u8 *buf, size_t len)
  *Import an RSA private key.*
- size_t crypto_rsa_get_modulus_len (struct crypto_rsa_key *key)
  *Get the modulus length of the RSA key.*
- int crypto_rsa_exptmod (const u8 *in, size_t inlen, u8 *out, size_t *outlen, struct crypto_rsa_key *key, int use_private)
  *RSA modular exponentiation.*
- void crypto_rsa_free (struct crypto_rsa_key *key)
  *Free RSA key.*

### 6.146.1 Detailed Description

RSA.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file rsa.h.

### 6.146.2 Function Documentation

#### 6.146.2.1 int crypto_rsa_exptmod (const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*, struct crypto_rsa_key * *key*, int *use_private*)

RSA modular exponentiation.

**Parameters:**

***in*** Input data

***inlen*** Input data length

***out*** Buffer for output data

***outlen*** Maximum size of the output buffer and used size on success

***key*** RSA key

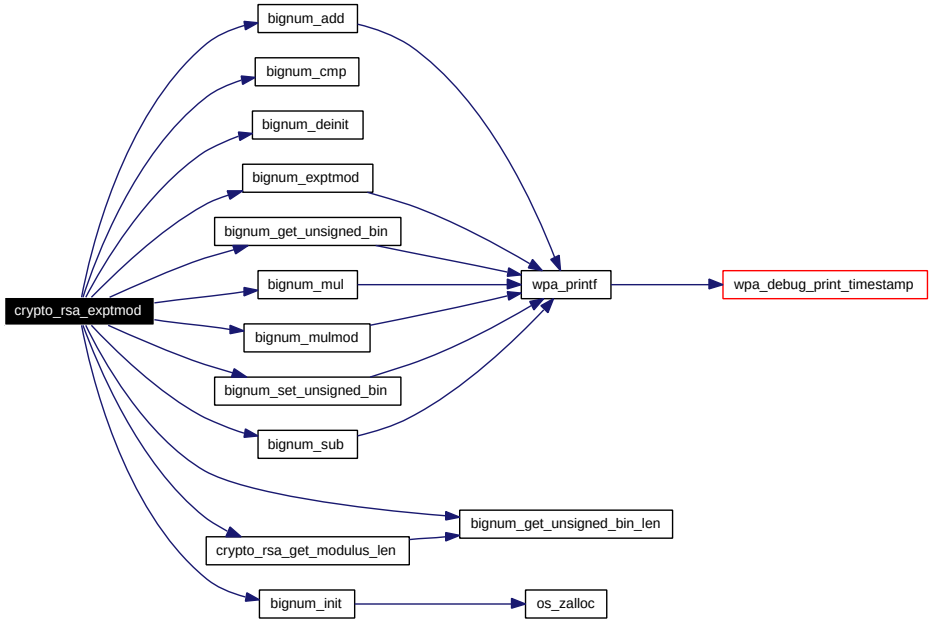
***use_private*** 1 = Use RSA private key, 0 = Use RSA public key

**Returns:**

0 on success, -1 on failure

Definition at line 254 of file rsa.c.

Here is the call graph for this function:




### 6.146.2.2 void crypto_rsa_free (struct crypto_rsa_key * *key*)

Free RSA key.

**Parameters:**

***key*** RSA key to be freed

This function frees an RSA key imported with either crypto_rsa_import_public_key() or crypto_rsa_import_private_key().

Definition at line 352 of file rsa.c.

Here is the call graph for this function:

#### 6.146.2.3 size_t crypto_rsa_get_modulus_len (struct crypto_rsa_key * *key*)

Get the modulus length of the RSA key.

**Parameters:**
***key*** RSA key

**Returns:**
Modulus length of the key

Definition at line 237 of file rsa.c.

Here is the call graph for this function:


#### 6.146.2.4 struct crypto_rsa_key* crypto_rsa_import_private_key (const u8 * *buf*, size_t *len*)

Import an RSA private key.

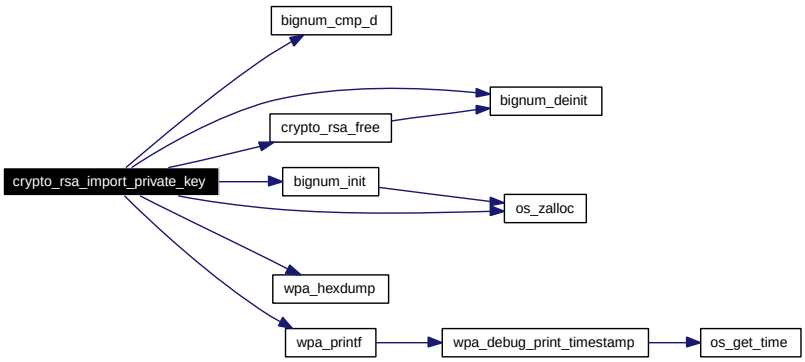
**Parameters:**
***buf*** Key buffer (DER encoded RSA private key)

***len*** Key buffer length in bytes

**Returns:**
Pointer to the private key or NULL on failure

Definition at line 136 of file rsa.c.

Here is the call graph for this function:




#### 6.146.2.5 struct crypto_rsa_key* crypto_rsa_import_public_key (const u8 * *buf*, size_t *len*)

Import an RSA public key.

**Parameters:**

***buf*** Key buffer (DER encoded RSA public key)

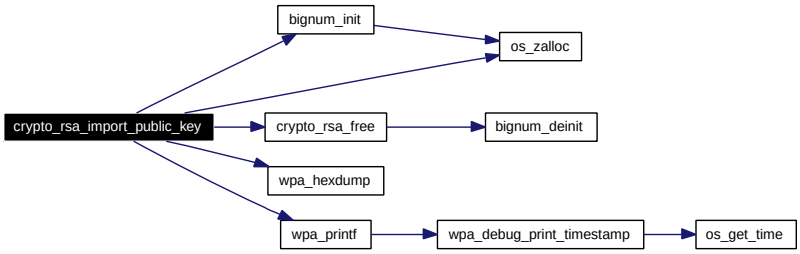
***len*** Key buffer length in bytes

**Returns:**

Pointer to the public key or NULL on failure

Definition at line 71 of file rsa.c.

Here is the call graph for this function:

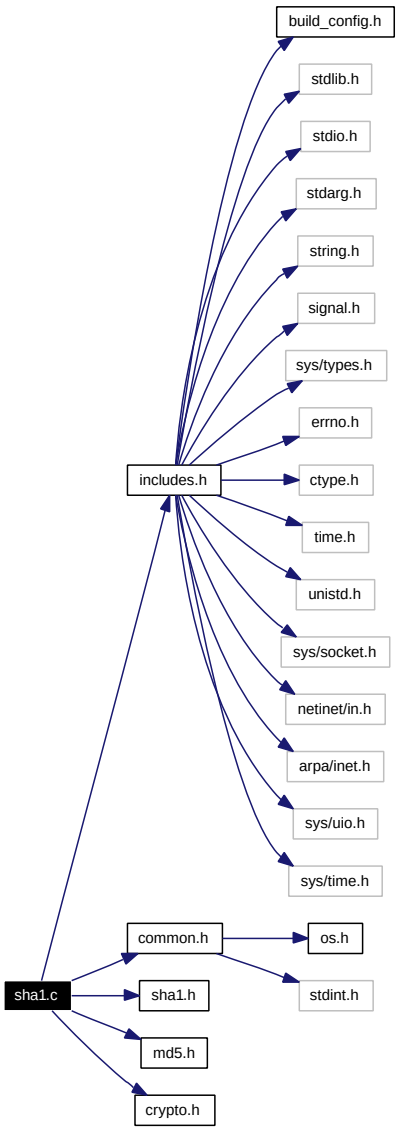


## 6.147 sha1.c File Reference

SHA1 hash implementation and interface functions.

```
#include "includes.h"
#include "common.h"
#include "sha1.h"
#include "md5.h"
#include "crypto.h"
```

Include dependency graph for sha1.c:



## Functions

- void hmac_sha1_vector (const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *HMAC-SHA1 over data vector (RFC 2104).*

- void hmac_sha1 (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)

  *HMAC-SHA1 over data buffer (RFC 2104).*

- void sha1_prf (const u8 *key, size_t key_len, const char *label, const u8 *data, size_t data_len, u8 *buf, size_t buf_len)

  *SHA1-based Pseudo-Random Function (PRF) (IEEE 802.11i, 8.5.1.1).*

- void sha1_t_prf (const u8 *key, size_t key_len, const char *label, const u8 *seed, size_t seed_len, u8 *buf, size_t buf_len)

  *EAP-FAST Pseudo-Random Function (T-PRF).*

- int tls_prf (const u8 *secret, size_t secret_len, const char *label, const u8 *seed, size_t seed_len, u8 *out, size_t outlen)

  *Pseudo-Random Function for TLS (TLS-PRF, RFC 2246).*

- void pbkdf2_sha1 (const char *passphrase, const char *ssid, size_t ssid_len, int iterations, u8 *buf, size_t buflen)

  *SHA1-based key derivation function (PBKDF2) for IEEE 802.11i.*

### 6.147.1 Detailed Description

SHA1 hash implementation and interface functions.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file sha1.c.

### 6.147.2 Function Documentation

#### 6.147.2.1 void hmac_sha1 (const u8 * *key*, size_t *key_len*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

HMAC-SHA1 over data buffer (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

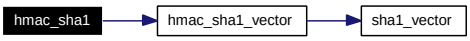
***data*** Pointers to the data area

***data_len*** Length of the data area

***mac*** Buffer for the hash (20 bytes)

Definition at line 106 of file sha1.c.

Here is the call graph for this function:




#### 6.147.2.2 void hmac_sha1_vector (const u8 * *key*, size_t *key_len*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

HMAC-SHA1 over data vector (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash (20 bytes)

Definition at line 34 of file sha1.c.

Here is the call graph for this function:


#### 6.147.2.3 void pbkdf2_sha1 (const char * *passphrase*, const char * *ssid*, size_t *ssid_len*, int *iterations*, u8 * *buf*, size_t *buflen*)

SHA1-based key derivation function (PBKDF2) for IEEE 802.11i.

**Parameters:**

***passphrase*** ASCII passphrase

***ssid*** SSID

***ssid_len*** SSID length in bytes

***interations*** Number of iterations to run

***buf*** Buffer for the generated key

***buflen*** Length of the buffer in bytes

This function is used to derive PSK for WPA-PSK. For this protocol, iterations is set to 4096 and buflen to 32. This function is described in IEEE Std 802.11-2004, Clause H.4. The main construction is from PKCS#5 v2.0.

Definition at line 356 of file sha1.c.

#### 6.147.2.4 void sha1_prf (const u8 * *key*, size_t *key_len*, const char * *label*, const u8 * *data*, size_t *data_len*, u8 * *buf*, size_t *buf_len*)

SHA1-based Pseudo-Random Function (PRF) (IEEE 802.11i, 8.5.1.1).

**Parameters:**

***key*** Key for PRF

***key_len*** Length of the key in bytes

***label*** A unique label for each purpose of the PRF

***data*** Extra data to bind into the key

***data_len*** Length of the data

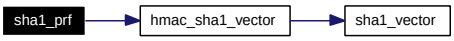
***buf*** Buffer for the generated pseudo-random key

***buf_len*** Number of bytes of key to generate

This function is used to derive new, cryptographically separate keys from a given key (e.g., PMK in IEEE 802.11i).

Definition at line 127 of file sha1.c.

Here is the call graph for this function:




#### 6.147.2.5 void sha1_t_prf (const u8 * *key*, size_t *key_len*, const char * *label*, const u8 * *seed*, size_t *seed_len*, u8 * *buf*, size_t *buf_len*)

EAP-FAST Pseudo-Random Function (T-PRF).

**Parameters:**

***key*** Key for PRF

***key_len*** Length of the key in bytes

***label*** A unique label for each purpose of the PRF

***seed*** Seed value to bind into the key

***seed_len*** Length of the seed

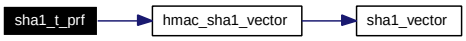
***buf*** Buffer for the generated pseudo-random key

***buf_len*** Number of bytes of key to generate

This function is used to derive new, cryptographically separate keys from a given key for EAP-FAST. T-PRF is defined in draft-cam-winget-eap-fast-02.txt, Appendix B.

Definition at line 179 of file sha1.c.

Here is the call graph for this function:



#### 6.147.2.6 int tls_prf (const u8 * *secret*, size_t *secret_len*, const char * *label*, const u8 * *seed*, size_t *seed_len*, u8 * *out*, size_t *outlen*)

Pseudo-Random Function for TLS (TLS-PRF, RFC 2246).

**Parameters:**

***secret*** Key for PRF

***secret_len*** Length of the key in bytes

***label*** A unique label for each purpose of the PRF

***seed*** Seed value to bind into the key

***seed_len*** Length of the seed

***out*** Buffer for the generated pseudo-random key

***outlen*** Number of bytes of key to generate

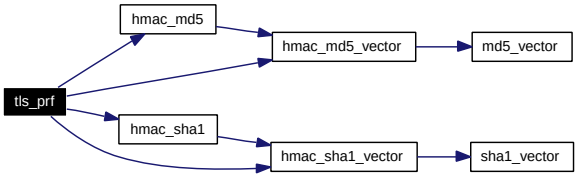
**Returns:**

0 on success, -1 on failure.

This function is used to derive new, cryptographically separate keys from a given key in TLS. This PRF is defined in RFC 2246, Chapter 5.

Definition at line 235 of file sha1.c.

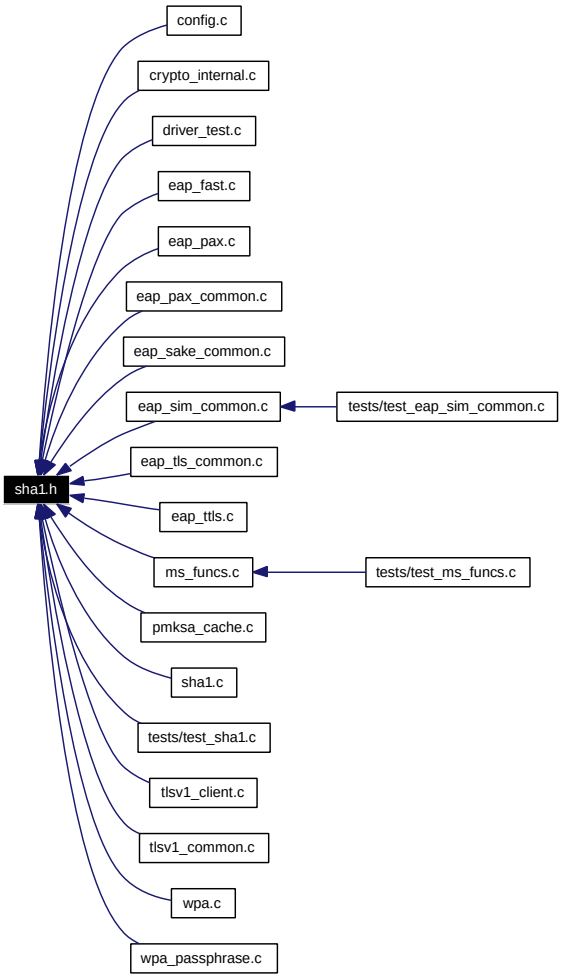
Here is the call graph for this function:



# 6.148 sha1.h File Reference

SHA1 hash implementation and interface functions.

This graph shows which files directly or indirectly include this file:




## Defines

- #define **SHA1_MAC_LEN** 20

## Functions

- void hmac_sha1_vector (const u8 ∗key, size_t key_len, size_t num_elem, const u8 ∗addr[ ], const size_t ∗len, u8 ∗mac)

  *HMAC-SHA1 over data vector (RFC 2104).*

- void hmac_sha1 (const u8 ∗key, size_t key_len, const u8 ∗data, size_t data_len, u8 ∗mac)

  *HMAC-SHA1 over data buffer (RFC 2104).*

- void sha1_prf (const u8 *key, size_t key_len, const char *label, const u8 *data, size_t data_len, u8 *buf, size_t buf_len)

  *SHA1-based Pseudo-Random Function (PRF) (IEEE 802.11i, 8.5.1.1).*

- void sha1_t_prf (const u8 *key, size_t key_len, const char *label, const u8 *seed, size_t seed_len, u8 *buf, size_t buf_len)

  *EAP-FAST Pseudo-Random Function (T-PRF).*

- int tls_prf (const u8 *secret, size_t secret_len, const char *label, const u8 *seed, size_t seed_len, u8 *out, size_t outlen)

  *Pseudo-Random Function for TLS (TLS-PRF, RFC 2246).*

- void pbkdf2_sha1 (const char *passphrase, const char *ssid, size_t ssid_len, int iterations, u8 *buf, size_t buflen)

  *SHA1-based key derivation function (PBKDF2) for IEEE 802.11i.*

## 6.148.1 Detailed Description

SHA1 hash implementation and interface functions.

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file sha1.h.

## 6.148.2 Function Documentation

### 6.148.2.1 void hmac_sha1 (const u8 * *key*, size_t *key_len*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

HMAC-SHA1 over data buffer (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

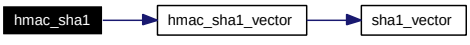
***data*** Pointers to the data area

***data_len*** Length of the data area

***mac*** Buffer for the hash (20 bytes)

Definition at line 106 of file sha1.c.

Here is the call graph for this function:



#### 6.148.2.2 void hmac_sha1_vector (const u8 * *key*, size_t *key_len*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

HMAC-SHA1 over data vector (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash (20 bytes)

Definition at line 34 of file sha1.c.

Here is the call graph for this function:


#### 6.148.2.3 void pbkdf2_sha1 (const char * *passphrase*, const char * *ssid*, size_t *ssid_len*, int *iterations*, u8 * *buf*, size_t *buflen*)

SHA1-based key derivation function (PBKDF2) for IEEE 802.11i.

**Parameters:**

***passphrase*** ASCII passphrase

***ssid*** SSID

***ssid_len*** SSID length in bytes

***interations*** Number of iterations to run

***buf*** Buffer for the generated key

***buflen*** Length of the buffer in bytes

This function is used to derive PSK for WPA-PSK. For this protocol, iterations is set to 4096 and buflen to 32. This function is described in IEEE Std 802.11-2004, Clause H.4. The main construction is from PKCS#5 v2.0.

Definition at line 356 of file sha1.c.

#### 6.148.2.4 void sha1_prf (const u8 * *key*, size_t *key_len*, const char * *label*, const u8 * *data*, size_t *data_len*, u8 * *buf*, size_t *buf_len*)

SHA1-based Pseudo-Random Function (PRF) (IEEE 802.11i, 8.5.1.1).

**Parameters:**

***key*** Key for PRF

***key_len*** Length of the key in bytes

***label*** A unique label for each purpose of the PRF

***data*** Extra data to bind into the key

***data_len*** Length of the data

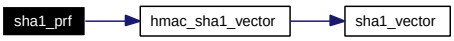
***buf*** Buffer for the generated pseudo-random key

***buf_len*** Number of bytes of key to generate

This function is used to derive new, cryptographically separate keys from a given key (e.g., PMK in IEEE 802.11i).

Definition at line 127 of file sha1.c.

Here is the call graph for this function:




#### 6.148.2.5 void sha1_t_prf (const u8 * *key*, size_t *key_len*, const char * *label*, const u8 * *seed*, size_t *seed_len*, u8 * *buf*, size_t *buf_len*)

EAP-FAST Pseudo-Random Function (T-PRF).

**Parameters:**

***key*** Key for PRF

***key_len*** Length of the key in bytes

***label*** A unique label for each purpose of the PRF

***seed*** Seed value to bind into the key

***seed_len*** Length of the seed

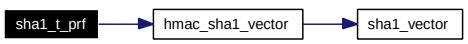
***buf*** Buffer for the generated pseudo-random key

***buf_len*** Number of bytes of key to generate

This function is used to derive new, cryptographically separate keys from a given key for EAP-FAST. T-PRF is defined in draft-cam-winget-eap-fast-02.txt, Appendix B.

Definition at line 179 of file sha1.c.

Here is the call graph for this function:




#### 6.148.2.6 int tls_prf (const u8 * *secret*, size_t *secret_len*, const char * *label*, const u8 * *seed*, size_t *seed_len*, u8 * *out*, size_t *outlen*)

Pseudo-Random Function for TLS (TLS-PRF, RFC 2246).

**Parameters:**

***secret*** Key for PRF

***secret_len*** Length of the key in bytes

***label*** A unique label for each purpose of the PRF

***seed*** Seed value to bind into the key

***seed_len*** Length of the seed

***out*** Buffer for the generated pseudo-random key

***outlen*** Number of bytes of key to generate

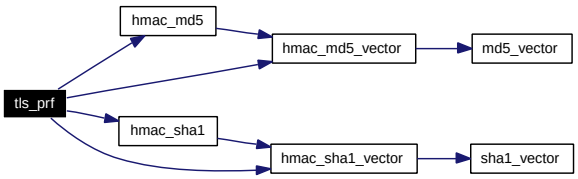
**Returns:**

0 on success, -1 on failure.

This function is used to derive new, cryptographically separate keys from a given key in TLS. This PRF is defined in RFC 2246, Chapter 5.

Definition at line 235 of file sha1.c.

Here is the call graph for this function:

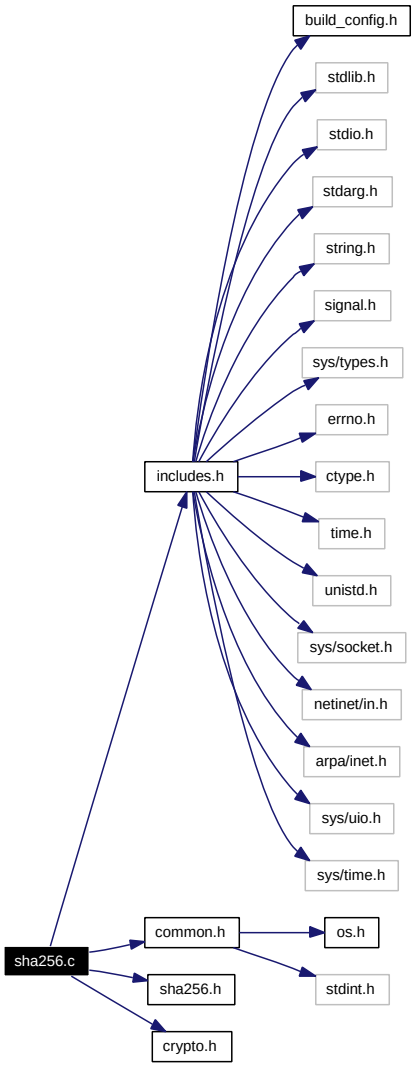


## 6.149 sha256.c File Reference

SHA-256 hash implementation and interface functions.

```
#include "includes.h"
#include "common.h"
#include "sha256.h"
#include "crypto.h"
```

Include dependency graph for sha256.c:




## Functions

- void hmac_sha256_vector (const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  *HMAC-SHA256 over data vector (RFC 2104).*

- void hmac_sha256 (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)

  *HMAC-SHA256 over data buffer (RFC 2104).*

- void sha256_prf (const u8 *key, size_t key_len, const char *label, const u8 *data, size_t data_len, u8 *buf, size_t buf_len)

  *SHA256-based Pseudo-Random Function (IEEE 802.11r, 8.5A.3).*

## 6.149.1 Detailed Description

SHA-256 hash implementation and interface functions.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file sha256.c.

## 6.149.2 Function Documentation

### 6.149.2.1 void hmac_sha256 (const u8 * *key*, size_t *key_len*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

HMAC-SHA256 over data buffer (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

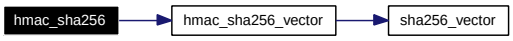
***data*** Pointers to the data area

***data_len*** Length of the data area

***mac*** Buffer for the hash (20 bytes)

Definition at line 105 of file sha256.c.

Here is the call graph for this function:




### 6.149.2.2 void hmac_sha256_vector (const u8 * *key*, size_t *key_len*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

HMAC-SHA256 over data vector (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash (32 bytes)

Definition at line 33 of file sha256.c.

Here is the call graph for this function:


#### 6.149.2.3 void sha256_prf (const u8 * *key*, size_t *key_len*, const char * *label*, const u8 * *data*, size_t *data_len*, u8 * *buf*, size_t *buf_len*)

SHA256-based Pseudo-Random Function (IEEE 802.11r, 8.5A.3).

**Parameters:**

***key*** Key for PRF

***key_len*** Length of the key in bytes

***label*** A unique label for each purpose of the PRF

***data*** Extra data to bind into the key

***data_len*** Length of the data

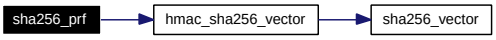
***buf*** Buffer for the generated pseudo-random key

***buf_len*** Number of bytes of key to generate

This function is used to derive new, cryptographically separate keys from a given key.

Definition at line 126 of file sha256.c.

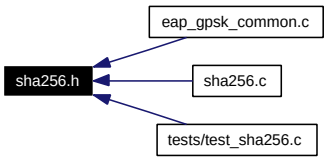
Here is the call graph for this function:



# 6.150 sha256.h File Reference

SHA256 hash implementation and interface functions.

This graph shows which files directly or indirectly include this file:




## Defines

- #define **SHA256_MAC_LEN** 32

## Functions

- void hmac_sha256_vector (const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  *HMAC-SHA256 over data vector (RFC 2104).*
- void hmac_sha256 (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)
  *HMAC-SHA256 over data buffer (RFC 2104).*
- void sha256_prf (const u8 *key, size_t key_len, const char *label, const u8 *data, size_t data_len, u8 *buf, size_t buf_len)
  *SHA256-based Pseudo-Random Function (IEEE 802.11r, 8.5A.3).*

### 6.150.1 Detailed Description

SHA256 hash implementation and interface functions.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file sha256.h.

### 6.150.2 Function Documentation

#### 6.150.2.1 void hmac_sha256 (const u8 * *key*, size_t *key_len*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

HMAC-SHA256 over data buffer (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

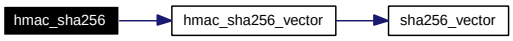
***data*** Pointers to the data area

***data_len*** Length of the data area

***mac*** Buffer for the hash (20 bytes)

Definition at line 105 of file sha256.c.

Here is the call graph for this function:




#### 6.150.2.2 void hmac_sha256_vector (const u8 * *key*, size_t *key_len*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

HMAC-SHA256 over data vector (RFC 2104).

**Parameters:**

***key*** Key for HMAC operations

***key_len*** Length of the key in bytes

***num_elem*** Number of elements in the data vector

***addr*** Pointers to the data areas

***len*** Lengths of the data blocks

***mac*** Buffer for the hash (32 bytes)

Definition at line 33 of file sha256.c.

Here is the call graph for this function:


#### 6.150.2.3 void sha256_prf (const u8 * *key*, size_t *key_len*, const char * *label*, const u8 * *data*, size_t *data_len*, u8 * *buf*, size_t *buf_len*)

SHA256-based Pseudo-Random Function (IEEE 802.11r, 8.5A.3).

**Parameters:**

***key*** Key for PRF

***key_len*** Length of the key in bytes

***label*** A unique label for each purpose of the PRF

***data*** Extra data to bind into the key

***data_len*** Length of the data

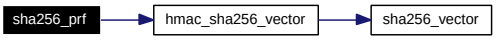
***buf*** Buffer for the generated pseudo-random key

***buf_len*** Number of bytes of key to generate

This function is used to derive new, cryptographically separate keys from a given key.

Definition at line 126 of file sha256.c.

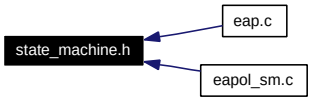
Here is the call graph for this function:



## 6.151 state_machine.h File Reference

wpa_supplicant/hostapd - State machine definitions

This graph shows which files directly or indirectly include this file:




### Defines

- #define SM_STATE(machine, state)

  *Declaration of a state machine function.*

- #define SM_ENTRY(machine, state)

  *State machine function entry point.*

- #define SM_ENTRY_M(machine, _state, data)

  *State machine function entry point for state machine group.*

- #define SM_ENTRY_MA(machine, _state, data)

  *State machine function entry point for state machine group.*

- #define SM_ENTER(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 0)

  *Enter a new state machine state.*

- #define SM_ENTER_GLOBAL(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 1)

  *Enter a new state machine state based on global rule.*

- #define SM_STEP(machine) static void sm_ ## machine ## _Step(STATE_MACHINE_DATA *sm)

  *Declaration of a state machine step function.*

- #define SM_STEP_RUN(machine) sm_ ## machine ## _Step(sm)

  *Call the state machine step function.*

### 6.151.1 Detailed Description

wpa_supplicant/hostapd - State machine definitions

**Copyright**

Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file includes a set of pre-processor macros that can be used to implement a state machine. In addition to including this header file, each file implementing a state machine must define STATE_MACHINE_DATA to be the data structure including state variables (enum <machine>_state, Boolean changed), and STATE_MACHINE_DEBUG_PREFIX to be a string that is used as a prefix for all debug messages. If SM_ENTRY_MA macro is used to define a group of state machines with shared data structure, STATE_MACHINE_ADDR needs to be defined to point to the MAC address used in debug output. SM_ENTRY_M macro can be used to define similar group of state machines without this additional debug info.

Definition in file state_machine.h.

## 6.151.2 Define Documentation

### 6.151.2.1 #define SM_ENTER(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 0)

Enter a new state machine state.

**Parameters:**

***machine*** State machine name

***state*** State machine state

This macro expands to a function call to a state machine function defined with SM_STATE macro. SM_ENTER is used in a state machine step function to move the state machine to a new state.

Definition at line 113 of file state_machine.h.

### 6.151.2.2 #define SM_ENTER_GLOBAL(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 1)

Enter a new state machine state based on global rule.

**Parameters:**

***machine*** State machine name

***state*** State machine state

This macro is like SM_ENTER, but this is used when entering a new state based on a global (not specific to any particular state) rule. A separate macro is used to avoid unwanted debug message floods when the same global rule is forcing a state machine to remain in on state.

Definition at line 127 of file state_machine.h.

### 6.151.2.3 #define SM_ENTRY(machine, state)

**Value:**

```
if (!global || sm->machine ## _state != machine ## _ ## state) { \
        sm->changed = TRUE; \
        wpa_printf(MSG_DEBUG, STATE_MACHINE_DEBUG_PREFIX ": " #machine \
                   " entering state " #state); \
} \
sm->machine ## _state = machine ## _ ## state;
```

State machine function entry point.

**Parameters:**

***machine*** State machine name

***state*** State machine state

This macro is used inside each state machine function declared with SM_STATE. SM_ENTRY should be in the beginning of the function body, but after declaration of possible local variables. This macro prints debug information about state transition and update the state machine state.

Definition at line 55 of file state_machine.h.

#### 6.151.2.4 #define SM_ENTRY_M(machine, _state, data)

**Value:**

```
if (!global || sm->data ## _ ## state != machine ## _ ## _state) { \
        sm->changed = TRUE; \
        wpa_printf(MSG_DEBUG, STATE_MACHINE_DEBUG_PREFIX ": " \
                   #machine " entering state " #_state); \
} \
sm->data ## _ ## state = machine ## _ ## _state;
```

State machine function entry point for state machine group.

**Parameters:**

***machine*** State machine name

***_state*** State machine state

***data*** State variable prefix (full variable: <prefix>_state)

This macro is like SM_ENTRY, but for state machine groups that use a shared data structure for more than one state machine. Both machine and prefix parameters are set to "sub-state machine" name. prefix is used to allow more than one state variable to be stored in the same data structure.

Definition at line 75 of file state_machine.h.

#### 6.151.2.5 #define SM_ENTRY_MA(machine, _state, data)

**Value:**

```
if (!global || sm->data ## _ ## state != machine ## _ ## _state) { \
        sm->changed = TRUE; \
        wpa_printf(MSG_DEBUG, STATE_MACHINE_DEBUG_PREFIX ": " MACSTR " " \
                   #machine " entering state " #_state, \
                   MAC2STR(STATE_MACHINE_ADDR)); \
} \
sm->data ## _ ## state = machine ## _ ## _state;
```

State machine function entry point for state machine group.

**Parameters:**

***machine*** State machine name

***_state*** State machine state

***data*** State variable prefix (full variable: <prefix>_state)

This macro is like SM_ENTRY_M, but a MAC address is included in debug output. STATE_MACHINE_-ADDR has to be defined to point to the MAC address to be included in debug.

Definition at line 94 of file state_machine.h.

#### 6.151.2.6 #define SM_STATE(machine, state)

**Value:**

```
static void sm_ ## machine ## _ ## state ## _Enter(STATE_MACHINE_DATA *sm, \
        int global)
```

Declaration of a state machine function.

**Parameters:**

***machine*** State machine name

***state*** State machine state

This macro is used to declare a state machine function. It is used in place of a C function definition to declare functions to be run when the state is entered by calling SM_ENTER or SM_ENTER_GLOBAL.

Definition at line 40 of file state_machine.h.

#### 6.151.2.7 #define SM_STEP(machine) static void sm_ ## machine ## _Step(STATE_MACHINE_DATA *sm)

Declaration of a state machine step function.

**Parameters:**

***machine*** State machine name

This macro is used to declare a state machine step function. It is used in place of a C function definition to declare a function that is used to move state machine to a new state based on state variables. This function uses SM_ENTER and SM_ENTER_GLOBAL macros to enter new state.

Definition at line 140 of file state_machine.h.

#### 6.151.2.8 #define SM_STEP_RUN(machine) sm_ ## machine ## _Step(sm)

Call the state machine step function.

**Parameters:**

***machine*** State machine name

This macro expands to a function call to a state machine step function defined with SM_STEP macro.

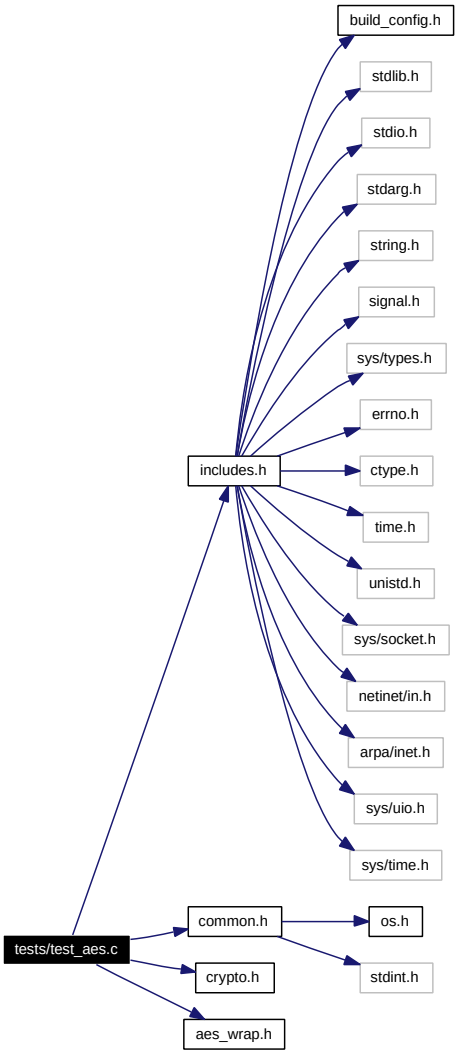
Definition at line 151 of file state_machine.h.

## 6.152 tests/test_aes.c File Reference

Test program for AES.

```
#include "includes.h"
#include "common.h"
#include "crypto.h"
#include "aes_wrap.h"
```

Include dependency graph for test_aes.c:




## Defines

- #define **BLOCK_SIZE** 16

## Functions

- int **main** (int argc, char ∗argv[ ])

### 6.152.1 Detailed Description

Test program for AES.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

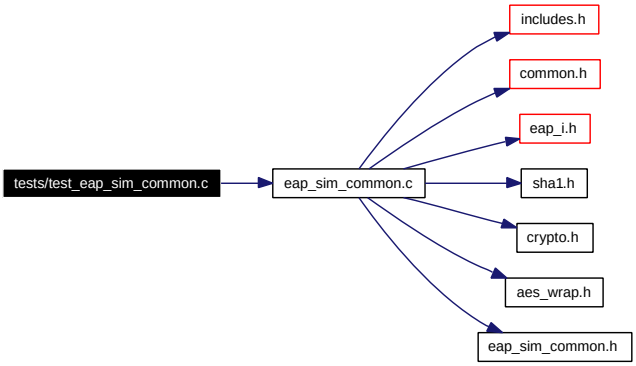
Definition in file test_aes.c.

## 6.153 tests/test_eap_sim_common.c File Reference

Test program for EAP-SIM PRF.

```
#include "eap_sim_common.c"
```

Include dependency graph for test_eap_sim_common.c:




### Functions

- int **main** (int argc, char *argv[ ])

### 6.153.1 Detailed Description

Test program for EAP-SIM PRF.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

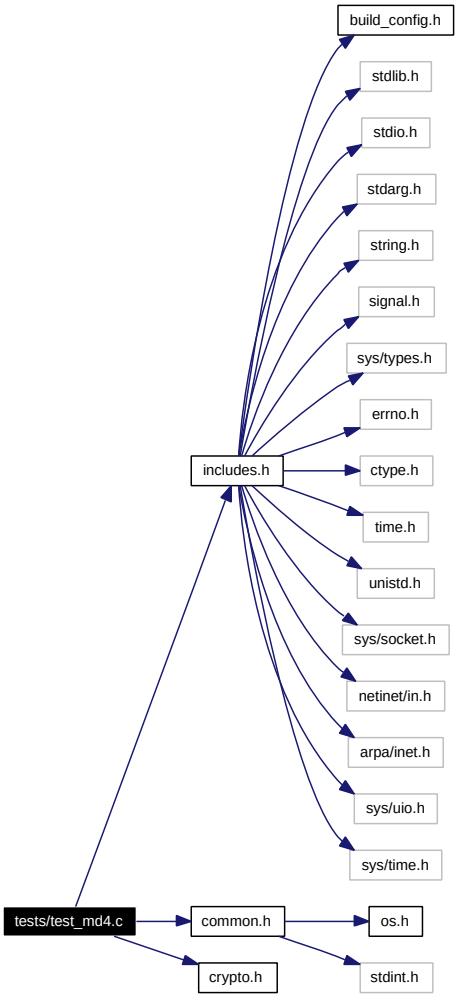
Definition in file test_eap_sim_common.c.

## 6.154 tests/test_md4.c File Reference

Test program for MD4 (test vectors from RFC 1320).

```
#include "includes.h"
#include "common.h"
#include "crypto.h"
```

Include dependency graph for test_md4.c:




### Functions

- int **main** (int argc, char ∗argv[ ])

### 6.154.1 Detailed Description

Test program for MD4 (test vectors from RFC 1320).

**Copyright**



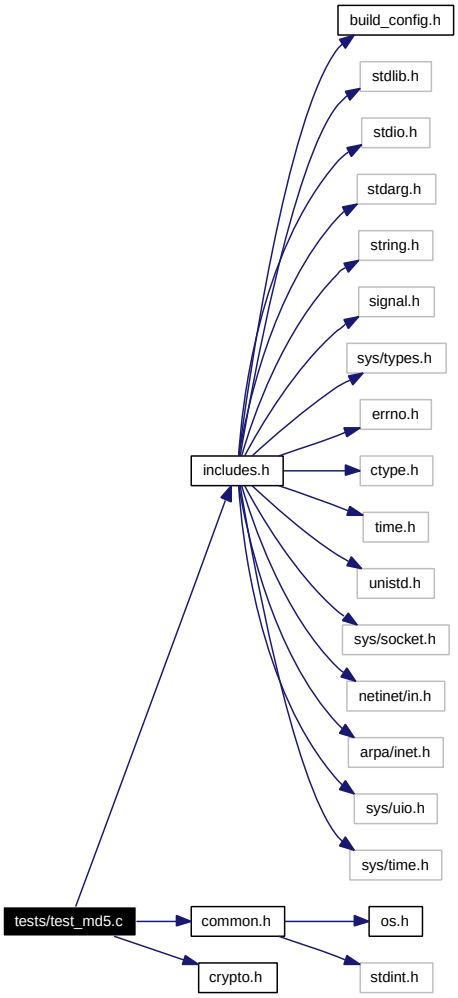
Definition in file test_md4.c.

## 6.155 tests/test_md5.c File Reference

Test program for MD5 (test vectors from RFC 1321).

```
#include "includes.h"
#include "common.h"
#include "crypto.h"
```

Include dependency graph for test_md5.c:




### Functions

- int **main** (int argc, char ∗argv[ ])

### 6.155.1 Detailed Description

Test program for MD5 (test vectors from RFC 1321).

**Copyright**



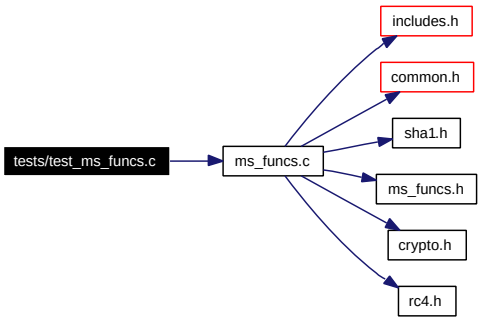
Definition in file test_md5.c.

# 6.156 tests/test_ms_funcs.c File Reference

Test program for ms_funcs.

```
#include "ms_funcs.c"
```

Include dependency graph for test_ms_funcs.c:




## Functions

- int **main** (int argc, char *argv[ ])

### 6.156.1 Detailed Description

Test program for ms_funcs.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

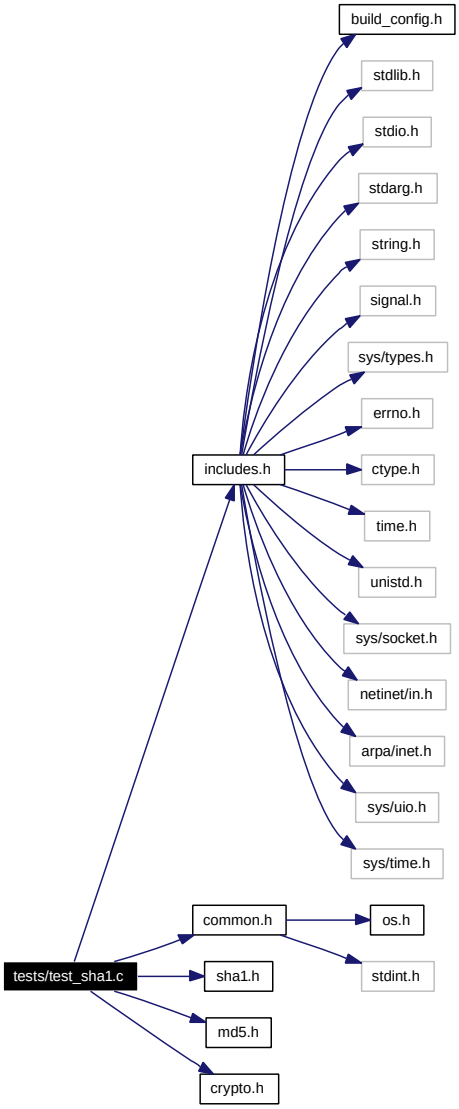
Definition in file test_ms_funcs.c.

## 6.157 tests/test_sha1.c File Reference

Test program for SHA1 and MD5.

```
#include "includes.h"
#include "common.h"
#include "sha1.h"
#include "md5.h"
#include "crypto.h"
```

Include dependency graph for test_sha1.c:




### Defines

- #define **NUM_PASSPHRASE_TESTS** (sizeof(passphrase_tests) / sizeof(passphrase_tests[0]))

## Functions

- int **main** (int argc, char ∗argv[ ])

### 6.157.1 Detailed Description

Test program for SHA1 and MD5.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

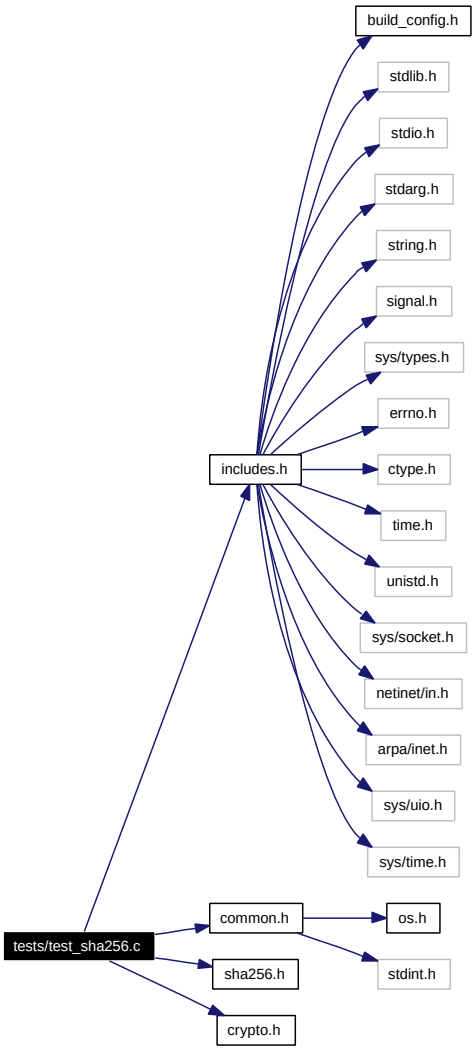
Definition in file test_sha1.c.

## 6.158 tests/test_sha256.c File Reference

Test program for SHA256.

```
#include "includes.h"
#include "common.h"
#include "sha256.h"
#include "crypto.h"
```

Include dependency graph for test_sha256.c:




## Functions

- int **main** (int argc, char *argv[ ])

## Variables

- struct {
  char * **data**
  u8 **hash** [32]
  } **tests** [ ]

- hmac_test **hmac_tests** [ ]

### 6.158.1 Detailed Description

Test program for SHA256.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

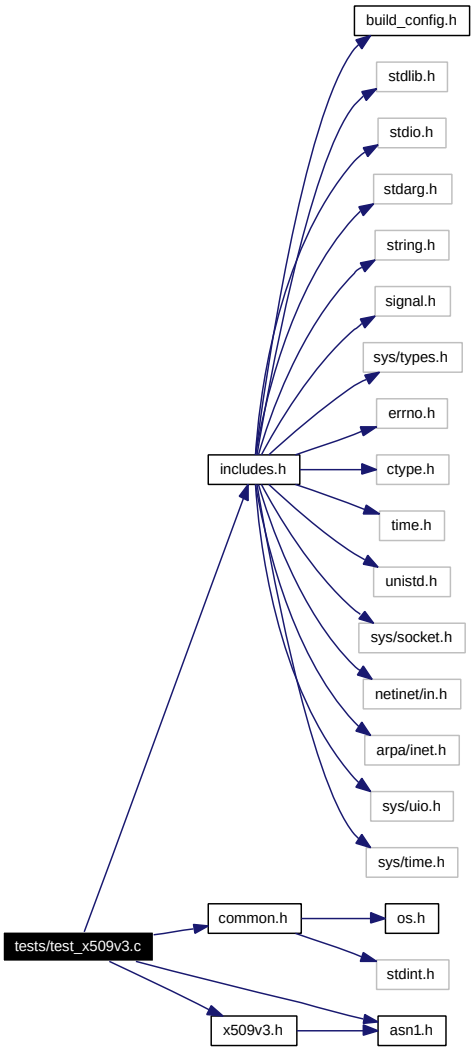
Definition in file test_sha256.c.

## 6.159 tests/test_x509v3.c File Reference

Testing tool for X.509v3 routines.

```
#include "includes.h"
#include "common.h"
#include "asn1.h"
#include "x509v3.h"
```

Include dependency graph for test_x509v3.c:




## Functions

- int **main** (int argc, char *argv[ ])

## Variables

- int **wpa_debug_level**

### 6.159.1 Detailed Description

Testing tool for X.509v3 routines.

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

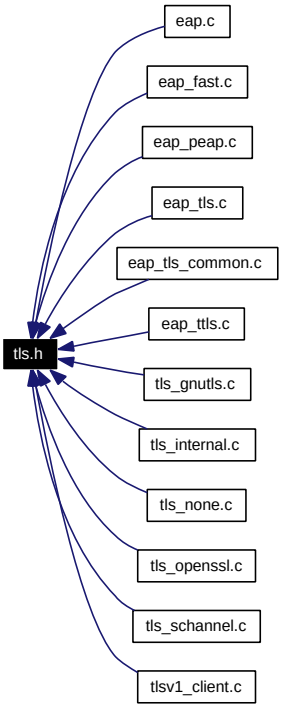
See README and COPYING for more details.

Definition in file test_x509v3.c.

## 6.160 tls.h File Reference

WPA Supplicant / SSL/TLS interface definition.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **TLS_CAPABILITY_IA** 0x0001

### Enumerations

- enum { **TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED** = -3, **TLS_SET_PARAMS_-ENGINE_PRV_INIT_FAILED** = -2 }
- enum {

  **TLS_CIPHER_NONE**, **TLS_CIPHER_RC4_SHA**, **TLS_CIPHER_AES128_SHA**, **TLS_-CIPHER_RSA_DHE_AES128_SHA**,

  **TLS_CIPHER_ANON_DH_AES128_SHA** }

### Functions

- void * tls_init (const struct tls_config *conf)

  *Initialize TLS library.*

- void tls_deinit (void *tls_ctx)

  *Deinitialize TLS library.*

- int tls_get_errors (void *tls_ctx)

  *Process pending errors.*

- tls_connection * tls_connection_init (void *tls_ctx)

  *Initialize a new TLS connection.*

- void tls_connection_deinit (void *tls_ctx, struct tls_connection *conn)

  *Free TLS connection data.*

- int tls_connection_established (void *tls_ctx, struct tls_connection *conn)

  *Has the TLS connection been completed?*

- int tls_connection_shutdown (void *tls_ctx, struct tls_connection *conn)

  *Shutdown TLS connection.*

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  *Set TLS connection parameters.*

- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)

  *Set TLS parameters for all TLS connection.*

- int tls_global_set_verify (void *tls_ctx, int check_crl)

  *Set global certificate verification options.*

- int tls_connection_set_verify (void *tls_ctx, struct tls_connection *conn, int verify_peer)

  *Set certificate verification options.*

- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

  *Set TLS/IA parameters.*

- int tls_connection_get_keys (void *tls_ctx, struct tls_connection *conn, struct tls_keys *keys)

  *Get master key and random data from TLS connection.*

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  *Use TLS-PRF to derive keying material.*

- u8 * tls_connection_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  *Process TLS handshake (client side).*

- u8 * tls_connection_server_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  *Process TLS handshake (server side).*

- int tls_connection_encrypt (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Encrypt data into TLS tunnel.*

- int tls_connection_decrypt (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Decrypt data from TLS tunnel.*

- int tls_connection_resumed (void *tls_ctx, struct tls_connection *conn)

  *Was session resumption used.*

- int tls_connection_set_master_key (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Configure master secret for TLS connection.*

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  *Configure acceptable cipher suites.*

- int tls_get_cipher (void *tls_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  *Get current cipher name.*

- int tls_connection_enable_workaround (void *tls_ctx, struct tls_connection *conn)

  *Enable TLS workaround options.*

- int tls_connection_client_hello_ext (void *tls_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)

  *Set TLS extension for ClientHello.*

- int tls_connection_get_failed (void *tls_ctx, struct tls_connection *conn)

  *Get connection failure status.*

- int tls_connection_get_read_alerts (void *tls_ctx, struct tls_connection *conn)

  *Get connection read alert status.*

- int tls_connection_get_write_alerts (void *tls_ctx, struct tls_connection *conn)

  *Get connection write alert status.*

- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)

  *Get TLS key_block size.*

- unsigned int tls_capabilities (void *tls_ctx)

  *Get supported TLS capabilities.*

- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  *Send a TLS/IA PhaseFinished message.*

- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  *Has final phase been completed.*

- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Permute TLS/IA inner secret.*

## 6.160.1 Detailed Description

WPA Supplicant / SSL/TLS interface definition.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tls.h.

## 6.160.2 Function Documentation

### 6.160.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

**Returns:**
Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

Definition at line 1231 of file tls_gnutls.c.

### 6.160.2.2 int tls_connection_client_hello_ext (void * *tls_ctx*, struct tls_connection * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()
***ext_type*** Extension type
***data*** Extension payload (NULL to remove extension)
***data_len*** Extension payload length

**Returns:**
0 on success, -1 on failure

Definition at line 1190 of file tls_gnutls.c.

### 6.160.2.3 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to input buffer (encrypted TLS data)

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (decrypted data from TLS tunnel)

***out_len*** Maximum out_data length

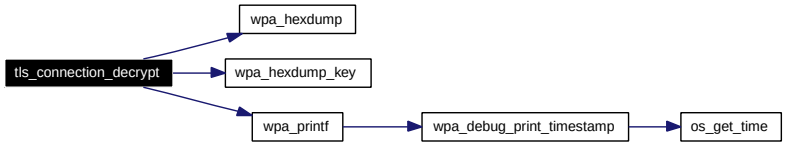
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

Definition at line 1067 of file tls_gnutls.c.

Here is the call graph for this function:




#### 6.160.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

Definition at line 361 of file tls_gnutls.c.

#### 6.160.2.5 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

Definition at line 1182 of file tls_gnutls.c.

#### 6.160.2.6 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to plaintext data to be encrypted

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (encrypted TLS data)

***out_len*** Maximum out_data length

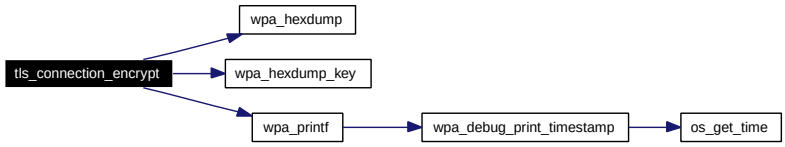
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

Definition at line 1038 of file tls_gnutls.c.

Here is the call graph for this function:




#### 6.160.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if TLS connection has been completed, 0 if not.

Definition at line 388 of file tls_gnutls.c.

#### 6.160.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

Definition at line 1199 of file tls_gnutls.c.

#### 6.160.2.9 int tls_connection_get_keyblock_size (void * *tls_ctx*, struct tls_connection * *conn*)

Get TLS key_block size.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()

**Returns:**
Size of the key_block for the negotiated cipher suite or -1 on failure

Definition at line 1223 of file tls_gnutls.c.

#### 6.160.2.10 int tls_connection_get_keys (void * *tls_ctx*, struct tls_connection * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()
***keys*** Structure of key/random data (filled on success)

**Returns:**
0 on success, -1 on failure

Definition at line 790 of file tls_gnutls.c.

#### 6.160.2.11 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()

**Returns:**
Number of times a fatal read (remote end reported error) has happened during this connection.

Definition at line 1207 of file tls_gnutls.c.

#### 6.160.2.12 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()

**Returns:**
Number of times a fatal write (locally detected error) has happened during this connection.

Definition at line 1215 of file tls_gnutls.c.

#### 6.160.2.13 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

***appl_data*** Pointer to application data pointer, or NULL if dropped

***appl_data_len*** Pointer to variable that is set to appl_data length

**Returns:**

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

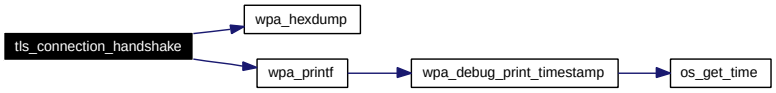
This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

Definition at line 930 of file tls_gnutls.c.

Here is the call graph for this function:




#### 6.160.2.14 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

Definition at line 1328 of file tls_gnutls.c.

#### 6.160.2.15 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***key*** Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

***key_len*** Length of session key material

**Returns:**

0 on success, -1 on failure

Definition at line 1338 of file tls_gnutls.c.

#### 6.160.2.16 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***final*** 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

***out_data*** Pointer to output buffer (encrypted TLS/IA data)

***out_len*** Maximum out_data length

**Returns:**

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

Definition at line 1280 of file tls_gnutls.c.

#### 6.160.2.17 struct tls_connection* tls_connection_init (void * *tls_ctx*)

Initialize a new TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

**Returns:**

Connection context data, conn for other function calls

Definition at line 325 of file tls_gnutls.c.

Here is the call graph for this function:

#### 6.160.2.18 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***label*** Label (e.g., description of the key) for PRF

***server_random_first*** seed is 0 = client_random|server_random, 1 = server_random|client_random

***out*** Buffer for output data from TLS-PRF

***out_len*** Length of the output buffer

**Returns:**

0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

Definition at line 827 of file tls_gnutls.c.

#### 6.160.2.19 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if current session used session resumption, 0 if not

Definition at line 1149 of file tls_gnutls.c.

#### 6.160.2.20 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

**Returns:**
pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

Definition at line 1028 of file tls_gnutls.c.

#### 6.160.2.21 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()
***ciphers*** Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

**Returns:**
0 on success, -1 on failure

Definition at line 1165 of file tls_gnutls.c.

#### 6.160.2.22 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()
***tls_ia*** 1 = enable TLS/IA

**Returns:**
0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

Definition at line 1243 of file tls_gnutls.c.

#### 6.160.2.23 int tls_connection_set_master_key (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Configure master secret for TLS connection.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()
***key*** TLS pre-master-secret
***key_len*** length of key in bytes

**Returns:**
0 on success, -1 on failure

Definition at line 1157 of file tls_gnutls.c.

#### 6.160.2.24 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***params*** Connection parameters

**Returns:**

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 548 of file tls_gnutls.c.

Here is the call graph for this function:


#### 6.160.2.25 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***verify_peer*** 1 = verify peer certificate

**Returns:**

0 on success, -1 on failure

Definition at line 775 of file tls_gnutls.c.

#### 6.160.2.26 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

Definition at line 394 of file tls_gnutls.c.

#### 6.160.2.27 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

Definition at line 215 of file tls_gnutls.c.

#### 6.160.2.28 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()
***buf*** Buffer for the cipher name
***buflen*** buf size

**Returns:**
0 on success, -1 on failure

Get the name of the currently used cipher.

Definition at line 1173 of file tls_gnutls.c.

#### 6.160.2.29 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

**Returns:**
Number of found error, 0 if no errors detected.

Process all pending TLS errors.

Definition at line 231 of file tls_gnutls.c.

#### 6.160.2.30 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***params*** Global TLS parameters

**Returns:**
0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 674 of file tls_gnutls.c.

#### 6.160.2.31 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***check_crl*** 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

**Returns:**
0 on success, -1 on failure

Definition at line 768 of file tls_gnutls.c.

#### 6.160.2.32 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

**Parameters:**
***conf*** Configuration data for TLS library

**Returns:**
Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

Definition at line 163 of file tls_gnutls.c.

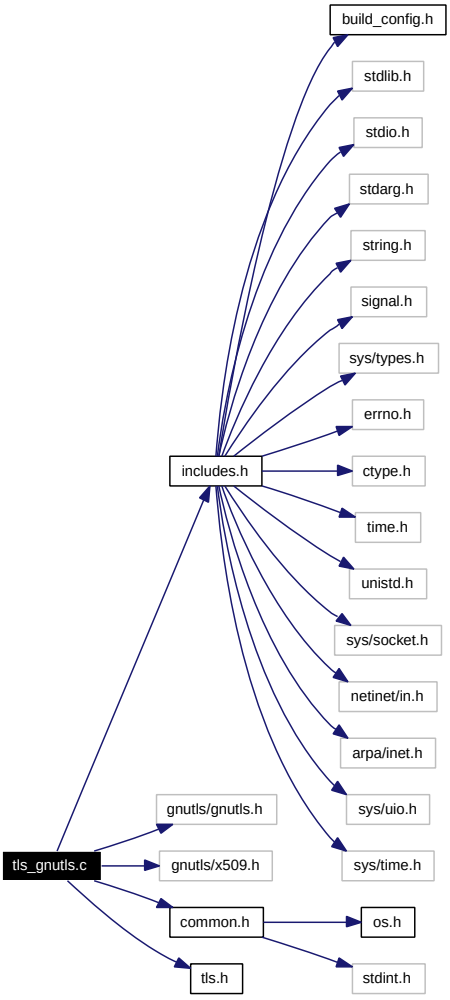
Here is the call graph for this function:

## 6.161 tls_gnutls.c File Reference

WPA Supplicant / SSL/TLS interface functions for openssl.

```
#include "includes.h"
#include <gnutls/gnutls.h>
#include <gnutls/x509.h>
#include "common.h"
#include "tls.h"
```

Include dependency graph for tls_gnutls.c:




### Defines

- #define **TLS_RANDOM_SIZE** 32
- #define **TLS_MASTER_SIZE** 48
- #define **GNUTLS_INTERNAL_STRUCTURE_HACK**

## Typedefs

- typedef u8 **uint8**
- typedef unsigned char **opaque**

## Functions

- void * tls_init (const struct tls_config *conf)

  *Initialize TLS library.*

- void tls_deinit (void *ssl_ctx)

  *Deinitialize TLS library.*

- int tls_get_errors (void *ssl_ctx)

  *Process pending errors.*

- tls_connection * tls_connection_init (void *ssl_ctx)

  *Initialize a new TLS connection.*

- void tls_connection_deinit (void *ssl_ctx, struct tls_connection *conn)

  *Free TLS connection data.*

- int tls_connection_established (void *ssl_ctx, struct tls_connection *conn)

  *Has the TLS connection been completed?*

- int tls_connection_shutdown (void *ssl_ctx, struct tls_connection *conn)

  *Shutdown TLS connection.*

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  *Set TLS connection parameters.*

- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)

  *Set TLS parameters for all TLS connection.*

- int tls_global_set_verify (void *ssl_ctx, int check_crl)

  *Set global certificate verification options.*

- int tls_connection_set_verify (void *ssl_ctx, struct tls_connection *conn, int verify_peer)

  *Set certificate verification options.*

- int tls_connection_get_keys (void *ssl_ctx, struct tls_connection *conn, struct tls_keys *keys)

  *Get master key and random data from TLS connection.*

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  *Use TLS-PRF to derive keying material.*

- u8 * tls_connection_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

*Process TLS handshake (client side).*

- u8 * tls_connection_server_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  *Process TLS handshake (server side).*

- int tls_connection_encrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Encrypt data into TLS tunnel.*

- int tls_connection_decrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Decrypt data from TLS tunnel.*

- int tls_connection_resumed (void *ssl_ctx, struct tls_connection *conn)

  *Was session resumption used.*

- int tls_connection_set_master_key (void *ssl_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Configure master secret for TLS connection.*

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  *Configure acceptable cipher suites.*

- int tls_get_cipher (void *ssl_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  *Get current cipher name.*

- int tls_connection_enable_workaround (void *ssl_ctx, struct tls_connection *conn)

  *Enable TLS workaround options.*

- int tls_connection_client_hello_ext (void *ssl_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)

  *Set TLS extension for ClientHello.*

- int tls_connection_get_failed (void *ssl_ctx, struct tls_connection *conn)

  *Get connection failure status.*

- int tls_connection_get_read_alerts (void *ssl_ctx, struct tls_connection *conn)

  *Get connection read alert status.*

- int tls_connection_get_write_alerts (void *ssl_ctx, struct tls_connection *conn)

  *Get connection write alert status.*

- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)

  *Get TLS key_block size.*

- unsigned int tls_capabilities (void *tls_ctx)

  *Get supported TLS capabilities.*

- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

*Set TLS/IA parameters.*

- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  *Send a TLS/IA PhaseFinished message.*

- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  *Has final phase been completed.*

- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Permute TLS/IA inner secret.*

## Variables

- int **wpa_debug_show_keys**

### 6.161.1 Detailed Description

WPA Supplicant / SSL/TLS interface functions for openssl.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tls_gnutls.c.

### 6.161.2 Function Documentation

#### 6.161.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

**Returns:**
Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

Definition at line 1231 of file tls_gnutls.c.

#### 6.161.2.2 int tls_connection_client_hello_ext (void * *tls_ctx*, struct tls_connection * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***ext_type*** Extension type

***data*** Extension payload (NULL to remove extension)

***data_len*** Extension payload length

**Returns:**

0 on success, -1 on failure

Definition at line 1190 of file tls_gnutls.c.

#### 6.161.2.3 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to input buffer (encrypted TLS data)

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (decrypted data from TLS tunnel)

***out_len*** Maximum out_data length

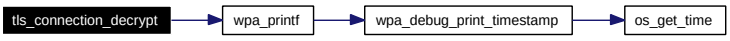
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

Definition at line 1067 of file tls_gnutls.c.

Here is the call graph for this function:




#### 6.161.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

Definition at line 361 of file tls_gnutls.c.

#### 6.161.2.5 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

Definition at line 1182 of file tls_gnutls.c.

#### 6.161.2.6 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to plaintext data to be encrypted

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (encrypted TLS data)

***out_len*** Maximum out_data length

**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

Definition at line 1038 of file tls_gnutls.c.

Here is the call graph for this function:

#### 6.161.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()

**Returns:**
1 if TLS connection has been completed, 0 if not.

Definition at line 388 of file tls_gnutls.c.

#### 6.161.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

Definition at line 1199 of file tls_gnutls.c.

#### 6.161.2.9 int tls_connection_get_keyblock_size (void * *tls_ctx*, struct tls_connection * *conn*)

Get TLS key_block size.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()

**Returns:**
Size of the key_block for the negotiated cipher suite or -1 on failure

Definition at line 1223 of file tls_gnutls.c.

#### 6.161.2.10 int tls_connection_get_keys (void * *tls_ctx*, struct tls_connection * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()
***conn*** Connection context data from tls_connection_init()
***keys*** Structure of key/random data (filled on success)

**Returns:**
0 on success, -1 on failure

Definition at line 790 of file tls_gnutls.c.

#### 6.161.2.11 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Number of times a fatal read (remote end reported error) has happened during this connection.

Definition at line 1207 of file tls_gnutls.c.

#### 6.161.2.12 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Number of times a fatal write (locally detected error) has happened during this connection.

Definition at line 1215 of file tls_gnutls.c.

#### 6.161.2.13 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

***appl_data*** Pointer to application data pointer, or NULL if dropped

***appl_data_len*** Pointer to variable that is set to appl_data length

**Returns:**

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server

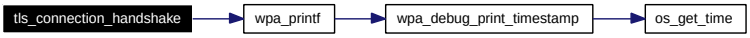
is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

Definition at line 930 of file tls_gnutls.c.

Here is the call graph for this function:




#### 6.161.2.14 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

Definition at line 1328 of file tls_gnutls.c.

#### 6.161.2.15 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***key*** Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

***key_len*** Length of session key material

**Returns:**

0 on success, -1 on failure

Definition at line 1338 of file tls_gnutls.c.

#### 6.161.2.16 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***final*** 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

***out_data*** Pointer to output buffer (encrypted TLS/IA data)

***out_len*** Maximum out_data length

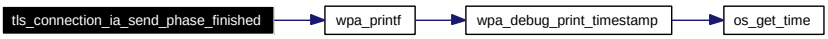
**Returns:**

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

Definition at line 1280 of file tls_gnutls.c.

Here is the call graph for this function:




### 6.161.2.17 struct tls_connection* tls_connection_init (void * *tls_ctx*)

Initialize a new TLS connection.

**Parameters:**

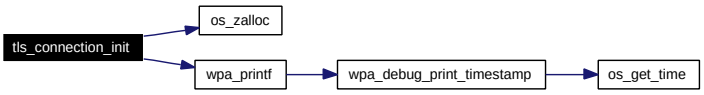
***tls_ctx*** TLS context data from tls_init()

**Returns:**

Connection context data, conn for other function calls

Definition at line 325 of file tls_gnutls.c.

Here is the call graph for this function:




### 6.161.2.18 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***label*** Label (e.g., description of the key) for PRF

***server_random_first*** seed is 0 = client_random|server_random, 1 = server_random|client_random

***out*** Buffer for output data from TLS-PRF

***out_len*** Length of the output buffer

**Returns:**
0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

Definition at line 827 of file tls_gnutls.c.

#### 6.161.2.19 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**
1 if current session used session resumption, 0 if not

Definition at line 1149 of file tls_gnutls.c.

#### 6.161.2.20 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

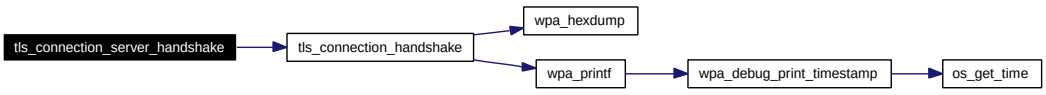
***out_len*** Length of the output buffer.

**Returns:**
pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

Definition at line 1028 of file tls_gnutls.c.

Here is the call graph for this function:



#### 6.161.2.21 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***ciphers*** Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

**Returns:**

0 on success, -1 on failure

Definition at line 1165 of file tls_gnutls.c.

#### 6.161.2.22 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***tls_ia*** 1 = enable TLS/IA

**Returns:**

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

Definition at line 1243 of file tls_gnutls.c.

Here is the call graph for this function:


#### 6.161.2.23 int tls_connection_set_master_key (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Configure master secret for TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***key*** TLS pre-master-secret

***key_len*** length of key in bytes

**Returns:**

0 on success, -1 on failure

Definition at line 1157 of file tls_gnutls.c.

#### 6.161.2.24 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***params*** Connection parameters

**Returns:**

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 548 of file tls_gnutls.c.

Here is the call graph for this function:


#### 6.161.2.25 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***verify_peer*** 1 = verify peer certificate

**Returns:**

0 on success, -1 on failure

Definition at line 775 of file tls_gnutls.c.

#### 6.161.2.26 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

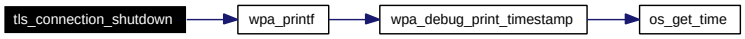
**Returns:**

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

Definition at line 394 of file tls_gnutls.c.

Here is the call graph for this function:




#### 6.161.2.27 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

Definition at line 215 of file tls_gnutls.c.

#### 6.161.2.28 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***buf*** Buffer for the cipher name

***buflen*** buf size

**Returns:**

0 on success, -1 on failure

Get the name of the currently used cipher.

Definition at line 1173 of file tls_gnutls.c.

#### 6.161.2.29 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

**Returns:**

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

Definition at line 231 of file tls_gnutls.c.

#### 6.161.2.30 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***params*** Global TLS parameters

**Returns:**

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 674 of file tls_gnutls.c.

Here is the call graph for this function:


#### 6.161.2.31 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***check_crl*** 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

**Returns:**

0 on success, -1 on failure

Definition at line 768 of file tls_gnutls.c.

#### 6.161.2.32 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

**Parameters:**

***conf*** Configuration data for TLS library

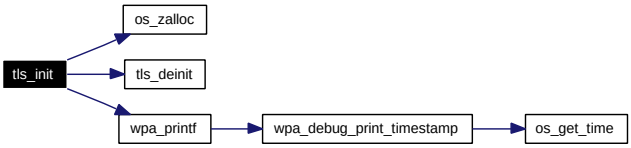
**Returns:**

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

Definition at line 163 of file tls_gnutls.c.

Here is the call graph for this function:

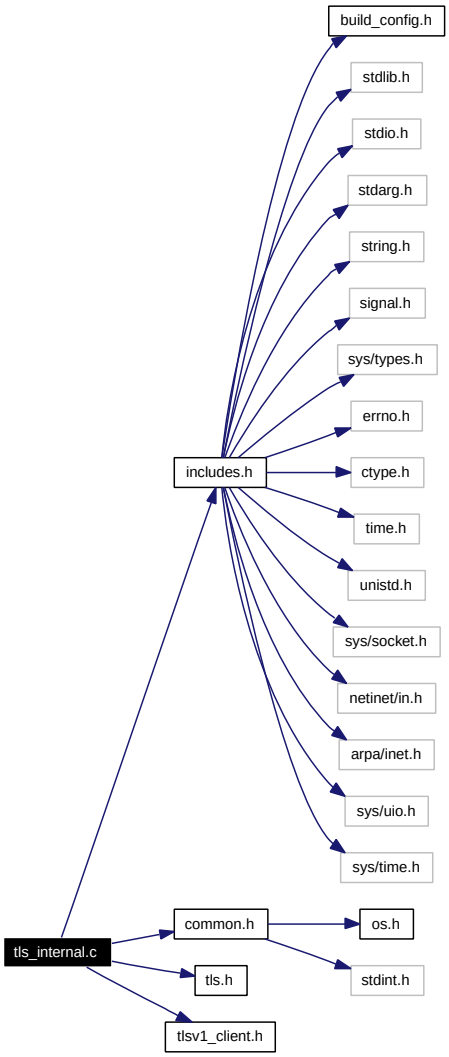


## 6.162 tls_internal.c File Reference

WPA Supplicant / TLS interface functions and an internal TLS implementation.

```
#include "includes.h"
#include "common.h"
#include "tls.h"
#include "tlsv1_client.h"
```

Include dependency graph for tls_internal.c:




### Data Structures

- struct **tls_global**
- struct **tls_connection**

## Functions

- void * tls_init (const struct tls_config *conf)

  *Initialize TLS library.*

- void tls_deinit (void *ssl_ctx)

  *Deinitialize TLS library.*

- int tls_get_errors (void *tls_ctx)

  *Process pending errors.*

- tls_connection * tls_connection_init (void *tls_ctx)

  *Initialize a new TLS connection.*

- void tls_connection_deinit (void *tls_ctx, struct tls_connection *conn)

  *Free TLS connection data.*

- int tls_connection_established (void *tls_ctx, struct tls_connection *conn)

  *Has the TLS connection been completed?*

- int tls_connection_shutdown (void *tls_ctx, struct tls_connection *conn)

  *Shutdown TLS connection.*

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  *Set TLS connection parameters.*

- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)

  *Set TLS parameters for all TLS connection.*

- int tls_global_set_verify (void *tls_ctx, int check_crl)

  *Set global certificate verification options.*

- int tls_connection_set_verify (void *tls_ctx, struct tls_connection *conn, int verify_peer)

  *Set certificate verification options.*

- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

  *Set TLS/IA parameters.*

- int tls_connection_get_keys (void *tls_ctx, struct tls_connection *conn, struct tls_keys *keys)

  *Get master key and random data from TLS connection.*

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  *Use TLS-PRF to derive keying material.*

- u8 * tls_connection_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  *Process TLS handshake (client side).*

- u8 * tls_connection_server_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  *Process TLS handshake (server side).*

- int tls_connection_encrypt (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Encrypt data into TLS tunnel.*

- int tls_connection_decrypt (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Decrypt data from TLS tunnel.*

- int tls_connection_resumed (void *tls_ctx, struct tls_connection *conn)

  *Was session resumption used.*

- int tls_connection_set_master_key (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Configure master secret for TLS connection.*

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  *Configure acceptable cipher suites.*

- int tls_get_cipher (void *tls_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  *Get current cipher name.*

- int tls_connection_enable_workaround (void *tls_ctx, struct tls_connection *conn)

  *Enable TLS workaround options.*

- int tls_connection_client_hello_ext (void *tls_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)

  *Set TLS extension for ClientHello.*

- int tls_connection_get_failed (void *tls_ctx, struct tls_connection *conn)

  *Get connection failure status.*

- int tls_connection_get_read_alerts (void *tls_ctx, struct tls_connection *conn)

  *Get connection read alert status.*

- int tls_connection_get_write_alerts (void *tls_ctx, struct tls_connection *conn)

  *Get connection write alert status.*

- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)

  *Get TLS key_block size.*

- unsigned int tls_capabilities (void *tls_ctx)

  *Get supported TLS capabilities.*

- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  *Send a TLS/IA PhaseFinished message.*

- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  *Has final phase been completed.*

- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Permute TLS/IA inner secret.*

## 6.162.1 Detailed Description

WPA Supplicant / TLS interface functions and an internal TLS implementation.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file interface functions for hostapd/wpa_supplicant to use the integrated TLSv1 implementation.

Definition in file tls_internal.c.

## 6.162.2 Function Documentation

### 6.162.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

**Returns:**
Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

Definition at line 300 of file tls_internal.c.

### 6.162.2.2 int tls_connection_client_hello_ext (void * *tls_ctx*, struct tls_connection * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***ext_type*** Extension type

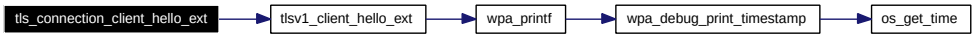
***data*** Extension payload (NULL to remove extension)

***data_len*** Extension payload length

**Returns:**
0 on success, -1 on failure

Definition at line 266 of file tls_internal.c.

Here is the call graph for this function:




#### 6.162.2.3 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to input buffer (encrypted TLS data)

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (decrypted data from TLS tunnel)

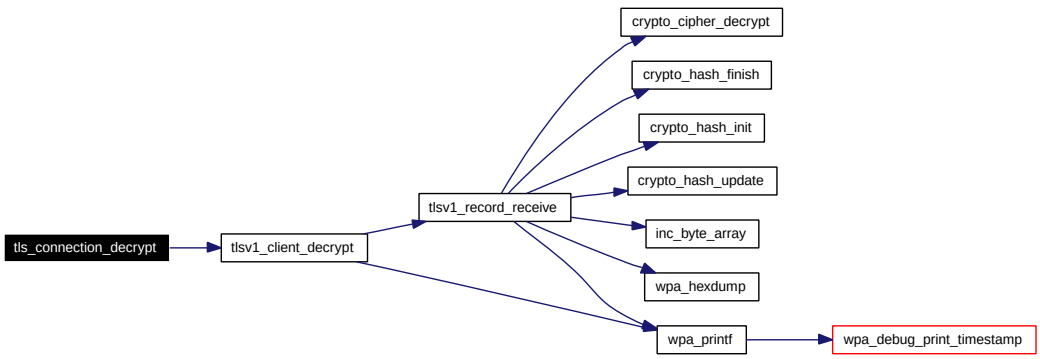
***out_len*** Maximum out_data length

**Returns:**
Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

Definition at line 221 of file tls_internal.c.

Here is the call graph for this function:



#### 6.162.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

**Parameters:**

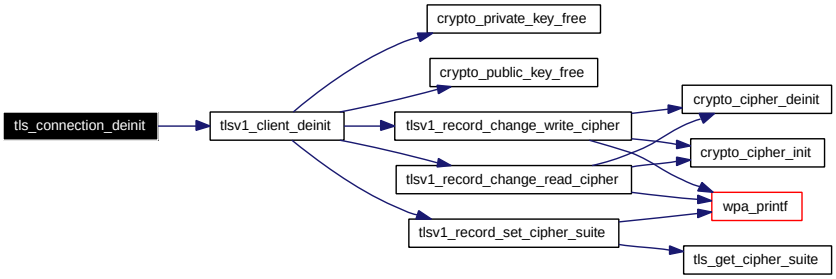
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

Definition at line 89 of file tls_internal.c.

Here is the call graph for this function:




#### 6.162.2.5 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

Definition at line 259 of file tls_internal.c.

#### 6.162.2.6 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to plaintext data to be encrypted

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (encrypted TLS data)

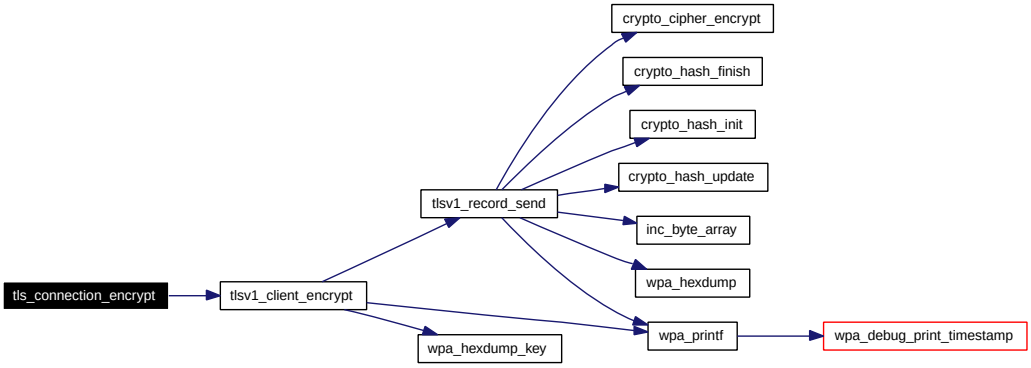
***out_len*** Maximum out_data length

**Returns:**
Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

Definition at line 212 of file tls_internal.c.

Here is the call graph for this function:




### 6.162.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**
1 if TLS connection has been completed, 0 if not.

Definition at line 98 of file tls_internal.c.

Here is the call graph for this function:


### 6.162.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

Definition at line 274 of file tls_internal.c.

#### 6.162.2.9 int tls_connection_get_keyblock_size (void * *tls_ctx*, struct tls_connection * *conn*)

Get TLS key_block size.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Size of the key_block for the negotiated cipher suite or -1 on failure

Definition at line 293 of file tls_internal.c.

Here is the call graph for this function:


#### 6.162.2.10 int tls_connection_get_keys (void * *tls_ctx*, struct tls_connection * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***keys*** Structure of key/random data (filled on success)

**Returns:**

0 on success, -1 on failure

Definition at line 172 of file tls_internal.c.

Here is the call graph for this function:


#### 6.162.2.11 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Number of times a fatal read (remote end reported error) has happened during this connection.

Definition at line 280 of file tls_internal.c.

#### 6.162.2.12 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Number of times a fatal write (locally detected error) has happened during this connection.

Definition at line 286 of file tls_internal.c.

#### 6.162.2.13 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

***appl_data*** Pointer to application data pointer, or NULL if dropped

***appl_data_len*** Pointer to variable that is set to appl_data length

**Returns:**

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

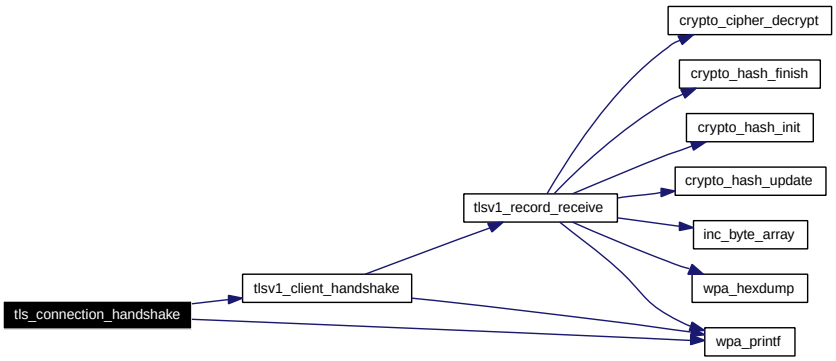
This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

Definition at line 188 of file tls_internal.c.

Here is the call graph for this function:




#### 6.162.2.14 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

Definition at line 315 of file tls_internal.c.

#### 6.162.2.15 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***key*** Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

***key_len*** Length of session key material

**Returns:**

0 on success, -1 on failure

Definition at line 322 of file tls_internal.c.

#### 6.162.2.16 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***final*** 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

***out_data*** Pointer to output buffer (encrypted TLS/IA data)

***out_len*** Maximum out_data length

**Returns:**

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

Definition at line 306 of file tls_internal.c.

#### 6.162.2.17 struct tls_connection* tls_connection_init (void * *tls_ctx*)

Initialize a new TLS connection.

**Parameters:**

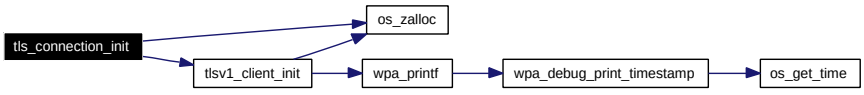
***tls_ctx*** TLS context data from tls_init()

**Returns:**

Connection context data, conn for other function calls

Definition at line 71 of file tls_internal.c.

Here is the call graph for this function:




#### 6.162.2.18 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***label*** Label (e.g., description of the key) for PRF

***server_random_first*** seed is 0 = client_random|server_random, 1 = server_random|client_random

***out*** Buffer for output data from TLS-PRF

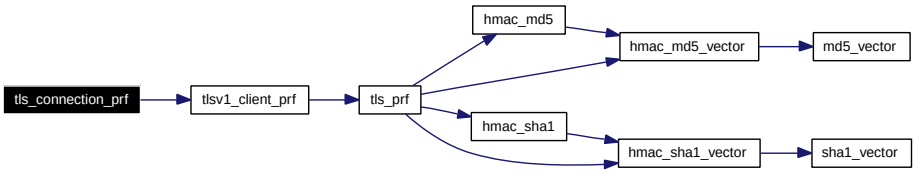
***out_len*** Length of the output buffer

**Returns:**
0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

Definition at line 179 of file tls_internal.c.

Here is the call graph for this function:




### 6.162.2.19 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**
1 if current session used session resumption, 0 if not

Definition at line 230 of file tls_internal.c.

Here is the call graph for this function:


### 6.162.2.20 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

**Returns:**
pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

Definition at line 202 of file tls_internal.c.

Here is the call graph for this function:


### 6.162.2.21 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***ciphers*** Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

**Returns:**
0 on success, -1 on failure

Definition at line 243 of file tls_internal.c.

Here is the call graph for this function:


### 6.162.2.22 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***tls_ia*** 1 = enable TLS/IA

**Returns:**
0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

Definition at line 165 of file tls_internal.c.

#### 6.162.2.23 int tls_connection_set_master_key (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Configure master secret for TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***key*** TLS pre-master-secret

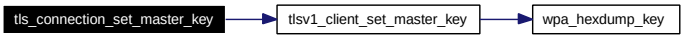
***key_len*** length of key in bytes

**Returns:**

0 on success, -1 on failure

Definition at line 236 of file tls_internal.c.

Here is the call graph for this function:




#### 6.162.2.24 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

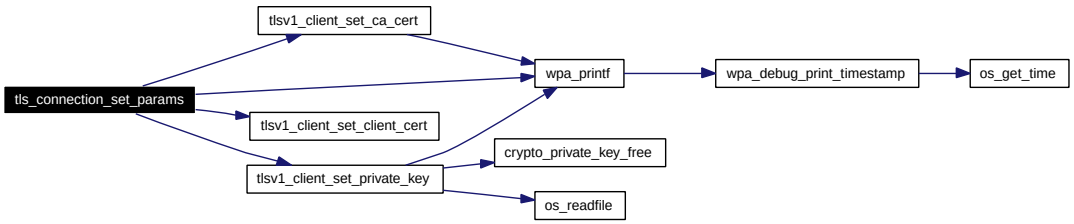
***params*** Connection parameters

**Returns:**

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 110 of file tls_internal.c.

Here is the call graph for this function:



#### 6.162.2.25 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***verify_peer*** 1 = verify peer certificate

**Returns:**

0 on success, -1 on failure

Definition at line 158 of file tls_internal.c.

#### 6.162.2.26 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

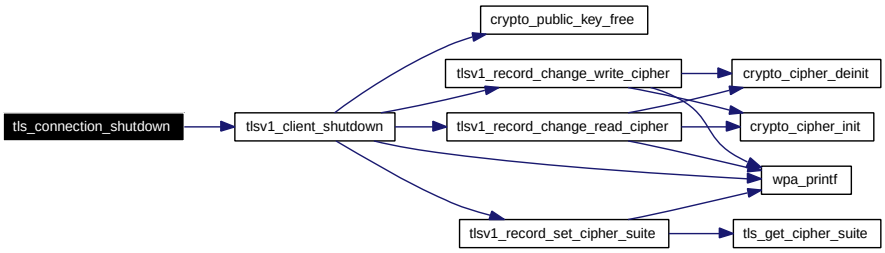
**Returns:**

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

Definition at line 104 of file tls_internal.c.

Here is the call graph for this function:




#### 6.162.2.27 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

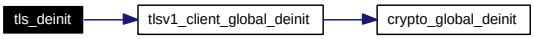
**Parameters:**

***tls_ctx*** TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

Definition at line 54 of file tls_internal.c.

Here is the call graph for this function:




#### 6.162.2.28 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***buf*** Buffer for the cipher name

***buflen*** buf size

**Returns:**

0 on success, -1 on failure

Get the name of the currently used cipher.

Definition at line 250 of file tls_internal.c.

Here is the call graph for this function:


#### 6.162.2.29 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

**Returns:**

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

Definition at line 65 of file tls_internal.c.

#### 6.162.2.30 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

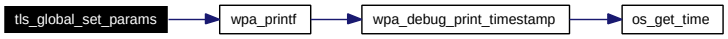
***params*** Global TLS parameters

**Returns:**

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 143 of file tls_internal.c.

Here is the call graph for this function:




#### 6.162.2.31 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***check_crl*** 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

**Returns:**

0 on success, -1 on failure

Definition at line 151 of file tls_internal.c.

Here is the call graph for this function:


#### 6.162.2.32 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

**Parameters:**

***conf*** Configuration data for TLS library

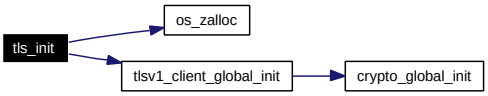
**Returns:**

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

Definition at line 37 of file tls_internal.c.

Here is the call graph for this function:

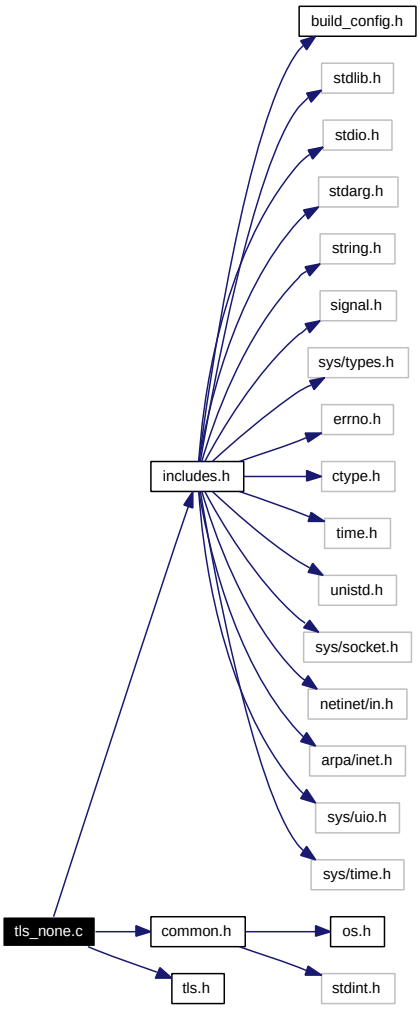


## 6.163 tls_none.c File Reference

WPA Supplicant / SSL/TLS interface functions for no TLS case.

```
#include "includes.h"
#include "common.h"
#include "tls.h"
```

Include dependency graph for tls_none.c:




## Functions

- void * tls_init (const struct tls_config *conf)

  *Initialize TLS library.*

- void tls_deinit (void *ssl_ctx)

  *Deinitialize TLS library.*

## 6.163.1 Detailed Description

WPA Supplicant / SSL/TLS interface functions for no TLS case.

**Copyright**
Copyright (c) 2004, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tls_none.c.

## 6.163.2 Function Documentation

### 6.163.2.1 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

Definition at line 26 of file tls_none.c.

### 6.163.2.2 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

**Parameters:**
***conf*** Configuration data for TLS library

**Returns:**
Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

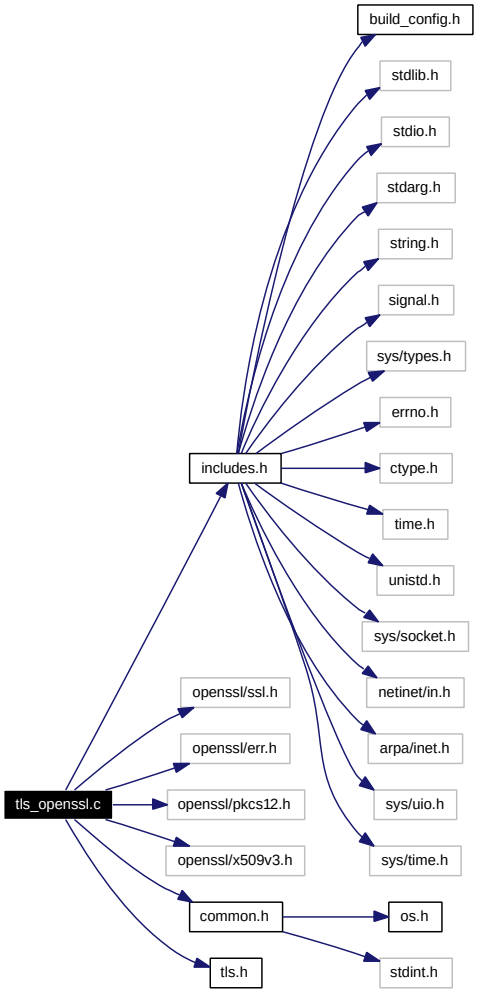
Definition at line 21 of file tls_none.c.

## 6.164 tls_openssl.c File Reference

WPA Supplicant / SSL/TLS interface functions for openssl.

```
#include "includes.h"
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <openssl/pkcs12.h>
#include <openssl/x509v3.h>
#include "common.h"
#include "tls.h"
```

Include dependency graph for tls_openssl.c:




## Data Structures

- struct **tls_connection**

## Defines

- #define **OPENSSL_d2i_TYPE** unsigned char **

## Functions

- void * tls_init (const struct tls_config *conf)

  *Initialize TLS library.*

- void tls_deinit (void *ssl_ctx)

  *Deinitialize TLS library.*

- int tls_get_errors (void *ssl_ctx)

  *Process pending errors.*

- tls_connection * tls_connection_init (void *ssl_ctx)

  *Initialize a new TLS connection.*

- void tls_connection_deinit (void *ssl_ctx, struct tls_connection *conn)

  *Free TLS connection data.*

- int tls_connection_established (void *ssl_ctx, struct tls_connection *conn)

  *Has the TLS connection been completed?*

- int tls_connection_shutdown (void *ssl_ctx, struct tls_connection *conn)

  *Shutdown TLS connection.*

- int tls_global_set_verify (void *ssl_ctx, int check_crl)

  *Set global certificate verification options.*

- int tls_connection_set_verify (void *ssl_ctx, struct tls_connection *conn, int verify_peer)

  *Set certificate verification options.*

- int tls_connection_get_keys (void *ssl_ctx, struct tls_connection *conn, struct tls_keys *keys)

  *Get master key and random data from TLS connection.*

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_-random_first, u8 *out, size_t out_len)

  *Use TLS-PRF to derive keying material.*

- u8 * tls_connection_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  *Process TLS handshake (client side).*

- u8 * tls_connection_server_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_-data, size_t in_len, size_t *out_len)

  *Process TLS handshake (server side).*

- int tls_connection_encrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_-len, u8 *out_data, size_t out_len)

*Encrypt data into TLS tunnel.*

- int tls_connection_decrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Decrypt data from TLS tunnel.*

- int tls_connection_resumed (void *ssl_ctx, struct tls_connection *conn)

  *Was session resumption used.*

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  *Configure acceptable cipher suites.*

- int tls_get_cipher (void *ssl_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  *Get current cipher name.*

- int tls_connection_enable_workaround (void *ssl_ctx, struct tls_connection *conn)

  *Enable TLS workaround options.*

- int tls_connection_get_failed (void *ssl_ctx, struct tls_connection *conn)

  *Get connection failure status.*

- int tls_connection_get_read_alerts (void *ssl_ctx, struct tls_connection *conn)

  *Get connection read alert status.*

- int tls_connection_get_write_alerts (void *ssl_ctx, struct tls_connection *conn)

  *Get connection write alert status.*

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  *Set TLS connection parameters.*

- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)

  *Set TLS parameters for all TLS connection.*

- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)

  *Get TLS key_block size.*

- unsigned int tls_capabilities (void *tls_ctx)

  *Get supported TLS capabilities.*

- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

  *Set TLS/IA parameters.*

- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  *Send a TLS/IA PhaseFinished message.*

- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  *Has final phase been completed.*

- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Permute TLS/IA inner secret.*

## 6.164.1 Detailed Description

WPA Supplicant / SSL/TLS interface functions for openssl.

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tls_openssl.c.

## 6.164.2 Function Documentation

### 6.164.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

**Returns:**
Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

Definition at line 2278 of file tls_openssl.c.

### 6.164.2.2 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to input buffer (encrypted TLS data)

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (decrypted data from TLS tunnel)

***out_len*** Maximum out_data length

**Returns:**
Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

Definition at line 1961 of file tls_openssl.c.

#### 6.164.2.3 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

Definition at line 920 of file tls_openssl.c.

#### 6.164.2.4 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

Definition at line 2121 of file tls_openssl.c.

#### 6.164.2.5 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to plaintext data to be encrypted

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (encrypted TLS data)

***out_len*** Maximum out_data length

**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

Definition at line 1927 of file tls_openssl.c.

#### 6.164.2.6 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if TLS connection has been completed, 0 if not.

Definition at line 933 of file tls_openssl.c.

#### 6.164.2.7 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

Definition at line 2150 of file tls_openssl.c.

#### 6.164.2.8 int tls_connection_get_keyblock_size (void * *tls_ctx*, struct tls_connection * *conn*)

Get TLS key_block size.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Size of the key_block for the negotiated cipher suite or -1 on failure

Definition at line 2257 of file tls_openssl.c.

#### 6.164.2.9 int tls_connection_get_keys (void * *tls_ctx*, struct tls_connection * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***keys*** Structure of key/random data (filled on success)

**Returns:**

0 on success, -1 on failure

Definition at line 1757 of file tls_openssl.c.

#### 6.164.2.10 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Number of times a fatal read (remote end reported error) has happened during this connection.

Definition at line 2158 of file tls_openssl.c.

#### 6.164.2.11 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Number of times a fatal write (locally detected error) has happened during this connection.

Definition at line 2166 of file tls_openssl.c.

#### 6.164.2.12 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

***appl_data*** Pointer to application data pointer, or NULL if dropped

***appl_data_len*** Pointer to variable that is set to appl_data length

**Returns:**

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server

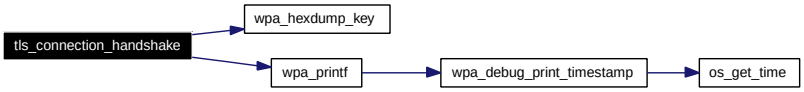
is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

Definition at line 1788 of file tls_openssl.c.

Here is the call graph for this function:




#### 6.164.2.13 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

Definition at line 2300 of file tls_openssl.c.

#### 6.164.2.14 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***key*** Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

***key_len*** Length of session key material

**Returns:**

0 on success, -1 on failure

Definition at line 2307 of file tls_openssl.c.

#### 6.164.2.15 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***final*** 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

***out_data*** Pointer to output buffer (encrypted TLS/IA data)

***out_len*** Maximum out_data length

**Returns:**

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

Definition at line 2291 of file tls_openssl.c.

#### 6.164.2.16 struct tls_connection* tls_connection_init (void * *tls_ctx*)

Initialize a new TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

**Returns:**

Connection context data, conn for other function calls

Definition at line 874 of file tls_openssl.c.

Here is the call graph for this function:


#### 6.164.2.17 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***label*** Label (e.g., description of the key) for PRF

***server_random_first*** seed is 0 = client_random|server_random, 1 = server_random|client_random

***out*** Buffer for output data from TLS-PRF

***out_len*** Length of the output buffer

**Returns:**
0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

Definition at line 1780 of file tls_openssl.c.

#### 6.164.2.18 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**
1 if current session used session resumption, 0 if not

Definition at line 1991 of file tls_openssl.c.

#### 6.164.2.19 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

**Returns:**
pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

Definition at line 1876 of file tls_openssl.c.

Here is the call graph for this function:

#### 6.164.2.20 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***ciphers*** Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

**Returns:**

0 on success, -1 on failure

Definition at line 2048 of file tls_openssl.c.

Here is the call graph for this function:


#### 6.164.2.21 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***tls_ia*** 1 = enable TLS/IA

**Returns:**

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

Definition at line 2284 of file tls_openssl.c.

#### 6.164.2.22 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

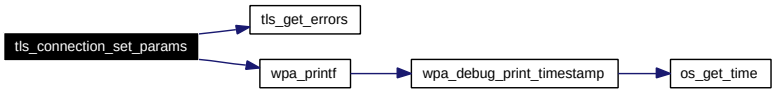
***params*** Connection parameters

**Returns:**

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 2174 of file tls_openssl.c.

Here is the call graph for this function:




#### 6.164.2.23 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***verify_peer*** 1 = verify peer certificate

**Returns:**

0 on success, -1 on failure

Definition at line 1238 of file tls_openssl.c.

#### 6.164.2.24 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

Definition at line 939 of file tls_openssl.c.

#### 6.164.2.25 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

Definition at line 761 of file tls_openssl.c.

#### 6.164.2.26 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***buf*** Buffer for the cipher name

***buflen*** buf size

**Returns:**

0 on success, -1 on failure

Get the name of the currently used cipher.

Definition at line 2104 of file tls_openssl.c.

#### 6.164.2.27 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

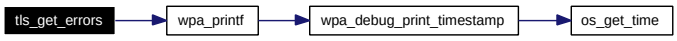
**Returns:**

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

Definition at line 860 of file tls_openssl.c.

Here is the call graph for this function:




#### 6.164.2.28 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

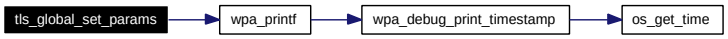
***params*** Global TLS parameters

**Returns:**

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 2232 of file tls_openssl.c.

Here is the call graph for this function:




#### 6.164.2.29 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***check_crl*** 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

**Returns:**

0 on success, -1 on failure

Definition at line 1193 of file tls_openssl.c.

#### 6.164.2.30 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

**Parameters:**

***conf*** Configuration data for TLS library

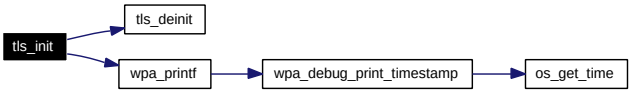
**Returns:**

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

Definition at line 717 of file tls_openssl.c.

Here is the call graph for this function:

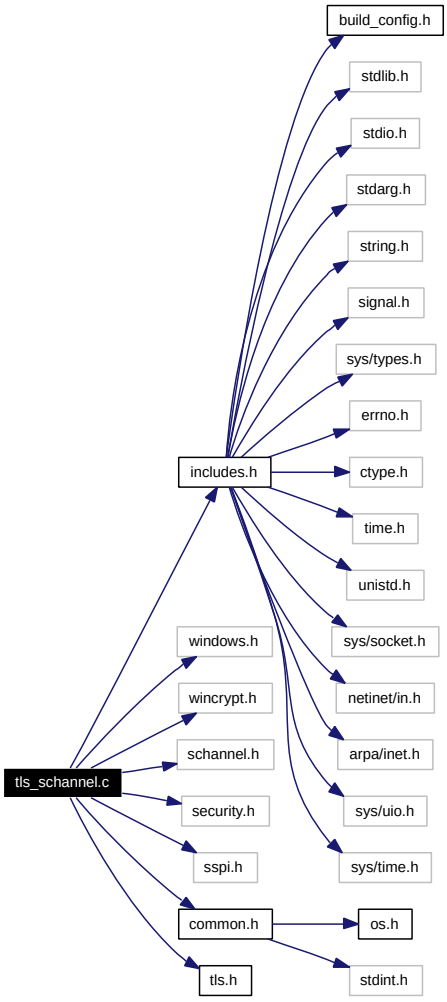


## 6.165 tls_schannel.c File Reference

WPA Supplicant / SSL/TLS interface functions for Microsoft Schannel.

```
#include "includes.h"
#include <windows.h>
#include <wincrypt.h>
#include <schannel.h>
#include <security.h>
#include <sspi.h>
#include "common.h"
#include "tls.h"
```

Include dependency graph for tls_schannel.c:



## Data Structures

- struct **tls_global**
- struct **tls_connection**

## Defines

- #define **SECURITY_WIN32**
- #define **SECPKG_ATTR_EAP_KEY_BLOCK** 0x5b

## Typedefs

- typedef _SecPkgContext_EapKeyBlock **SecPkgContext_EapKeyBlock**
- typedef _SecPkgContext_EapKeyBlock * **PSecPkgContext_EapKeyBlock**

## Functions

- void * tls_init (const struct tls_config *conf)
  *Initialize TLS library.*
- void tls_deinit (void *ssl_ctx)
  *Deinitialize TLS library.*
- int tls_get_errors (void *ssl_ctx)
  *Process pending errors.*
- tls_connection * tls_connection_init (void *ssl_ctx)
  *Initialize a new TLS connection.*
- void tls_connection_deinit (void *ssl_ctx, struct tls_connection *conn)
  *Free TLS connection data.*
- int tls_connection_established (void *ssl_ctx, struct tls_connection *conn)
  *Has the TLS connection been completed?*
- int tls_connection_shutdown (void *ssl_ctx, struct tls_connection *conn)
  *Shutdown TLS connection.*
- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)
  *Set TLS parameters for all TLS connection.*
- int tls_global_set_verify (void *ssl_ctx, int check_crl)
  *Set global certificate verification options.*
- int tls_connection_set_verify (void *ssl_ctx, struct tls_connection *conn, int verify_peer)
  *Set certificate verification options.*
- int tls_connection_get_keys (void *ssl_ctx, struct tls_connection *conn, struct tls_keys *keys)
  *Get master key and random data from TLS connection.*

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  *Use TLS-PRF to derive keying material.*

- u8 * tls_connection_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  *Process TLS handshake (client side).*

- u8 * tls_connection_server_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  *Process TLS handshake (server side).*

- int tls_connection_encrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Encrypt data into TLS tunnel.*

- int tls_connection_decrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Decrypt data from TLS tunnel.*

- int tls_connection_resumed (void *ssl_ctx, struct tls_connection *conn)

  *Was session resumption used.*

- int tls_connection_set_master_key (void *ssl_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Configure master secret for TLS connection.*

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  *Configure acceptable cipher suites.*

- int tls_get_cipher (void *ssl_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  *Get current cipher name.*

- int tls_connection_enable_workaround (void *ssl_ctx, struct tls_connection *conn)

  *Enable TLS workaround options.*

- int tls_connection_client_hello_ext (void *ssl_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)

  *Set TLS extension for ClientHello.*

- int tls_connection_get_failed (void *ssl_ctx, struct tls_connection *conn)

  *Get connection failure status.*

- int tls_connection_get_read_alerts (void *ssl_ctx, struct tls_connection *conn)

  *Get connection read alert status.*

- int tls_connection_get_write_alerts (void *ssl_ctx, struct tls_connection *conn)

  *Get connection write alert status.*

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  *Set TLS connection parameters.*

- unsigned int tls_capabilities (void *tls_ctx)

  *Get supported TLS capabilities.*

- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

  *Set TLS/IA parameters.*

- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  *Send a TLS/IA PhaseFinished message.*

- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  *Has final phase been completed.*

- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  *Permute TLS/IA inner secret.*

## 6.165.1 Detailed Description

WPA Supplicant / SSL/TLS interface functions for Microsoft Schannel.

**Copyright**

Copyright (c) 2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tls_schannel.c.

## 6.165.2 Function Documentation

### 6.165.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

**Returns:**

Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

Definition at line 763 of file tls_schannel.c.

#### 6.165.2.2 int tls_connection_client_hello_ext (void * *tls_ctx*, struct tls_connection * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***ext_type*** Extension type

***data*** Extension payload (NULL to remove extension)

***data_len*** Extension payload length

**Returns:**

0 on success, -1 on failure

Definition at line 686 of file tls_schannel.c.

#### 6.165.2.3 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to input buffer (encrypted TLS data)

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (decrypted data from TLS tunnel)

***out_len*** Maximum out_data length

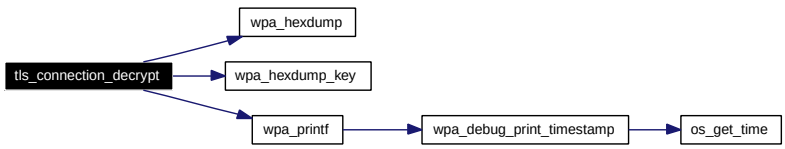
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

Definition at line 571 of file tls_schannel.c.

Here is the call graph for this function:



#### 6.165.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

Definition at line 136 of file tls_schannel.c.

#### 6.165.2.5 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

Definition at line 679 of file tls_schannel.c.

#### 6.165.2.6 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Pointer to plaintext data to be encrypted

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (encrypted TLS data)

***out_len*** Maximum out_data length

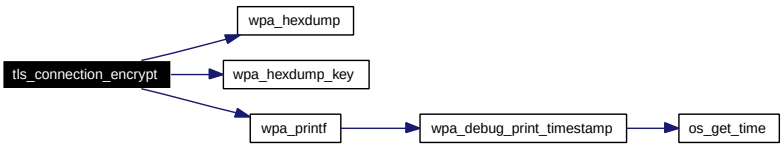
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

Definition at line 483 of file tls_schannel.c.

Here is the call graph for this function:




#### 6.165.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if TLS connection has been completed, 0 if not.

Definition at line 145 of file tls_schannel.c.

#### 6.165.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

Definition at line 694 of file tls_schannel.c.

#### 6.165.2.9 int tls_connection_get_keys (void * *tls_ctx*, struct tls_connection * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***keys*** Structure of key/random data (filled on success)

**Returns:**

0 on success, -1 on failure

Definition at line 187 of file tls_schannel.c.

#### 6.165.2.10 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Number of times a fatal read (remote end reported error) has happened during this connection.

Definition at line 702 of file tls_schannel.c.

#### 6.165.2.11 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

Number of times a fatal write (locally detected error) has happened during this connection.

Definition at line 710 of file tls_schannel.c.

#### 6.165.2.12 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

***appl_data*** Pointer to application data pointer, or NULL if dropped

***appl_data_len*** Pointer to variable that is set to appl_data length

**Returns:**

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server

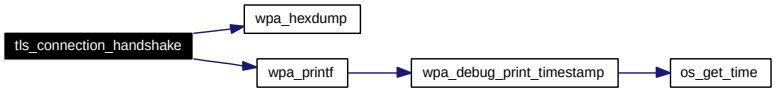
is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

Definition at line 320 of file tls_schannel.c.

Here is the call graph for this function:




#### 6.165.2.13 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

Definition at line 785 of file tls_schannel.c.

#### 6.165.2.14 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***key*** Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

***key_len*** Length of session key material

**Returns:**

0 on success, -1 on failure

Definition at line 792 of file tls_schannel.c.

### 6.165.2.15 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***final*** 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

***out_data*** Pointer to output buffer (encrypted TLS/IA data)

***out_len*** Maximum out_data length

**Returns:**

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

Definition at line 776 of file tls_schannel.c.

### 6.165.2.16 struct tls_connection* tls_connection_init (void * *tls_ctx*)

Initialize a new TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

**Returns:**

Connection context data, conn for other function calls

Definition at line 123 of file tls_schannel.c.

Here is the call graph for this function:


### 6.165.2.17 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***label*** Label (e.g., description of the key) for PRF

***server_random_first*** seed is 0 = client_random|server_random, 1 = server_random|client_random

***out*** Buffer for output data from TLS-PRF

***out_len*** Length of the output buffer

**Returns:**
0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

Definition at line 195 of file tls_schannel.c.

#### 6.165.2.18 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**
1 if current session used session resumption, 0 if not

Definition at line 652 of file tls_schannel.c.

#### 6.165.2.19 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

***out_len*** Length of the output buffer.

**Returns:**
pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

Definition at line 474 of file tls_schannel.c.

#### 6.165.2.20 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***ciphers*** Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_∗).

**Returns:**

0 on success, -1 on failure

Definition at line 665 of file tls_schannel.c.

#### 6.165.2.21 int tls_connection_set_ia (void ∗ *tls_ctx*, struct tls_connection ∗ *conn*, int *tls_ia*)

Set TLS/IA parameters.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***tls_ia*** 1 = enable TLS/IA

**Returns:**

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

Definition at line 769 of file tls_schannel.c.

#### 6.165.2.22 int tls_connection_set_master_key (void ∗ *tls_ctx*, struct tls_connection ∗ *conn*, const u8 ∗ *key*, size_t *key_len*)

Configure master secret for TLS connection.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***key*** TLS pre-master-secret

***key_len*** length of key in bytes

**Returns:**

0 on success, -1 on failure

Definition at line 658 of file tls_schannel.c.

#### 6.165.2.23 int tls_connection_set_params (void ∗ *tls_ctx*, struct tls_connection ∗ *conn*, const struct tls_connection_params ∗ *params*)

Set TLS connection parameters.

**Parameters:**

***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***params*** Connection parameters

**Returns:**
0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 718 of file tls_schannel.c.

Here is the call graph for this function:


### 6.165.2.24 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***verify_peer*** 1 = verify peer certificate

**Returns:**
0 on success, -1 on failure

Definition at line 180 of file tls_schannel.c.

### 6.165.2.25 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

**Returns:**
0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

Definition at line 151 of file tls_schannel.c.

#### 6.165.2.26 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

Definition at line 106 of file tls_schannel.c.

#### 6.165.2.27 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***conn*** Connection context data from tls_connection_init()

***buf*** Buffer for the cipher name

***buflen*** buf size

**Returns:**
0 on success, -1 on failure

Get the name of the currently used cipher.

Definition at line 672 of file tls_schannel.c.

#### 6.165.2.28 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

**Returns:**
Number of found error, 0 if no errors detected.

Process all pending TLS errors.

Definition at line 117 of file tls_schannel.c.

#### 6.165.2.29 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***params*** Global TLS parameters

**Returns:**
0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 167 of file tls_schannel.c.

#### 6.165.2.30 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

**Parameters:**
***tls_ctx*** TLS context data from tls_init()

***check_crl*** 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

**Returns:**
0 on success, -1 on failure

Definition at line 174 of file tls_schannel.c.

#### 6.165.2.31 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

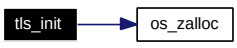
**Parameters:**
***conf*** Configuration data for TLS library

**Returns:**
Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

Definition at line 91 of file tls_schannel.c.

Here is the call graph for this function:

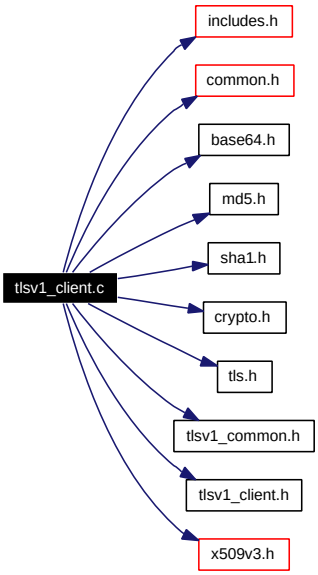


# 6.166 tlsv1_client.c File Reference

wpa_supplicant: TLSv1 client (RFC 2246)

```
#include "includes.h"
#include "common.h"
#include "base64.h"
#include "md5.h"
#include "sha1.h"
#include "crypto.h"
#include "tls.h"
#include "tlsv1_common.h"
#include "tlsv1_client.h"
#include "x509v3.h"
```

Include dependency graph for tlsv1_client.c:




## Defines

- #define **MAX_CIPHER_COUNT** 30

## Functions

- u8 * tlsv1_client_handshake (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  *Process TLS handshake.*

- int tlsv1_client_encrypt (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

*Encrypt data into TLS tunnel.*

- int tlsv1_client_decrypt (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Decrypt data from TLS tunnel.*

- int tlsv1_client_global_init (void)

  *Initialize TLSv1 client.*

- void tlsv1_client_global_deinit (void)

  *Deinitialize TLSv1 client.*

- tlsv1_client * tlsv1_client_init (void)

  *Initialize TLSv1 client connection.*

- void tlsv1_client_deinit (struct tlsv1_client *conn)

  *Deinitialize TLSv1 client connection.*

- int tlsv1_client_established (struct tlsv1_client *conn)

  *Check whether connection has been established.*

- int tlsv1_client_prf (struct tlsv1_client *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  *Use TLS-PRF to derive keying material.*

- int tlsv1_client_get_cipher (struct tlsv1_client *conn, char *buf, size_t buflen)

  *Get current cipher name.*

- int tlsv1_client_shutdown (struct tlsv1_client *conn)

  *Shutdown TLS connection.*

- int tlsv1_client_resumed (struct tlsv1_client *conn)

  *Was session resumption used.*

- int tlsv1_client_hello_ext (struct tlsv1_client *conn, int ext_type, const u8 *data, size_t data_len)

  *Set TLS extension for ClientHello.*

- int tlsv1_client_get_keys (struct tlsv1_client *conn, struct tls_keys *keys)

  *Get master key and random data from TLS connection.*

- int tlsv1_client_set_master_key (struct tlsv1_client *conn, const u8 *key, size_t key_len)

  *Configure master secret for TLS connection.*

- int tlsv1_client_get_keyblock_size (struct tlsv1_client *conn)

  *Get TLS key_block size.*

- int tlsv1_client_set_cipher_list (struct tlsv1_client *conn, u8 *ciphers)

  *Configure acceptable cipher suites.*

- int tlsv1_client_set_ca_cert (struct tlsv1_client *conn, const char *cert, const u8 *cert_blob, size_t cert_blob_len, const char *path)

*Set trusted CA certificate(s).*

- int tlsv1_client_set_client_cert (struct tlsv1_client *conn, const char *cert, const u8 *cert_blob, size_t cert_blob_len)

  *Set client certificate.*

- int tlsv1_client_set_private_key (struct tlsv1_client *conn, const char *private_key, const char *private_key_passwd, const u8 *private_key_blob, size_t private_key_blob_len)

  *Set client private key.*

## 6.166.1 Detailed Description

wpa_supplicant: TLSv1 client (RFC 2246)

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tlsv1_client.c.

## 6.166.2 Function Documentation

### 6.166.2.1 int tlsv1_client_decrypt (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***in_data*** Pointer to input buffer (encrypted TLS data)

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (decrypted data from TLS tunnel)

***out_len*** Maximum out_data length

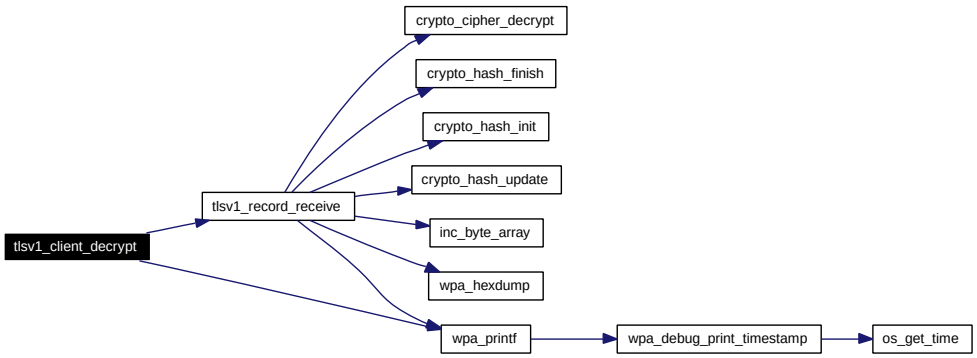
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

Definition at line 1962 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.166.2.2 void tlsv1_client_deinit (struct tlsv1_client * *conn*)

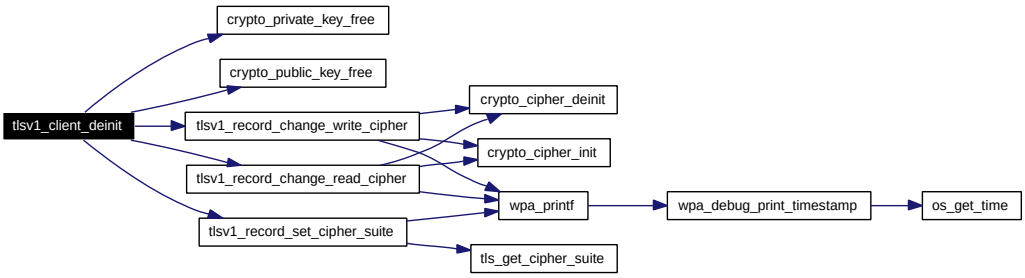
Deinitialize TLSv1 client connection.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

Definition at line 2124 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.166.2.3 int tlsv1_client_encrypt (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***in_data*** Pointer to plaintext data to be encrypted

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (encrypted TLS data)

***out_len*** Maximum out_data length

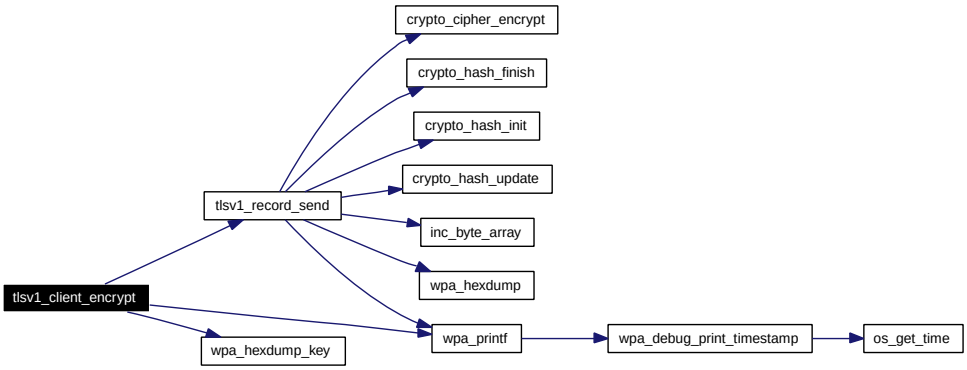
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

Definition at line 1926 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.166.2.4 int tlsv1_client_established (struct tlsv1_client * *conn*)

Check whether connection has been established.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

**Returns:**

1 if connection is established, 0 if not

Definition at line 2146 of file tlsv1_client.c.

#### 6.166.2.5 int tlsv1_client_get_cipher (struct tlsv1_client * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***buf*** Buffer for the cipher name

***buflen*** buf size

**Returns:**

0 on success, -1 on failure

Get the name of the currently used cipher.

Definition at line 2196 of file tlsv1_client.c.

#### 6.166.2.6 int tlsv1_client_get_keyblock_size (struct tlsv1_client * *conn*)

Get TLS key_block size.

**Parameters:**
***conn*** TLSv1 client connection data from tlsv1_client_init()

**Returns:**
Size of the key_block for the negotiated cipher suite or -1 on failure

Definition at line 2365 of file tlsv1_client.c.

#### 6.166.2.7 int tlsv1_client_get_keys (struct tlsv1_client * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

**Parameters:**
***conn*** TLSv1 client connection data from tlsv1_client_init()

***keys*** Structure of key/random data (filled on success)

**Returns:**
0 on success, -1 on failure

Definition at line 2316 of file tlsv1_client.c.

#### 6.166.2.8 void tlsv1_client_global_deinit (void)

Deinitialize TLSv1 client.

This function can be used to deinitialize the TLSv1 client that was initialized by calling tlsv1_client_global_init(). No TLSv1 client functions can be called after this before calling tlsv1_client_global_init() again.

Definition at line 2031 of file tlsv1_client.c.

Here is the call graph for this function:


#### 6.166.2.9 int tlsv1_client_global_init (void)

Initialize TLSv1 client.

**Returns:**
0 on success, -1 on failure

This function must be called before using any other TLSv1 client functions.

Definition at line 2017 of file tlsv1_client.c.

Here is the call graph for this function:


#### 6.166.2.10 u8* tlsv1_client_handshake (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***in_data*** Input data from TLS peer

***in_len*** Input data length

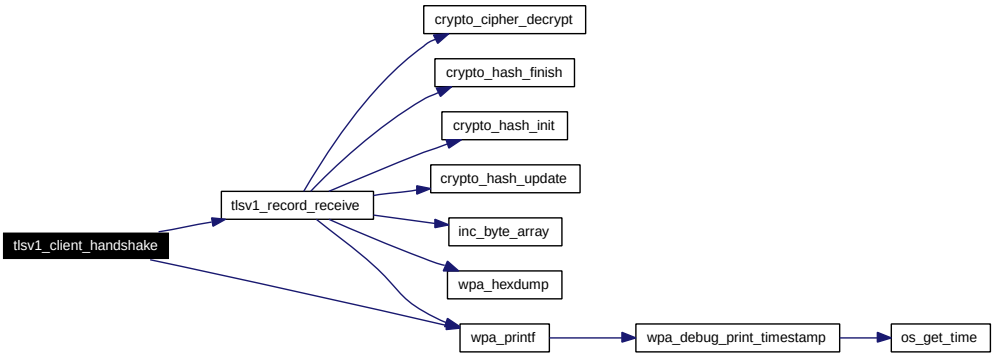
***out_len*** Length of the output buffer.

**Returns:**

Pointer to output data, NULL on failure

Definition at line 1825 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.166.2.11 int tlsv1_client_hello_ext (struct tlsv1_client * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***ext_type*** Extension type

***data*** Extension payload (NULL to remove extension)

***data_len*** Extension payload length

**Returns:**

0 on success, -1 on failure

Definition at line 2274 of file tlsv1_client.c.

Here is the call graph for this function:

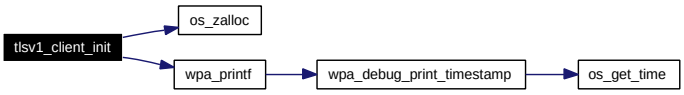

#### 6.166.2.12 struct tlsv1_client* tlsv1_client_init (void)

Initialize TLSv1 client connection.

**Returns:**
Pointer to TLSv1 client connection data or NULL on failure

Definition at line 2085 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.166.2.13 int tlsv1_client_prf (struct tlsv1_client * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

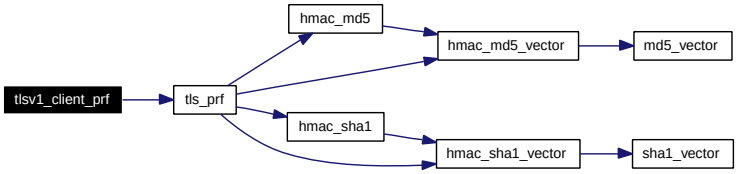
Use TLS-PRF to derive keying material.

**Parameters:**
***conn*** TLSv1 client connection data from tlsv1_client_init()
***label*** Label (e.g., description of the key) for PRF
***server_random_first*** seed is 0 = client_random|server_random, 1 = server_random|client_random
***out*** Buffer for output data from TLS-PRF
***out_len*** Length of the output buffer

**Returns:**
0 on success, -1 on failure

Definition at line 2163 of file tlsv1_client.c.

Here is the call graph for this function:



#### 6.166.2.14 int tlsv1_client_resumed (struct tlsv1_client * *conn*)

Was session resumption used.

**Parameters:**
***conn*** TLSv1 client connection data from tlsv1_client_init()

**Returns:**
1 if current session used session resumption, 0 if not

Definition at line 2259 of file tlsv1_client.c.

#### 6.166.2.15 int tlsv1_client_set_ca_cert (struct tlsv1_client * *conn*, const char * *cert*, const u8 * *cert_blob*, size_t *cert_blob_len*, const char * *path*)

Set trusted CA certificate(s).

**Parameters:**
***conn*** TLSv1 client connection data from tlsv1_client_init()

***cert*** File or reference name for X.509 certificate in PEM or DER format

***cert_blob*** cert as inlined data or NULL if not used

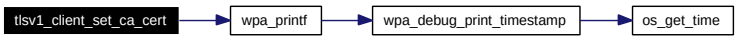
***cert_blob_len*** ca_cert_blob length

***path*** Path to CA certificates (not yet supported)

**Returns:**
0 on success, -1 on failure

Definition at line 2540 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.166.2.16 int tlsv1_client_set_cipher_list (struct tlsv1_client * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

**Parameters:**
***conn*** TLSv1 client connection data from tlsv1_client_init()

***ciphers*** Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

**Returns:**
0 on success, -1 on failure

Definition at line 2383 of file tlsv1_client.c.

#### 6.166.2.17 int tlsv1_client_set_client_cert (struct tlsv1_client * *conn*, const char * *cert*, const u8 * *cert_blob*, size_t *cert_blob_len*)

Set client certificate.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***cert*** File or reference name for X.509 certificate in PEM or DER format

***cert_blob*** cert as inlined data or NULL if not used

***cert_blob_len*** ca_cert_blob length

**Returns:**

0 on success, -1 on failure

Definition at line 2568 of file tlsv1_client.c.

#### 6.166.2.18 int tlsv1_client_set_master_key (struct tlsv1_client * *conn*, const u8 * *key*, size_t *key_len*)

Configure master secret for TLS connection.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***key*** TLS pre-master-secret

***key_len*** length of key in bytes

**Returns:**

0 on success, -1 on failure

Definition at line 2344 of file tlsv1_client.c.

Here is the call graph for this function:


#### 6.166.2.19 int tlsv1_client_set_private_key (struct tlsv1_client * *conn*, const char * *private_key*, const char * *private_key_passwd*, const u8 * *private_key_blob*, size_t *private_key_blob_len*)

Set client private key.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***private_key*** File or reference name for the key in PEM or DER format

***private_key_passwd*** Passphrase for decrypted private key, NULL if no passphrase is used.

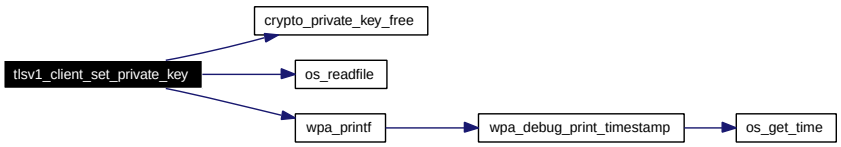
***private_key_blob*** private_key as inlined data or NULL if not used

***private_key_blob_len*** private_key_blob length

**Returns:**
0 on success, -1 on failure

Definition at line 2599 of file tlsv1_client.c.

Here is the call graph for this function:




### 6.166.2.20 int tlsv1_client_shutdown (struct tlsv1_client * *conn*)

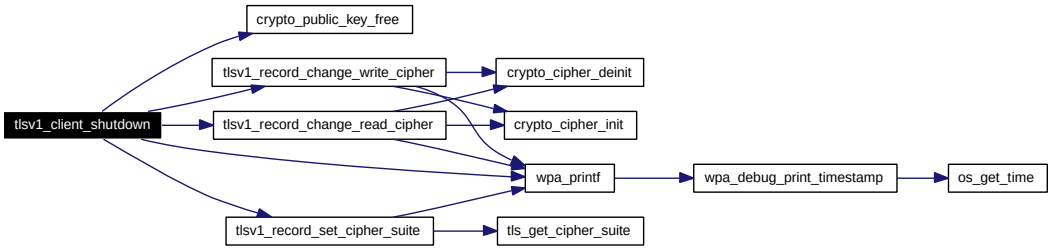
Shutdown TLS connection.

**Parameters:**
***conn*** TLSv1 client connection data from tlsv1_client_init()

**Returns:**
0 on success, -1 on failure

Definition at line 2229 of file tlsv1_client.c.

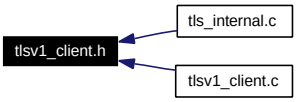
Here is the call graph for this function:



# 6.167 tlsv1_client.h File Reference

wpa_supplicant: TLSv1 client (RFC 2246)

This graph shows which files directly or indirectly include this file:




## Functions

- int tlsv1_client_global_init (void)

  *Initialize TLSv1 client.*

- void tlsv1_client_global_deinit (void)

  *Deinitialize TLSv1 client.*

- tlsv1_client * tlsv1_client_init (void)

  *Initialize TLSv1 client connection.*

- void tlsv1_client_deinit (struct tlsv1_client *conn)

  *Deinitialize TLSv1 client connection.*

- int tlsv1_client_established (struct tlsv1_client *conn)

  *Check whether connection has been established.*

- int tlsv1_client_prf (struct tlsv1_client *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  *Use TLS-PRF to derive keying material.*

- u8 * tlsv1_client_handshake (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  *Process TLS handshake.*

- int tlsv1_client_encrypt (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Encrypt data into TLS tunnel.*

- int tlsv1_client_decrypt (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  *Decrypt data from TLS tunnel.*

- int tlsv1_client_get_cipher (struct tlsv1_client *conn, char *buf, size_t buflen)

  *Get current cipher name.*

- int tlsv1_client_shutdown (struct tlsv1_client *conn)

  *Shutdown TLS connection.*

- int tlsv1_client_resumed (struct tlsv1_client *conn)
  *Was session resumption used.*
- int tlsv1_client_hello_ext (struct tlsv1_client *conn, int ext_type, const u8 *data, size_t data_len)
  *Set TLS extension for ClientHello.*
- int tlsv1_client_get_keys (struct tlsv1_client *conn, struct tls_keys *keys)
  *Get master key and random data from TLS connection.*
- int tlsv1_client_set_master_key (struct tlsv1_client *conn, const u8 *key, size_t key_len)
  *Configure master secret for TLS connection.*
- int tlsv1_client_get_keyblock_size (struct tlsv1_client *conn)
  *Get TLS key_block size.*
- int tlsv1_client_set_cipher_list (struct tlsv1_client *conn, u8 *ciphers)
  *Configure acceptable cipher suites.*
- int tlsv1_client_set_ca_cert (struct tlsv1_client *conn, const char *cert, const u8 *cert_blob, size_t cert_blob_len, const char *path)
  *Set trusted CA certificate(s).*
- int tlsv1_client_set_client_cert (struct tlsv1_client *conn, const char *cert, const u8 *cert_blob, size_t cert_blob_len)
  *Set client certificate.*
- int tlsv1_client_set_private_key (struct tlsv1_client *conn, const char *private_key, const char *private_key_passwd, const u8 *private_key_blob, size_t private_key_blob_len)
  *Set client private key.*

## 6.167.1 Detailed Description

wpa_supplicant: TLSv1 client (RFC 2246)

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tlsv1_client.h.

## 6.167.2 Function Documentation

### 6.167.2.1 int tlsv1_client_decrypt (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***in_data*** Pointer to input buffer (encrypted TLS data)

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (decrypted data from TLS tunnel)

***out_len*** Maximum out_data length

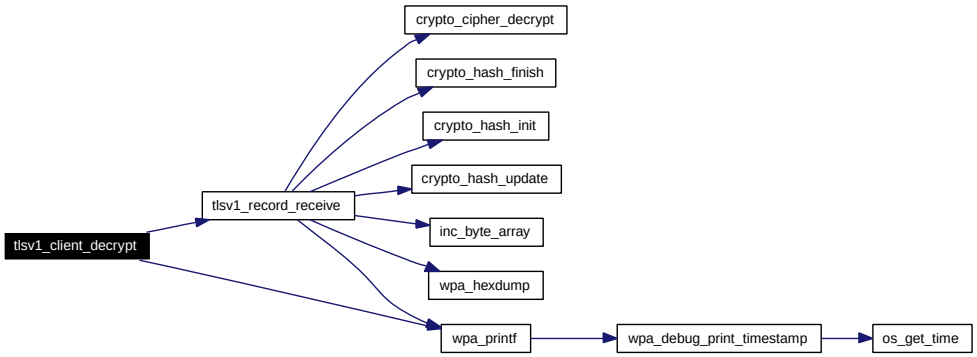
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

Definition at line 1962 of file tlsv1_client.c.

Here is the call graph for this function:




### 6.167.2.2 void tlsv1_client_deinit (struct tlsv1_client * *conn*)

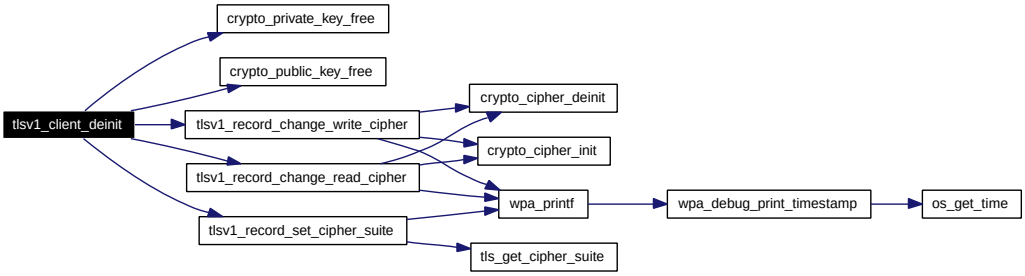
Deinitialize TLSv1 client connection.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

Definition at line 2124 of file tlsv1_client.c.

Here is the call graph for this function:



#### 6.167.2.3 int tlsv1_client_encrypt (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***in_data*** Pointer to plaintext data to be encrypted

***in_len*** Input buffer length

***out_data*** Pointer to output buffer (encrypted TLS data)

***out_len*** Maximum out_data length

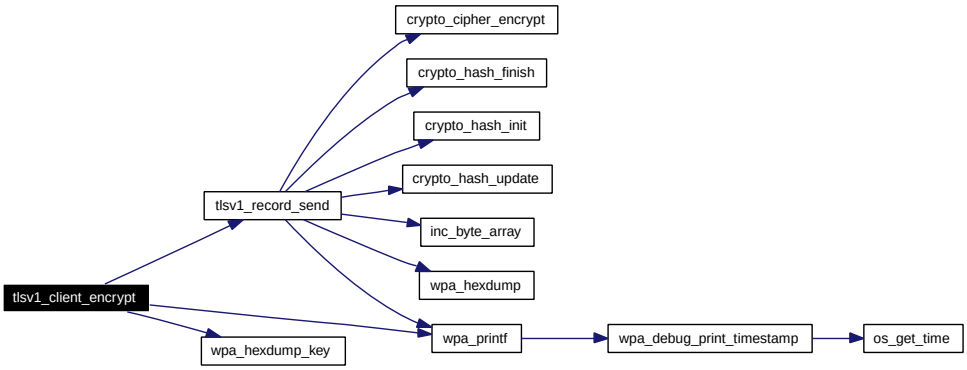
**Returns:**

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

Definition at line 1926 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.167.2.4 int tlsv1_client_established (struct tlsv1_client * *conn*)

Check whether connection has been established.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

**Returns:**

1 if connection is established, 0 if not

Definition at line 2146 of file tlsv1_client.c.

#### 6.167.2.5 int tlsv1_client_get_cipher (struct tlsv1_client * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***buf*** Buffer for the cipher name

***buflen*** buf size

**Returns:**

0 on success, -1 on failure

Get the name of the currently used cipher.

Definition at line 2196 of file tlsv1_client.c.

#### 6.167.2.6 int tlsv1_client_get_keyblock_size (struct tlsv1_client * *conn*)

Get TLS key_block size.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

**Returns:**

Size of the key_block for the negotiated cipher suite or -1 on failure

Definition at line 2365 of file tlsv1_client.c.

#### 6.167.2.7 int tlsv1_client_get_keys (struct tlsv1_client * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***keys*** Structure of key/random data (filled on success)

**Returns:**

0 on success, -1 on failure

Definition at line 2316 of file tlsv1_client.c.

#### 6.167.2.8 void tlsv1_client_global_deinit (void)

Deinitialize TLSv1 client.

This function can be used to deinitialize the TLSv1 client that was initialized by calling tlsv1_client_global_init(). No TLSv1 client functions can be called after this before calling tlsv1_client_global_init() again.

Definition at line 2031 of file tlsv1_client.c.

Here is the call graph for this function:

#### 6.167.2.9 int tlsv1_client_global_init (void)

Initialize TLSv1 client.

**Returns:**
0 on success, -1 on failure

This function must be called before using any other TLSv1 client functions.

Definition at line 2017 of file tlsv1_client.c.

Here is the call graph for this function:


#### 6.167.2.10 u8* tlsv1_client_handshake (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

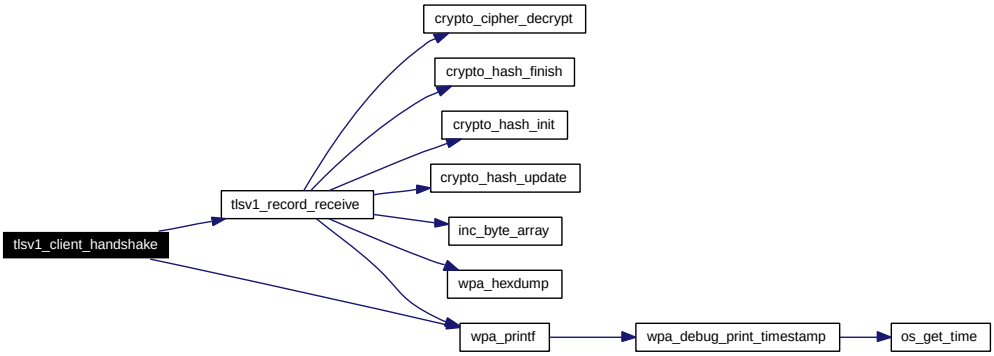
Process TLS handshake.

**Parameters:**
***conn*** TLSv1 client connection data from tlsv1_client_init()
***in_data*** Input data from TLS peer
***in_len*** Input data length
***out_len*** Length of the output buffer.

**Returns:**
Pointer to output data, NULL on failure

Definition at line 1825 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.167.2.11 int tlsv1_client_hello_ext (struct tlsv1_client * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***ext_type*** Extension type

***data*** Extension payload (NULL to remove extension)

***data_len*** Extension payload length

**Returns:**

0 on success, -1 on failure

Definition at line 2274 of file tlsv1_client.c.

Here is the call graph for this function:


### 6.167.2.12 struct tlsv1_client* tlsv1_client_init (void)

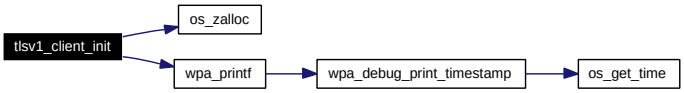
Initialize TLSv1 client connection.

**Returns:**

Pointer to TLSv1 client connection data or NULL on failure

Definition at line 2085 of file tlsv1_client.c.

Here is the call graph for this function:




### 6.167.2.13 int tlsv1_client_prf (struct tlsv1_client * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***label*** Label (e.g., description of the key) for PRF

***server_random_first*** seed is 0 = client_random|server_random, 1 = server_random|client_random

***out*** Buffer for output data from TLS-PRF

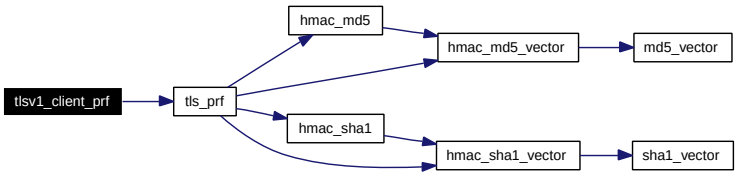
***out_len*** Length of the output buffer

**Returns:**

0 on success, -1 on failure

Definition at line 2163 of file tlsv1_client.c.

Here is the call graph for this function:




#### 6.167.2.14 int tlsv1_client_resumed (struct tlsv1_client * *conn*)

Was session resumption used.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

**Returns:**

1 if current session used session resumption, 0 if not

Definition at line 2259 of file tlsv1_client.c.

#### 6.167.2.15 int tlsv1_client_set_ca_cert (struct tlsv1_client * *conn*, const char * *cert*, const u8 * *cert_blob*, size_t *cert_blob_len*, const char * *path*)

Set trusted CA certificate(s).

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***cert*** File or reference name for X.509 certificate in PEM or DER format

***cert_blob*** cert as inlined data or NULL if not used

***cert_blob_len*** ca_cert_blob length

***path*** Path to CA certificates (not yet supported)

**Returns:**

0 on success, -1 on failure

Definition at line 2540 of file tlsv1_client.c.

Here is the call graph for this function:

#### 6.167.2.16 int tlsv1_client_set_cipher_list (struct tlsv1_client * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***ciphers*** Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

**Returns:**

0 on success, -1 on failure

Definition at line 2383 of file tlsv1_client.c.

#### 6.167.2.17 int tlsv1_client_set_client_cert (struct tlsv1_client * *conn*, const char * *cert*, const u8 * *cert_blob*, size_t *cert_blob_len*)

Set client certificate.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***cert*** File or reference name for X.509 certificate in PEM or DER format

***cert_blob*** cert as inlined data or NULL if not used

***cert_blob_len*** ca_cert_blob length

**Returns:**

0 on success, -1 on failure

Definition at line 2568 of file tlsv1_client.c.

#### 6.167.2.18 int tlsv1_client_set_master_key (struct tlsv1_client * *conn*, const u8 * *key*, size_t *key_len*)

Configure master secret for TLS connection.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***key*** TLS pre-master-secret

***key_len*** length of key in bytes

**Returns:**

0 on success, -1 on failure

Definition at line 2344 of file tlsv1_client.c.

Here is the call graph for this function:

### 6.167.2.19 int tlsv1_client_set_private_key (struct tlsv1_client * *conn*, const char * *private_key*, const char * *private_key_passwd*, const u8 * *private_key_blob*, size_t *private_key_blob_len*)

Set client private key.

**Parameters:**

***conn*** TLSv1 client connection data from tlsv1_client_init()

***private_key*** File or reference name for the key in PEM or DER format

***private_key_passwd*** Passphrase for decrypted private key, NULL if no passphrase is used.

***private_key_blob*** private_key as inlined data or NULL if not used

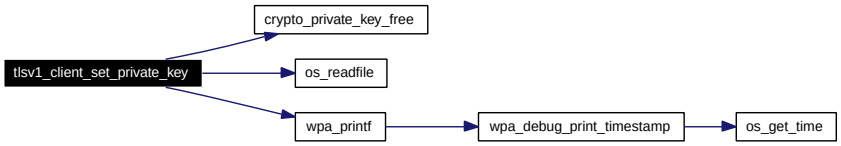
***private_key_blob_len*** private_key_blob length

**Returns:**

0 on success, -1 on failure

Definition at line 2599 of file tlsv1_client.c.

Here is the call graph for this function:




### 6.167.2.20 int tlsv1_client_shutdown (struct tlsv1_client * *conn*)

Shutdown TLS connection.

**Parameters:**

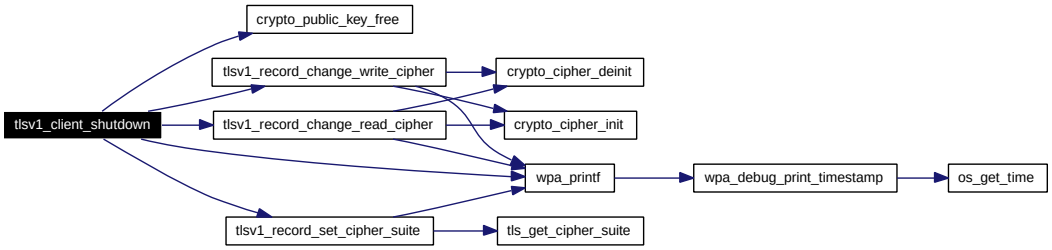
***conn*** TLSv1 client connection data from tlsv1_client_init()

**Returns:**

0 on success, -1 on failure

Definition at line 2229 of file tlsv1_client.c.

Here is the call graph for this function:

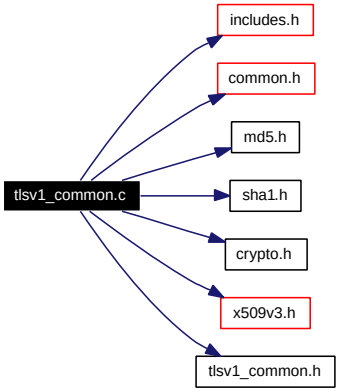


## 6.168 tlsv1_common.c File Reference

wpa_supplicant/hostapd: TLSv1 common routines

```
#include "includes.h"
#include "common.h"
#include "md5.h"
#include "sha1.h"
#include "crypto.h"
#include "x509v3.h"
#include "tlsv1_common.h"
```

Include dependency graph for tlsv1_common.c:




### Defines

- #define **NUM_ELEMS**(a) (sizeof(a) / sizeof((a)[0]))
- #define **NUM_TLS_CIPHER_SUITES** NUM_ELEMS(tls_cipher_suites)
- #define **NUM_TLS_CIPHER_DATA** NUM_ELEMS(tls_ciphers)

### Functions

- const struct tls_cipher_suite * tls_get_cipher_suite (u16 suite)

  *Get TLS cipher suite.*

- int tls_parse_cert (const u8 *buf, size_t len, struct crypto_public_key **pk)

  *Parse DER encoded X.509 certificate and get public key.*

- int tlsv1_record_set_cipher_suite (struct tlsv1_record_layer *rl, u16 cipher_suite)

  *TLS record layer: Set cipher suite.*

- int tlsv1_record_change_write_cipher (struct tlsv1_record_layer *rl)

  *TLS record layer: Change write cipher.*

- int tlsv1_record_change_read_cipher (struct tlsv1_record_layer *rl)

*TLS record layer: Change read cipher.*

- int tlsv1_record_send (struct tlsv1_record_layer *rl, u8 content_type, u8 *buf, size_t buf_size, size_t payload_len, size_t *out_len)

  *TLS record layer: Send a message.*

- int tlsv1_record_receive (struct tlsv1_record_layer *rl, const u8 *in_data, size_t in_len, u8 *out_data, size_t *out_len, u8 *alert)

  *TLS record layer: Process a received message.*

## 6.168.1 Detailed Description

wpa_supplicant/hostapd: TLSv1 common routines

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tlsv1_common.c.

## 6.168.2 Function Documentation

### 6.168.2.1 const struct tls_cipher_suite* tls_get_cipher_suite (u16 *suite*)

Get TLS cipher suite.

**Parameters:**
***suite*** Cipher suite identifier

**Returns:**
Pointer to the cipher data or NULL if not found

Definition at line 96 of file tlsv1_common.c.

### 6.168.2.2 int tls_parse_cert (const u8 * *buf*, size_t *len*, struct crypto_public_key ** *pk*)

Parse DER encoded X.509 certificate and get public key.

**Parameters:**
***buf*** ASN.1 DER encoded certificate

***len*** Length of the buffer

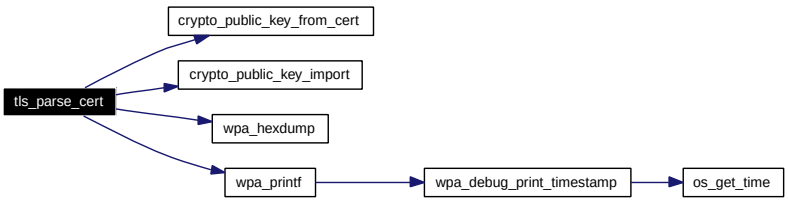
***pk*** Buffer for returning the allocated public key

**Returns:**
0 on success, -1 on failure

This functions parses an ASN.1 DER encoded X.509 certificate and retrieves the public key from it. The caller is responsible for freeing the public key by calling crypto_public_key_free().

Definition at line 128 of file tlsv1_common.c.

Here is the call graph for this function:




#### 6.168.2.3 int tlsv1_record_change_read_cipher (struct tlsv1_record_layer * *rl*)

TLS record layer: Change read cipher.

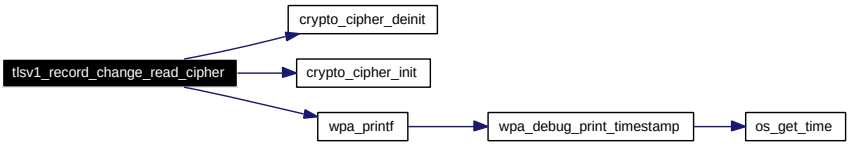
**Parameters:**
*rl* Pointer to TLS record layer data

**Returns:**
0 on success (cipher changed), -1 on failure

This function changes TLS record layer to use the new cipher suite configured with tlsv1_record_set_cipher_suite() for reading.

Definition at line 261 of file tlsv1_common.c.

Here is the call graph for this function:




#### 6.168.2.4 int tlsv1_record_change_write_cipher (struct tlsv1_record_layer * *rl*)

TLS record layer: Change write cipher.

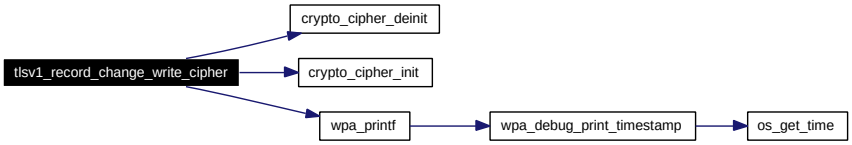
**Parameters:**
*rl* Pointer to TLS record layer data

**Returns:**
0 on success (cipher changed), -1 on failure

This function changes TLS record layer to use the new cipher suite configured with tlsv1_record_set_cipher_suite() for writing.

Definition at line 226 of file tlsv1_common.c.

Here is the call graph for this function:




#### 6.168.2.5 int tlsv1_record_receive (struct tlsv1_record_layer * *rl*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t * *out_len*, u8 * *alert*)

TLS record layer: Process a received message.

**Parameters:**

***rl*** Pointer to TLS record layer data

***in_data*** Received data

***in_len*** Length of the received data

***out_data*** Buffer for output data (must be at least as long as in_data)

***out_len*** Set to maximum out_data length by caller; used to return the length of the used data

***alert*** Buffer for returning an alert value on failure

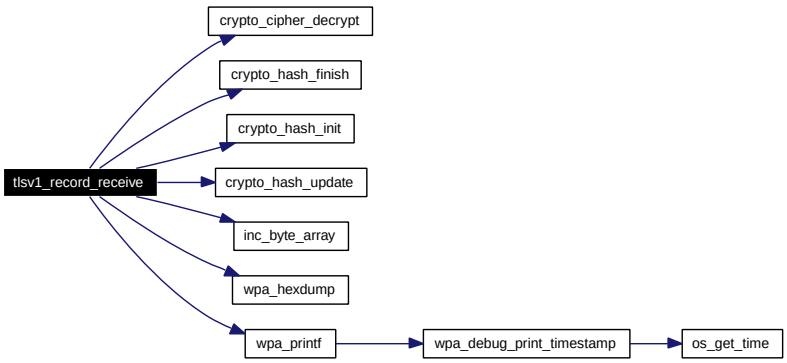
**Returns:**

0 on success, -1 on failure

This function decrypts the received message, verifies HMAC and TLS record layer header.

Definition at line 397 of file tlsv1_common.c.

Here is the call graph for this function:




#### 6.168.2.6 int tlsv1_record_send (struct tlsv1_record_layer * *rl*, u8 *content_type*, u8 * *buf*, size_t *buf_size*, size_t *payload_len*, size_t * *out_len*)

TLS record layer: Send a message.

**Parameters:**

***rl*** Pointer to TLS record layer data

***content_type*** Content type (TLS_CONTENT_TYPE_∗)

***buf*** Buffer to send (with TLS_RECORD_HEADER_LEN octets reserved in the beginning for record layer to fill in; payload filled in after this and extra space in the end for HMAC).

***buf_size*** Maximum buf size

***payload_len*** Length of the payload

***out_len*** Buffer for returning the used buf length

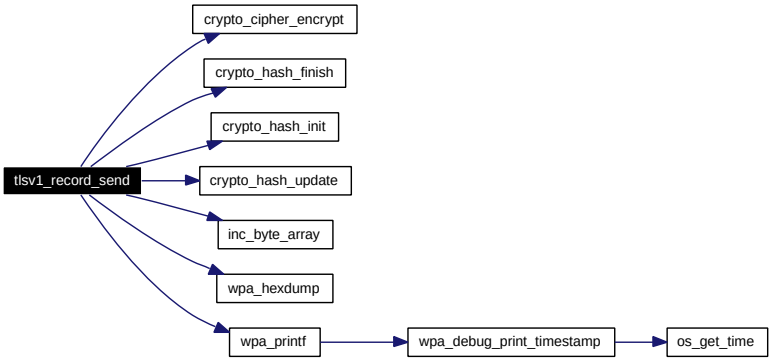
**Returns:**

0 on success, -1 on failure

This function fills in the TLS record layer header, adds HMAC, and encrypts the data using the current write cipher.

Definition at line 303 of file tlsv1_common.c.

Here is the call graph for this function:




### 6.168.2.7 int tlsv1_record_set_cipher_suite (struct tlsv1_record_layer ∗ *rl*, u16 *cipher_suite*)

TLS record layer: Set cipher suite.

**Parameters:**

***rl*** Pointer to TLS record layer data

***cipher_suite*** New cipher suite

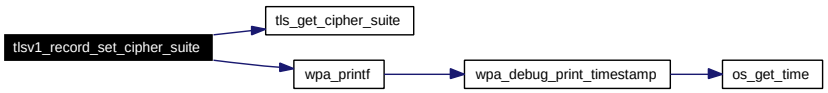
**Returns:**

0 on success, -1 on failure

This function is used to prepare TLS record layer for cipher suite change. tlsv1_record_change_write_cipher() and tlsv1_record_change_read_cipher() functions can then be used to change the currently used ciphers.

Definition at line 183 of file tlsv1_common.c.

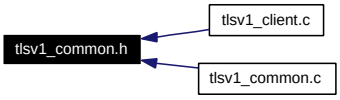
Here is the call graph for this function:


tlsv1_record_set_cipher_suite
tls_get_cipher_suite
wpa_printf
wpa_debug_print_timestamp
os_get_time

## 6.169 tlsv1_common.h File Reference

wpa_supplicant/hostapd: TLSv1 common definitions

This graph shows which files directly or indirectly include this file:




### Defines

- #define **TLS_VERSION** 0x0301
- #define **TLS_RANDOM_LEN** 32
- #define **TLS_PRE_MASTER_SECRET_LEN** 48
- #define **TLS_MASTER_SECRET_LEN** 48
- #define **TLS_SESSION_ID_MAX_LEN** 32
- #define **TLS_VERIFY_DATA_LEN** 12
- #define **TLS_MAX_WRITE_MAC_SECRET_LEN** 20
- #define **TLS_MAX_WRITE_KEY_LEN** 32
- #define **TLS_MAX_IV_LEN** 16
- #define **TLS_MAX_KEY_BLOCK_LEN**
- #define **TLS_SEQ_NUM_LEN** 8
- #define **TLS_RECORD_HEADER_LEN** 5
- #define **TLS_NULL_WITH_NULL_NULL** 0x0000
- #define **TLS_RSA_WITH_NULL_MD5** 0x0001
- #define **TLS_RSA_WITH_NULL_SHA** 0x0002
- #define **TLS_RSA_EXPORT_WITH_RC4_40_MD5** 0x0003
- #define **TLS_RSA_WITH_RC4_128_MD5** 0x0004
- #define **TLS_RSA_WITH_RC4_128_SHA** 0x0005
- #define **TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5** 0x0006
- #define **TLS_RSA_WITH_IDEA_CBC_SHA** 0x0007
- #define **TLS_RSA_EXPORT_WITH_DES40_CBC_SHA** 0x0008
- #define **TLS_RSA_WITH_DES_CBC_SHA** 0x0009
- #define **TLS_RSA_WITH_3DES_EDE_CBC_SHA** 0x000A
- #define **TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA** 0x000B
- #define **TLS_DH_DSS_WITH_DES_CBC_SHA** 0x000C
- #define **TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA** 0x000D
- #define **TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA** 0x000E
- #define **TLS_DH_RSA_WITH_DES_CBC_SHA** 0x000F
- #define **TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA** 0x0010
- #define **TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA** 0x0011
- #define **TLS_DHE_DSS_WITH_DES_CBC_SHA** 0x0012
- #define **TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA** 0x0013
- #define **TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA** 0x0014
- #define **TLS_DHE_RSA_WITH_DES_CBC_SHA** 0x0015
- #define **TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA** 0x0016
- #define **TLS_DH_anon_EXPORT_WITH_RC4_40_MD5** 0x0017
- #define **TLS_DH_anon_WITH_RC4_128_MD5** 0x0018

- #define **TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA** 0x0019
- #define **TLS_DH_anon_WITH_DES_CBC_SHA** 0x001A
- #define **TLS_DH_anon_WITH_3DES_EDE_CBC_SHA** 0x001B
- #define **TLS_RSA_WITH_AES_128_CBC_SHA** 0x002F
- #define **TLS_DH_DSS_WITH_AES_128_CBC_SHA** 0x0030
- #define **TLS_DH_RSA_WITH_AES_128_CBC_SHA** 0x0031
- #define **TLS_DHE_DSS_WITH_AES_128_CBC_SHA** 0x0032
- #define **TLS_DHE_RSA_WITH_AES_128_CBC_SHA** 0x0033
- #define **TLS_DH_anon_WITH_AES_128_CBC_SHA** 0x0034
- #define **TLS_RSA_WITH_AES_256_CBC_SHA** 0x0035
- #define **TLS_DH_DSS_WITH_AES_256_CBC_SHA** 0x0036
- #define **TLS_DH_RSA_WITH_AES_256_CBC_SHA** 0x0037
- #define **TLS_DHE_DSS_WITH_AES_256_CBC_SHA** 0x0038
- #define **TLS_DHE_RSA_WITH_AES_256_CBC_SHA** 0x0039
- #define **TLS_DH_anon_WITH_AES_256_CBC_SHA** 0x003A
- #define **TLS_COMPRESSION_NULL** 0
- #define **TLS_ALERT_LEVEL_WARNING** 1
- #define **TLS_ALERT_LEVEL_FATAL** 2
- #define **TLS_ALERT_CLOSE_NOTIFY** 0
- #define **TLS_ALERT_UNEXPECTED_MESSAGE** 10
- #define **TLS_ALERT_BAD_RECORD_MAC** 20
- #define **TLS_ALERT_DECRYPTION_FAILED** 21
- #define **TLS_ALERT_RECORD_OVERFLOW** 22
- #define **TLS_ALERT_DECOMPRESSION_FAILURE** 30
- #define **TLS_ALERT_HANDSHAKE_FAILURE** 40
- #define **TLS_ALERT_BAD_CERTIFICATE** 42
- #define **TLS_ALERT_UNSUPPORTED_CERTIFICATE** 43
- #define **TLS_ALERT_CERTIFICATE_REVOKED** 44
- #define **TLS_ALERT_CERTIFICATE_EXPIRED** 45
- #define **TLS_ALERT_CERTIFICATE_UNKNOWN** 46
- #define **TLS_ALERT_ILLEGAL_PARAMETER** 47
- #define **TLS_ALERT_UNKNOWN_CA** 48
- #define **TLS_ALERT_ACCESS_DENIED** 49
- #define **TLS_ALERT_DECODE_ERROR** 50
- #define **TLS_ALERT_DECRYPT_ERROR** 51
- #define **TLS_ALERT_EXPORT_RESTRICTION** 60
- #define **TLS_ALERT_PROTOCOL_VERSION** 70
- #define **TLS_ALERT_INSUFFICIENT_SECURITY** 71
- #define **TLS_ALERT_INTERNAL_ERROR** 80
- #define **TLS_ALERT_USER_CANCELED** 90
- #define **TLS_ALERT_NO_RENEGOTIATION** 100
- #define **TLS_EXT_PAC_OPAQUE** 35

## Enumerations

- enum { **TLS_CONTENT_TYPE_CHANGE_CIPHER_SPEC** = 20, **TLS_CONTENT_TYPE_ALERT** = 21, **TLS_CONTENT_TYPE_HANDSHAKE** = 22, **TLS_CONTENT_TYPE_APPLICATION_DATA** = 23 }
- enum {

  **TLS_HANDSHAKE_TYPE_HELLO_REQUEST** = 0, **TLS_HANDSHAKE_TYPE_CLIENT_HELLO** = 1, **TLS_HANDSHAKE_TYPE_SERVER_HELLO** = 2, **TLS_HANDSHAKE_TYPE_CERTIFICATE** = 11,

  **TLS_HANDSHAKE_TYPE_SERVER_KEY_EXCHANGE** = 12, **TLS_HANDSHAKE_TYPE_CERTIFICATE_REQUEST** = 13, **TLS_HANDSHAKE_TYPE_SERVER_HELLO_DONE** = 14, **TLS_HANDSHAKE_TYPE_CERTIFICATE_VERIFY** = 15,

  **TLS_HANDSHAKE_TYPE_CLIENT_KEY_EXCHANGE** = 16, **TLS_HANDSHAKE_TYPE_FINISHED** = 20 }
- enum { **TLS_CHANGE_CIPHER_SPEC** = 1 }
- enum **tls_key_exchange** {

  **TLS_KEY_X_NULL**, **TLS_KEY_X_RSA**, **TLS_KEY_X_RSA_EXPORT**, **TLS_KEY_X_DH_DSS_EXPORT**,

  **TLS_KEY_X_DH_DSS**, **TLS_KEY_X_DH_RSA_EXPORT**, **TLS_KEY_X_DH_RSA**, **TLS_KEY_X_DHE_DSS_EXPORT**,

  **TLS_KEY_X_DHE_DSS**, **TLS_KEY_X_DHE_RSA_EXPORT**, **TLS_KEY_X_DHE_RSA**, **TLS_KEY_X_DH_anon_EXPORT**,

  **TLS_KEY_X_DH_anon** }
- enum **tls_cipher** {

  **TLS_CIPHER_NULL**, **TLS_CIPHER_RC4_40**, **TLS_CIPHER_RC4_128**, **TLS_CIPHER_RC2_CBC_40**,

  **TLS_CIPHER_IDEA_CBC**, **TLS_CIPHER_DES40_CBC**, **TLS_CIPHER_DES_CBC**, **TLS_CIPHER_3DES_EDE_CBC**,

  **TLS_CIPHER_AES_128_CBC**, **TLS_CIPHER_AES_256_CBC** }
- enum **tls_hash** { **TLS_HASH_NULL**, **TLS_HASH_MD5**, **TLS_HASH_SHA** }
- enum **tls_cipher_type** { **TLS_CIPHER_STREAM**, **TLS_CIPHER_BLOCK** }

## Functions

- const struct tls_cipher_suite * tls_get_cipher_suite (u16 suite)

  *Get TLS cipher suite.*

- int tls_parse_cert (const u8 *buf, size_t len, struct crypto_public_key **pk)

  *Parse DER encoded X.509 certificate and get public key.*

- int tlsv1_record_set_cipher_suite (struct tlsv1_record_layer *rl, u16 cipher_suite)

  *TLS record layer: Set cipher suite.*

- int tlsv1_record_change_write_cipher (struct tlsv1_record_layer *rl)

  *TLS record layer: Change write cipher.*

- int tlsv1_record_change_read_cipher (struct tlsv1_record_layer *rl)

  *TLS record layer: Change read cipher.*

- int tlsv1_record_send (struct tlsv1_record_layer *rl, u8 content_type, u8 *buf, size_t buf_size, size_t payload_len, size_t *out_len)

  *TLS record layer: Send a message.*

- int tlsv1_record_receive (struct tlsv1_record_layer *rl, const u8 *in_data, size_t in_len, u8 *out_data, size_t *out_len, u8 *alert)

  *TLS record layer: Process a received message.*

## 6.169.1 Detailed Description

wpa_supplicant/hostapd: TLSv1 common definitions

**Copyright**
Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file tlsv1_common.h.

## 6.169.2 Define Documentation

### 6.169.2.1 #define TLS_MAX_KEY_BLOCK_LEN

**Value:**

```
(2 * (TLS_MAX_WRITE_MAC_SECRET_LEN + \
                                      TLS_MAX_WRITE_KEY_LEN + TLS_MAX_IV_LEN))
```

Definition at line 28 of file tlsv1_common.h.

## 6.169.3 Function Documentation

### 6.169.3.1 const struct tls_cipher_suite* tls_get_cipher_suite (u16 *suite*)

Get TLS cipher suite.

**Parameters:**
***suite*** Cipher suite identifier

**Returns:**
Pointer to the cipher data or NULL if not found

Definition at line 96 of file tlsv1_common.c.

#### 6.169.3.2 int tls_parse_cert (const u8 * *buf*, size_t *len*, struct crypto_public_key ** *pk*)

Parse DER encoded X.509 certificate and get public key.

**Parameters:**

***buf*** ASN.1 DER encoded certificate

***len*** Length of the buffer

***pk*** Buffer for returning the allocated public key

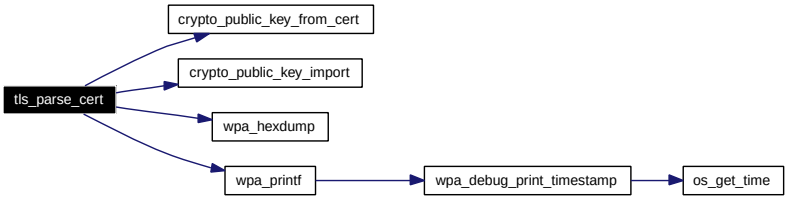
**Returns:**

0 on success, -1 on failure

This functions parses an ASN.1 DER encoded X.509 certificate and retrieves the public key from it. The caller is responsible for freeing the public key by calling crypto_public_key_free().

Definition at line 128 of file tlsv1_common.c.

Here is the call graph for this function:




#### 6.169.3.3 int tlsv1_record_change_read_cipher (struct tlsv1_record_layer * *rl*)

TLS record layer: Change read cipher.

**Parameters:**

***rl*** Pointer to TLS record layer data

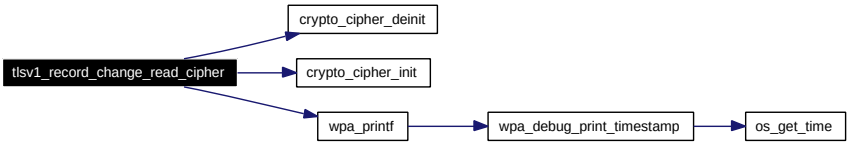
**Returns:**

0 on success (cipher changed), -1 on failure

This function changes TLS record layer to use the new cipher suite configured with tlsv1_record_set_cipher_suite() for reading.

Definition at line 261 of file tlsv1_common.c.

Here is the call graph for this function:



#### 6.169.3.4 int tlsv1_record_change_write_cipher (struct tlsv1_record_layer * *rl*)

TLS record layer: Change write cipher.

**Parameters:**

***rl*** Pointer to TLS record layer data

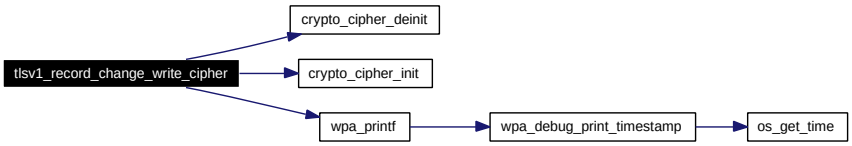
**Returns:**

0 on success (cipher changed), -1 on failure

This function changes TLS record layer to use the new cipher suite configured with tlsv1_record_set_cipher_suite() for writing.

Definition at line 226 of file tlsv1_common.c.

Here is the call graph for this function:




#### 6.169.3.5 int tlsv1_record_receive (struct tlsv1_record_layer * *rl*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t * *out_len*, u8 * *alert*)

TLS record layer: Process a received message.

**Parameters:**

***rl*** Pointer to TLS record layer data

***in_data*** Received data

***in_len*** Length of the received data

***out_data*** Buffer for output data (must be at least as long as in_data)

***out_len*** Set to maximum out_data length by caller; used to return the length of the used data

***alert*** Buffer for returning an alert value on failure

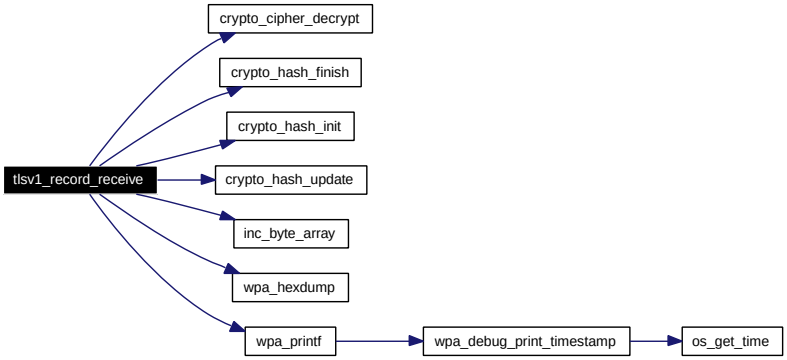
**Returns:**

0 on success, -1 on failure

This function decrypts the received message, verifies HMAC and TLS record layer header.

Definition at line 397 of file tlsv1_common.c.

Here is the call graph for this function:




#### 6.169.3.6 int tlsv1_record_send (struct tlsv1_record_layer * *rl*, u8 *content_type*, u8 * *buf*, size_t *buf_size*, size_t *payload_len*, size_t * *out_len*)

TLS record layer: Send a message.

**Parameters:**

***rl*** Pointer to TLS record layer data

***content_type*** Content type (TLS_CONTENT_TYPE_*)

***buf*** Buffer to send (with TLS_RECORD_HEADER_LEN octets reserved in the beginning for record layer to fill in; payload filled in after this and extra space in the end for HMAC).

***buf_size*** Maximum buf size

***payload_len*** Length of the payload

***out_len*** Buffer for returning the used buf length

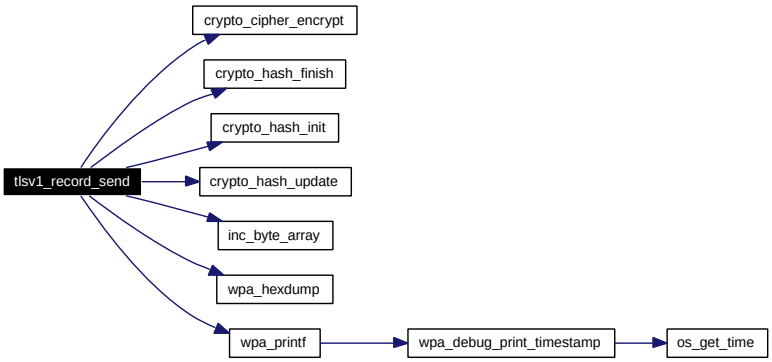
**Returns:**

0 on success, -1 on failure

This function fills in the TLS record layer header, adds HMAC, and encrypts the data using the current write cipher.

Definition at line 303 of file tlsv1_common.c.

Here is the call graph for this function:



#### 6.169.3.7 int tlsv1_record_set_cipher_suite (struct tlsv1_record_layer * *rl*, u16 *cipher_suite*)

TLS record layer: Set cipher suite.

**Parameters:**

***rl*** Pointer to TLS record layer data

***cipher_suite*** New cipher suite

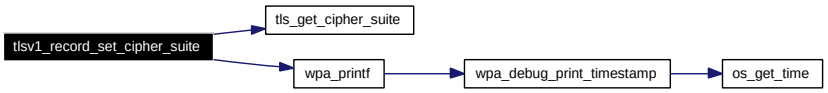
**Returns:**

0 on success, -1 on failure

This function is used to prepare TLS record layer for cipher suite change. tlsv1_record_change_write_cipher() and tlsv1_record_change_read_cipher() functions can then be used to change the currently used ciphers.

Definition at line 183 of file tlsv1_common.c.

Here is the call graph for this function:

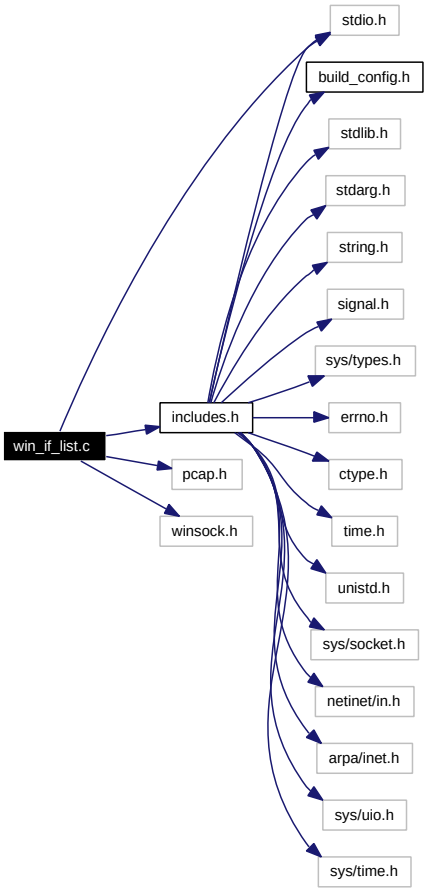


## 6.170 win_if_list.c File Reference

win_if_list - Display network interfaces with description (for Windows)

```
#include "includes.h"
#include <stdio.h>
#include "pcap.h"
#include <winsock.h>
```

Include dependency graph for win_if_list.c:




### Functions

- int **main** (int argc, char *argv[ ])

### 6.170.1 Detailed Description

win_if_list - Display network interfaces with description (for Windows)

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This small tool is for the Windows build to provide an easy way of fetching a list of available network interfaces.

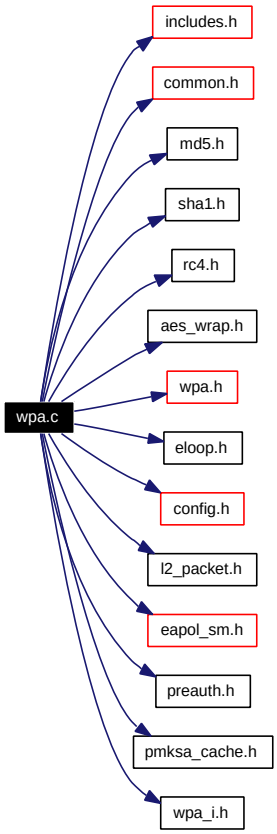
Definition in file win_if_list.c.

## 6.171 wpa.c File Reference

WPA Supplicant - WPA state machine and EAPOL-Key processing.

```
#include "includes.h"
#include "common.h"
#include "md5.h"
#include "sha1.h"
#include "rc4.h"
#include "aes_wrap.h"
#include "wpa.h"
#include "eloop.h"
#include "config.h"
#include "l2_packet.h"
#include "eapol_sm.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_i.h"
```

Include dependency graph for wpa.c:



## Defines

- #define **WPA_KEY_INFO_TYPE_MASK** ((u16) (BIT(0) | BIT(1) | BIT(2)))
- #define **WPA_KEY_INFO_TYPE_HMAC_MD5_RC4** BIT(0)
- #define **WPA_KEY_INFO_TYPE_HMAC_SHA1_AES** BIT(1)
- #define **WPA_KEY_INFO_KEY_TYPE** BIT(3)
- #define **WPA_KEY_INFO_KEY_INDEX_MASK** (BIT(4) | BIT(5))
- #define **WPA_KEY_INFO_KEY_INDEX_SHIFT** 4
- #define **WPA_KEY_INFO_INSTALL** BIT(6)
- #define **WPA_KEY_INFO_TXRX** BIT(6)
- #define **WPA_KEY_INFO_ACK** BIT(7)
- #define **WPA_KEY_INFO_MIC** BIT(8)
- #define **WPA_KEY_INFO_SECURE** BIT(9)
- #define **WPA_KEY_INFO_ERROR** BIT(10)
- #define **WPA_KEY_INFO_REQUEST** BIT(11)
- #define **WPA_KEY_INFO_ENCR_KEY_DATA** BIT(12)
- #define **WPA_KEY_INFO_SMK_MESSAGE** BIT(13)
- #define **RSN_SUITE** "%02x-%02x-%02x-%d"
- #define **RSN_SUITE_ARG**(s) (s)[0], (s)[1], (s)[2], (s)[3]

## Functions

- int wpa_parse_wpa_ie (const u8 *wpa_ie, size_t wpa_ie_len, struct wpa_ie_data *data)
  *Parse WPA/RSN IE.*
- void wpa_sm_key_request (struct wpa_sm *sm, int error, int pairwise)
  *Send EAPOL-Key Request.*
- int wpa_sm_stkstart (struct wpa_sm *sm, const u8 *peer)
  *Send EAPOL-Key Request for STK handshake (STK M1).*
- void wpa_sm_aborted_cached (struct wpa_sm *sm)
  *Notify WPA that PMKSA caching was aborted.*
- int wpa_sm_rx_eapol (struct wpa_sm *sm, const u8 *src_addr, const u8 *buf, size_t len)
  *Process received WPA EAPOL frames.*
- int wpa_sm_get_mib (struct wpa_sm *sm, char *buf, size_t buflen)
  *Dump text list of MIB entries.*
- wpa_sm * wpa_sm_init (struct wpa_sm_ctx *ctx)
  *Initialize WPA state machine.*
- void wpa_sm_deinit (struct wpa_sm *sm)
  *Deinitialize WPA state machine.*
- void wpa_sm_notify_assoc (struct wpa_sm *sm, const u8 *bssid)
  *Notify WPA state machine about association.*
- void wpa_sm_notify_disassoc (struct wpa_sm *sm)

*Notify WPA state machine about disassociation.*

- void wpa_sm_set_pmk (struct wpa_sm *sm, const u8 *pmk, size_t pmk_len)
  *Set PMK.*
- void wpa_sm_set_pmk_from_pmksa (struct wpa_sm *sm)
  *Set PMK based on the current PMKSA.*
- void wpa_sm_set_fast_reauth (struct wpa_sm *sm, int fast_reauth)
  *Set fast reauthentication (EAP) enabled/disabled.*
- void wpa_sm_set_scard_ctx (struct wpa_sm *sm, void *scard_ctx)
  *Set context pointer for smartcard callbacks.*
- void wpa_sm_set_config (struct wpa_sm *sm, struct wpa_ssid *config)
  *Notification of current configration change.*
- void wpa_sm_set_own_addr (struct wpa_sm *sm, const u8 *addr)
  *Set own MAC address.*
- void wpa_sm_set_ifname (struct wpa_sm *sm, const char *ifname, const char *bridge_ifname)
  *Set network interface name.*
- void wpa_sm_set_eapol (struct wpa_sm *sm, struct eapol_sm *eapol)
  *Set EAPOL state machine pointer.*
- int wpa_sm_set_param (struct wpa_sm *sm, enum wpa_sm_conf_params param, unsigned int value)
  *Set WPA state machine parameters.*
- unsigned int wpa_sm_get_param (struct wpa_sm *sm, enum wpa_sm_conf_params param)
  *Get WPA state machine parameters.*
- int wpa_sm_get_status (struct wpa_sm *sm, char *buf, size_t buflen, int verbose)
  *Get WPA state machine.*
- int wpa_sm_set_assoc_wpa_ie_default (struct wpa_sm *sm, u8 *wpa_ie, size_t *wpa_ie_len)
  *Generate own WPA/RSN IE from configuration.*
- int wpa_sm_set_assoc_wpa_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  *Set own WPA/RSN IE from (Re)AssocReq.*
- int wpa_sm_set_ap_wpa_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  *Set AP WPA IE from Beacon/ProbeResp.*
- int wpa_sm_set_ap_rsn_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  *Set AP RSN IE from Beacon/ProbeResp.*
- int wpa_sm_parse_own_wpa_ie (struct wpa_sm *sm, struct wpa_ie_data *data)
  *Parse own WPA/RSN IE.*

## Variables

- wpa_ie_hdr **STRUCT_PACKED**

### 6.171.1 Detailed Description

WPA Supplicant - WPA state machine and EAPOL-Key processing.

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file wpa.c.

### 6.171.2 Function Documentation

#### 6.171.2.1 int wpa_parse_wpa_ie (const u8 * *wpa_ie*, size_t *wpa_ie_len*, struct wpa_ie_data * *data*)

Parse WPA/RSN IE.

**Parameters:**
***wpa_ie*** Pointer to WPA or RSN IE
***wpa_ie_len*** Length of the WPA/RSN IE
***data*** Pointer to data area for parsing results

**Returns:**
0 on success, -1 on failure

Parse the contents of WPA or RSN IE and write the parsed data into data.

Definition at line 647 of file wpa.c.

#### 6.171.2.2 void wpa_sm_aborted_cached (struct wpa_sm * *sm*)

Notify WPA that PMKSA caching was aborted.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 3343 of file wpa.c.

Here is the call graph for this function:

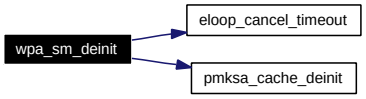
#### 6.171.2.3 void wpa_sm_deinit (struct wpa_sm * *sm*)

Deinitialize WPA state machine.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 3861 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.4 int wpa_sm_get_mib (struct wpa_sm * *sm*, char * *buf*, size_t *buflen*)

Dump text list of MIB entries.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()
***buf*** Buffer for the list
***buflen*** Length of the buffer

**Returns:**
Number of bytes written to buffer

This function is used fetch dot11 MIB variables.

Definition at line 3726 of file wpa.c.

Here is the call graph for this function:


#### 6.171.2.5 unsigned int wpa_sm_get_param (struct wpa_sm * *sm*, enum wpa_sm_conf_params *param*)

Get WPA state machine parameters.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()
***param*** Parameter field

**Returns:**
Parameter value

Definition at line 4125 of file wpa.c.

#### 6.171.2.6 int wpa_sm_get_status (struct wpa_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get WPA state machine.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

**Returns:**

Number of bytes written to buf.

Query WPA state machine for status information. This function fills in a text area with current status information. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

Definition at line 4168 of file wpa.c.

#### 6.171.2.7 struct wpa_sm* wpa_sm_init (struct wpa_sm_ctx * *ctx*)

Initialize WPA state machine.

**Parameters:**

***ctx*** Context pointer for callbacks; this needs to be an allocated buffer

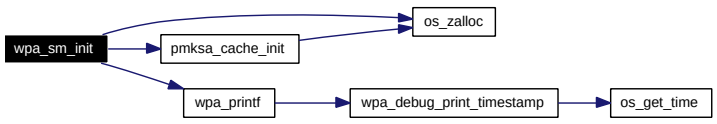
**Returns:**

Pointer to the allocated WPA state machine data

This function is used to allocate a new WPA state machine and the returned value is passed to all WPA state machine calls.

Definition at line 3830 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.8 void wpa_sm_key_request (struct wpa_sm * *sm*, int *error*, int *pairwise*)

Send EAPOL-Key Request.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***error*** Indicate whether this is an Michael MIC error report

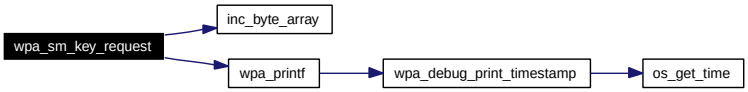
***pairwise*** 1 = error report for pairwise packet, 0 = for group packet

**Returns:**
Pointer to the current network structure or NULL on failure

Send an EAPOL-Key Request to the current authenticator. This function is used to request rekeying and it is usually called when a local Michael MIC failure is detected.

Definition at line 990 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.9 void wpa_sm_notify_assoc (struct wpa_sm * *sm*, const u8 * *bssid*)

Notify WPA state machine about association.

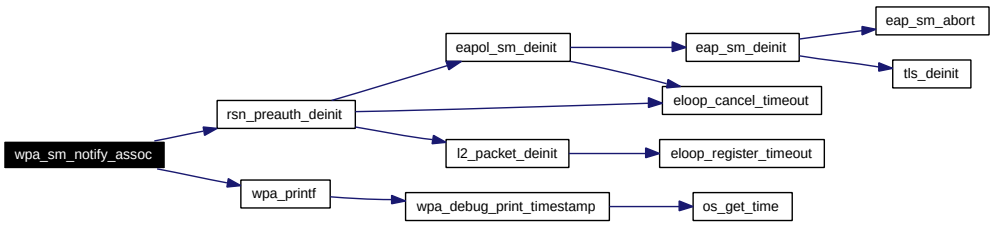
**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***bssid*** The BSSID of the new association

This function is called to let WPA state machine know that the connection was established.

Definition at line 3894 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.10 void wpa_sm_notify_disassoc (struct wpa_sm * *sm*)

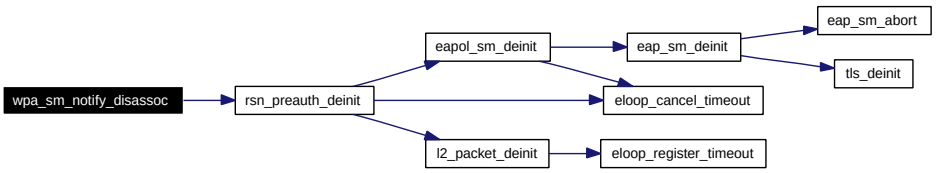
Notify WPA state machine about disassociation.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

This function is called to let WPA state machine know that the connection was lost. This will abort any existing pre-authentication session.

Definition at line 3917 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.11 int wpa_sm_parse_own_wpa_ie (struct wpa_sm * *sm*, struct wpa_ie_data * *data*)

Parse own WPA/RSN IE.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***data*** Pointer to data area for parsing results

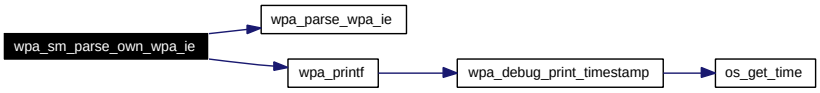
**Returns:**

0 on success, -1 if IE is not known, or -2 on parsing failure

Parse the contents of the own WPA or RSN IE from (Re)AssocReq and write the parsed data into data.

Definition at line 4350 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.12 int wpa_sm_rx_eapol (struct wpa_sm * *sm*, const u8 * *src_addr*, const u8 * *buf*, size_t *len*)

Process received WPA EAPOL frames.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***src_addr*** Source MAC address of the EAPOL packet

***buf*** Pointer to the beginning of the EAPOL data (EAPOL header)

***len*** Length of the EAPOL frame

**Returns:**

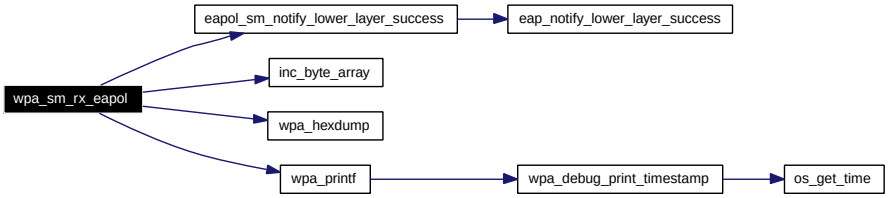
1 = WPA EAPOL-Key processed, 0 = not a WPA EAPOL-Key, -1 failure

This function is called for each received EAPOL frame. Other than EAPOL-Key frames can be skipped if filtering is done elsewhere. wpa_sm_rx_eapol() is only processing WPA and WPA2 EAPOL-Key frames.

The received EAPOL-Key packets are validated and valid packets are replied to. In addition, key material (PTK, GTK) is configured at the end of a successful key handshake.

Definition at line 3403 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.13 int wpa_sm_set_ap_rsn_ie (struct wpa_sm * *sm*, const u8 * *ie*, size_t *len*)

Set AP RSN IE from Beacon/ProbeResp.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***ie*** Pointer to IE data (starting from id)

***len*** IE length

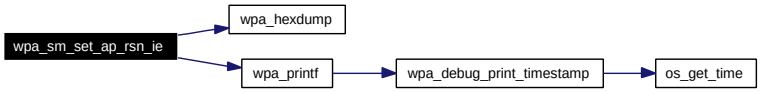
**Returns:**

0 on success, -1 on failure

Inform WPA state machine about the RSN IE used in Beacon / Probe Response frame.

Definition at line 4316 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.14 int wpa_sm_set_ap_wpa_ie (struct wpa_sm * *sm*, const u8 * *ie*, size_t *len*)

Set AP WPA IE from Beacon/ProbeResp.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***ie*** Pointer to IE data (starting from id)

***len*** IE length

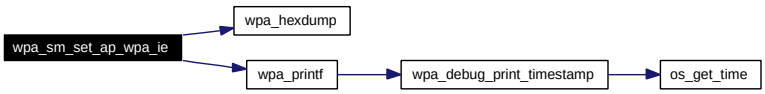
**Returns:**

0 on success, -1 on failure

Inform WPA state machine about the WPA IE used in Beacon / Probe Response frame.

Definition at line 4281 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.15 int wpa_sm_set_assoc_wpa_ie (struct wpa_sm * *sm*, const u8 * *ie*, size_t *len*)

Set own WPA/RSN IE from (Re)AssocReq.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***ie*** Pointer to IE data (starting from id)

***len*** IE length

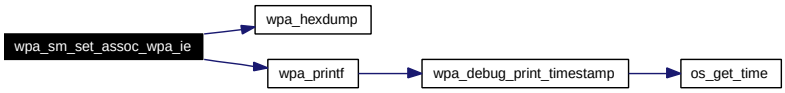
**Returns:**

0 on success, -1 on failure

Inform WPA state machine about the WPA/RSN IE used in (Re)Association Request frame. The IE will be used to override the default value generated with wpa_sm_set_assoc_wpa_ie_default().

Definition at line 4246 of file wpa.c.

Here is the call graph for this function:




#### 6.171.2.16 int wpa_sm_set_assoc_wpa_ie_default (struct wpa_sm * *sm*, u8 * *wpa_ie*, size_t * *wpa_ie_len*)

Generate own WPA/RSN IE from configuration.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***wpa_ie*** Pointer to buffer for WPA/RSN IE

***wpa_ie_len*** Pointer to the length of the wpa_ie buffer

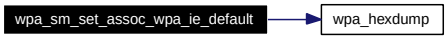
**Returns:**

0 on success, -1 on failure

Inform WPA state machine about the WPA/RSN IE used in (Re)Association Request frame. The IE will be used to override the default value generated with wpa_sm_set_assoc_wpa_ie_default().

Definition at line 4200 of file wpa.c.

Here is the call graph for this function:



#### 6.171.2.17 void wpa_sm_set_config (struct wpa_sm * *sm*, struct wpa_ssid * *config*)

Notification of current configration change.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***config*** Pointer to current network configuration

Notify WPA state machine that configuration has changed. config will be stored as a backpointer to network configuration. This can be NULL to clear the stored pointed.

Definition at line 4006 of file wpa.c.

Here is the call graph for this function:


#### 6.171.2.18 void wpa_sm_set_eapol (struct wpa_sm * *sm*, struct eapol_sm * *eapol*)

Set EAPOL state machine pointer.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***eapol*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Definition at line 4051 of file wpa.c.

#### 6.171.2.19 void wpa_sm_set_fast_reauth (struct wpa_sm * *sm*, int *fast_reauth*)

Set fast reauthentication (EAP) enabled/disabled.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***fast_reauth*** Whether fast reauthentication (EAP) is allowed

Definition at line 3973 of file wpa.c.

#### 6.171.2.20 void wpa_sm_set_ifname (struct wpa_sm * *sm*, const char * *ifname*, const char * *bridge_ifname*)

Set network interface name.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***ifname*** Interface name

***bridge_ifname*** Optional bridge interface name (for pre-auth)

Definition at line 4035 of file wpa.c.

#### 6.171.2.21 void wpa_sm_set_own_addr (struct wpa_sm * *sm*, const u8 * *addr*)

Set own MAC address.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***addr*** Own MAC address

Definition at line 4021 of file wpa.c.

#### 6.171.2.22 int wpa_sm_set_param (struct wpa_sm * *sm*, enum wpa_sm_conf_params *param*, unsigned int *value*)

Set WPA state machine parameters.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***param*** Parameter field

***value*** Parameter value

**Returns:**

0 on success, -1 on failure

Definition at line 4066 of file wpa.c.

#### 6.171.2.23 void wpa_sm_set_pmk (struct wpa_sm * *sm*, const u8 * *pmk*, size_t *pmk_len*)

Set PMK.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***pmk*** The new PMK

***pmk_len*** The length of the new PMK in bytes

Configure the PMK for WPA state machine.

Definition at line 3934 of file wpa.c.

#### 6.171.2.24 void wpa_sm_set_pmk_from_pmksa (struct wpa_sm * *sm*)

Set PMK based on the current PMKSA.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

Take the PMK from the current PMKSA into use. If no PMKSA is active, the PMK will be cleared.

Definition at line 3952 of file wpa.c.

#### 6.171.2.25 void wpa_sm_set_scard_ctx (struct wpa_sm * *sm*, void * *scard_ctx*)

Set context pointer for smartcard callbacks.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***scard_ctx*** Context pointer for smartcard related callback functions

Definition at line 3986 of file wpa.c.

Here is the call graph for this function:


#### 6.171.2.26 int wpa_sm_stkstart (struct wpa_sm * *sm*, const u8 * *peer*)

Send EAPOL-Key Request for STK handshake (STK M1).

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***peer*** MAC address of the peer STA

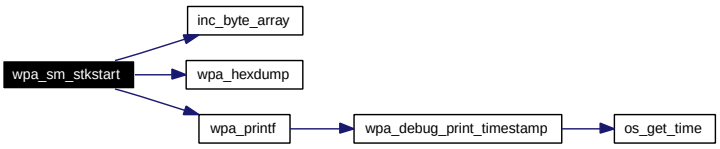
**Returns:**

0 on success, or -1 on failure

Send an EAPOL-Key Request to the current authenticator to start STK handshake with the peer.

Definition at line 1049 of file wpa.c.

Here is the call graph for this function:

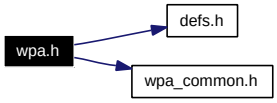


# 6.172 wpa.h File Reference

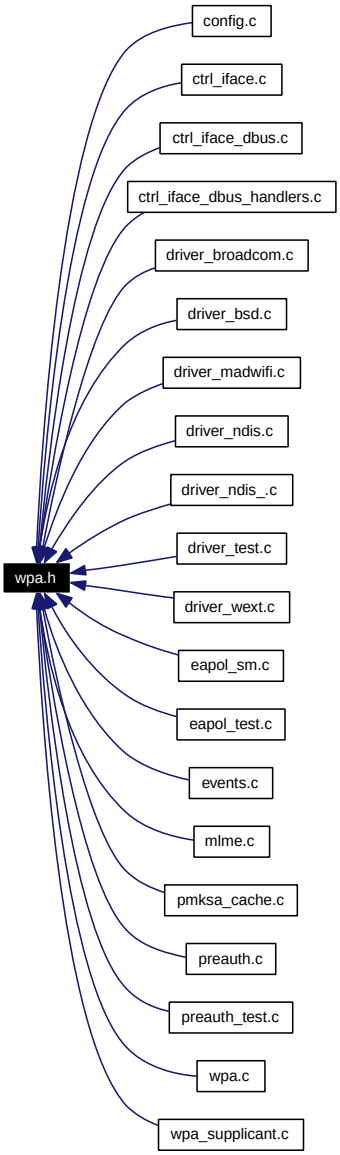
wpa_supplicant - WPA definitions

```
#include "defs.h"
#include "wpa_common.h"
```

Include dependency graph for wpa.h:




This graph shows which files directly or indirectly include this file:



## Defines

- #define **BIT**(n) (1 << (n))
- #define **WPA_CAPABILITY_PREAUTH** BIT(0)
- #define **WPA_CAPABILITY_MGMT_FRAME_PROTECTION** BIT(6)
- #define **WPA_CAPABILITY_PEERKEY_ENABLED** BIT(9)
- #define **GENERIC_INFO_ELEM** 0xdd
- #define **RSN_INFO_ELEM** 0x30
- #define **PMKID_LEN** 16

## Enumerations

- enum {

  **REASON_UNSPECIFIED** = 1, **REASON_DEAUTH_LEAVING** = 3, **REASON_INVALID_IE** = 13, **REASON_MICHAEL_MIC_FAILURE** = 14,

  **REASON_4WAY_HANDSHAKE_TIMEOUT** = 15, **REASON_GROUP_KEY_UPDATE_-TIMEOUT** = 16, **REASON_IE_IN_4WAY_DIFFERS** = 17, **REASON_GROUP_CIPHER_-NOT_VALID** = 18,

  **REASON_PAIRWISE_CIPHER_NOT_VALID** = 19, **REASON_AKMP_NOT_VALID** = 20, **REASON_UNSUPPORTED_RSN_IE_VERSION** = 21, **REASON_INVALID_RSN_IE_-CAPAB** = 22,

  **REASON_IEEE_802_1X_AUTH_FAILED** = 23, **REASON_CIPHER_SUITE_REJECTED** = 24 }
- enum **wpa_sm_conf_params** {

  **RSNA_PMK_LIFETIME**, **RSNA_PMK_REAUTH_THRESHOLD**, **RSNA_SA_TIMEOUT**, **WPA_PARAM_PROTO**,

  **WPA_PARAM_PAIRWISE**, **WPA_PARAM_GROUP**, **WPA_PARAM_KEY_MGMT**, **WPA_-PARAM_MGMT_GROUP** }

## Functions

- wpa_sm * wpa_sm_init (struct wpa_sm_ctx *ctx)

  *Initialize WPA state machine.*

- void wpa_sm_deinit (struct wpa_sm *sm)

  *Deinitialize WPA state machine.*

- void wpa_sm_notify_assoc (struct wpa_sm *sm, const u8 *bssid)

  *Notify WPA state machine about association.*

- void wpa_sm_notify_disassoc (struct wpa_sm *sm)

  *Notify WPA state machine about disassociation.*

- void wpa_sm_set_pmk (struct wpa_sm *sm, const u8 *pmk, size_t pmk_len)

  *Set PMK.*

- void wpa_sm_set_pmk_from_pmksa (struct wpa_sm *sm)

  *Set PMK based on the current PMKSA.*

- void wpa_sm_set_fast_reauth (struct wpa_sm *sm, int fast_reauth)
  *Set fast reauthentication (EAP) enabled/disabled.*
- void wpa_sm_set_scard_ctx (struct wpa_sm *sm, void *scard_ctx)
  *Set context pointer for smartcard callbacks.*
- void wpa_sm_set_config (struct wpa_sm *sm, struct wpa_ssid *config)
  *Notification of current configration change.*
- void wpa_sm_set_own_addr (struct wpa_sm *sm, const u8 *addr)
  *Set own MAC address.*
- void wpa_sm_set_ifname (struct wpa_sm *sm, const char *ifname, const char *bridge_ifname)
  *Set network interface name.*
- void wpa_sm_set_eapol (struct wpa_sm *sm, struct eapol_sm *eapol)
  *Set EAPOL state machine pointer.*
- int wpa_sm_set_assoc_wpa_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  *Set own WPA/RSN IE from (Re)AssocReq.*
- int wpa_sm_set_assoc_wpa_ie_default (struct wpa_sm *sm, u8 *wpa_ie, size_t *wpa_ie_len)
  *Generate own WPA/RSN IE from configuration.*
- int wpa_sm_set_ap_wpa_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  *Set AP WPA IE from Beacon/ProbeResp.*
- int wpa_sm_set_ap_rsn_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  *Set AP RSN IE from Beacon/ProbeResp.*
- int wpa_sm_get_mib (struct wpa_sm *sm, char *buf, size_t buflen)
  *Dump text list of MIB entries.*
- int wpa_sm_set_param (struct wpa_sm *sm, enum wpa_sm_conf_params param, unsigned int value)
  *Set WPA state machine parameters.*
- unsigned int wpa_sm_get_param (struct wpa_sm *sm, enum wpa_sm_conf_params param)
  *Get WPA state machine parameters.*
- int wpa_sm_get_status (struct wpa_sm *sm, char *buf, size_t buflen, int verbose)
  *Get WPA state machine.*
- void wpa_sm_key_request (struct wpa_sm *sm, int error, int pairwise)
  *Send EAPOL-Key Request.*
- int wpa_sm_stkstart (struct wpa_sm *sm, const u8 *peer)
  *Send EAPOL-Key Request for STK handshake (STK M1).*
- int wpa_parse_wpa_ie (const u8 *wpa_ie, size_t wpa_ie_len, struct wpa_ie_data *data)

*Parse WPA/RSN IE.*

- void wpa_sm_aborted_cached (struct wpa_sm *sm)
  *Notify WPA that PMKSA caching was aborted.*
- int wpa_sm_rx_eapol (struct wpa_sm *sm, const u8 *src_addr, const u8 *buf, size_t len)
  *Process received WPA EAPOL frames.*
- int wpa_sm_parse_own_wpa_ie (struct wpa_sm *sm, struct wpa_ie_data *data)
  *Parse own WPA/RSN IE.*

## 6.172.1 Detailed Description

wpa_supplicant - WPA definitions

**Copyright**
Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file wpa.h.

## 6.172.2 Function Documentation

### 6.172.2.1 int wpa_parse_wpa_ie (const u8 * *wpa_ie*, size_t *wpa_ie_len*, struct wpa_ie_data * *data*)

Parse WPA/RSN IE.

**Parameters:**
***wpa_ie*** Pointer to WPA or RSN IE
***wpa_ie_len*** Length of the WPA/RSN IE
***data*** Pointer to data area for parsing results

**Returns:**
0 on success, -1 on failure

Parse the contents of WPA or RSN IE and write the parsed data into data.

Definition at line 647 of file wpa.c.

### 6.172.2.2 void wpa_sm_aborted_cached (struct wpa_sm * *sm*)

Notify WPA that PMKSA caching was aborted.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 3343 of file wpa.c.

Here is the call graph for this function:


#### 6.172.2.3 void wpa_sm_deinit (struct wpa_sm * *sm*)

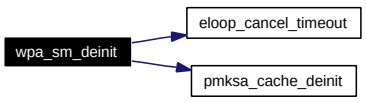
Deinitialize WPA state machine.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

Definition at line 3861 of file wpa.c.

Here is the call graph for this function:




#### 6.172.2.4 int wpa_sm_get_mib (struct wpa_sm * *sm*, char * *buf*, size_t *buflen*)

Dump text list of MIB entries.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***buf*** Buffer for the list

***buflen*** Length of the buffer

**Returns:**

Number of bytes written to buffer

This function is used fetch dot11 MIB variables.

Definition at line 3726 of file wpa.c.

Here is the call graph for this function:


#### 6.172.2.5 unsigned int wpa_sm_get_param (struct wpa_sm * *sm*, enum wpa_sm_conf_params *param*)

Get WPA state machine parameters.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***param*** Parameter field

**Returns:**

Parameter value

Definition at line 4125 of file wpa.c.

#### 6.172.2.6 int wpa_sm_get_status (struct wpa_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get WPA state machine.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***buf*** Buffer for status information

***buflen*** Maximum buffer length

***verbose*** Whether to include verbose status information

**Returns:**

Number of bytes written to buf.

Query WPA state machine for status information. This function fills in a text area with current status information. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

Definition at line 4168 of file wpa.c.

#### 6.172.2.7 struct wpa_sm* wpa_sm_init (struct wpa_sm_ctx * *ctx*)

Initialize WPA state machine.

**Parameters:**

***ctx*** Context pointer for callbacks; this needs to be an allocated buffer

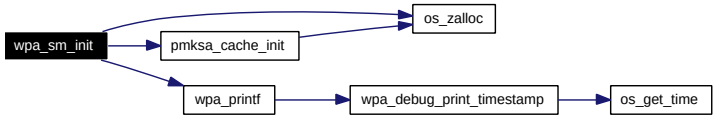
**Returns:**

Pointer to the allocated WPA state machine data

This function is used to allocate a new WPA state machine and the returned value is passed to all WPA state machine calls.

Definition at line 3830 of file wpa.c.

Here is the call graph for this function:



#### 6.172.2.8 void wpa_sm_key_request (struct wpa_sm * *sm*, int *error*, int *pairwise*)

Send EAPOL-Key Request.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***error*** Indicate whether this is an Michael MIC error report

***pairwise*** 1 = error report for pairwise packet, 0 = for group packet

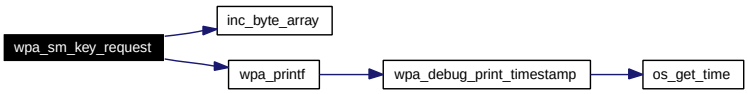
**Returns:**

Pointer to the current network structure or NULL on failure

Send an EAPOL-Key Request to the current authenticator. This function is used to request rekeying and it is usually called when a local Michael MIC failure is detected.

Definition at line 990 of file wpa.c.

Here is the call graph for this function:




#### 6.172.2.9 void wpa_sm_notify_assoc (struct wpa_sm * *sm*, const u8 * *bssid*)

Notify WPA state machine about association.

**Parameters:**

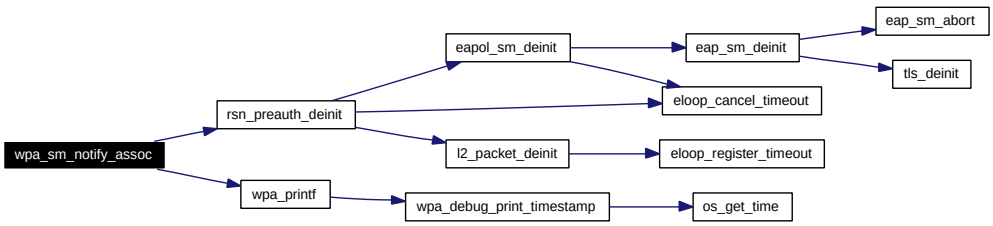
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***bssid*** The BSSID of the new association

This function is called to let WPA state machine know that the connection was established.

Definition at line 3894 of file wpa.c.

Here is the call graph for this function:




#### 6.172.2.10 void wpa_sm_notify_disassoc (struct wpa_sm * *sm*)

Notify WPA state machine about disassociation.

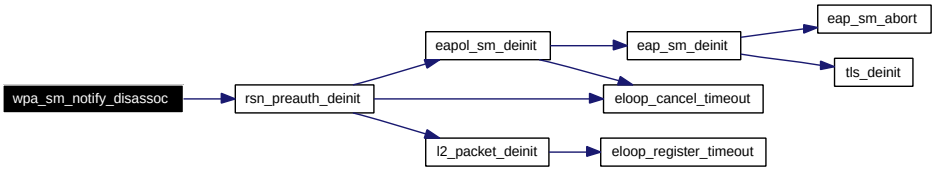
**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

This function is called to let WPA state machine know that the connection was lost. This will abort any existing pre-authentication session.

Definition at line 3917 of file wpa.c.

Here is the call graph for this function:




#### 6.172.2.11 int wpa_sm_parse_own_wpa_ie (struct wpa_sm * *sm*, struct wpa_ie_data * *data*)

Parse own WPA/RSN IE.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***data*** Pointer to data area for parsing results

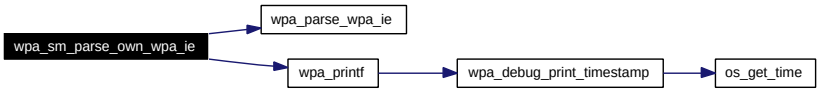
**Returns:**

0 on success, -1 if IE is not known, or -2 on parsing failure

Parse the contents of the own WPA or RSN IE from (Re)AssocReq and write the parsed data into data.

Definition at line 4350 of file wpa.c.

Here is the call graph for this function:




#### 6.172.2.12 int wpa_sm_rx_eapol (struct wpa_sm * *sm*, const u8 * *src_addr*, const u8 * *buf*, size_t *len*)

Process received WPA EAPOL frames.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***src_addr*** Source MAC address of the EAPOL packet

***buf*** Pointer to the beginning of the EAPOL data (EAPOL header)

***len*** Length of the EAPOL frame

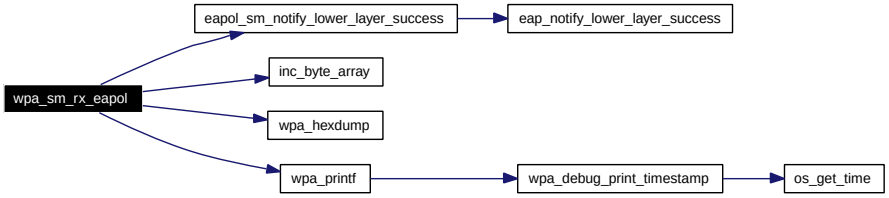
**Returns:**
1 = WPA EAPOL-Key processed, 0 = not a WPA EAPOL-Key, -1 failure

This function is called for each received EAPOL frame. Other than EAPOL-Key frames can be skipped if filtering is done elsewhere. wpa_sm_rx_eapol() is only processing WPA and WPA2 EAPOL-Key frames.

The received EAPOL-Key packets are validated and valid packets are replied to. In addition, key material (PTK, GTK) is configured at the end of a successful key handshake.

Definition at line 3403 of file wpa.c.

Here is the call graph for this function:




### 6.172.2.13 int wpa_sm_set_ap_rsn_ie (struct wpa_sm * *sm*, const u8 * *ie*, size_t *len*)

Set AP RSN IE from Beacon/ProbeResp.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***ie*** Pointer to IE data (starting from id)

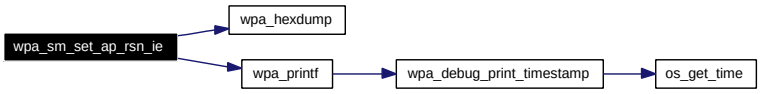
***len*** IE length

**Returns:**
0 on success, -1 on failure

Inform WPA state machine about the RSN IE used in Beacon / Probe Response frame.

Definition at line 4316 of file wpa.c.

Here is the call graph for this function:




### 6.172.2.14 int wpa_sm_set_ap_wpa_ie (struct wpa_sm * *sm*, const u8 * *ie*, size_t *len*)

Set AP WPA IE from Beacon/ProbeResp.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***ie*** Pointer to IE data (starting from id)

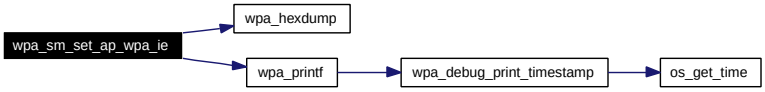
***len*** IE length

**Returns:**
0 on success, -1 on failure

Inform WPA state machine about the WPA IE used in Beacon / Probe Response frame.

Definition at line 4281 of file wpa.c.

Here is the call graph for this function:




#### 6.172.2.15 int wpa_sm_set_assoc_wpa_ie (struct wpa_sm * *sm*, const u8 * *ie*, size_t *len*)

Set own WPA/RSN IE from (Re)AssocReq.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***ie*** Pointer to IE data (starting from id)

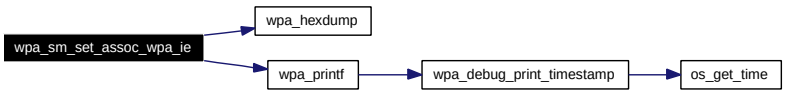
***len*** IE length

**Returns:**
0 on success, -1 on failure

Inform WPA state machine about the WPA/RSN IE used in (Re)Association Request frame. The IE will be used to override the default value generated with wpa_sm_set_assoc_wpa_ie_default().

Definition at line 4246 of file wpa.c.

Here is the call graph for this function:




#### 6.172.2.16 int wpa_sm_set_assoc_wpa_ie_default (struct wpa_sm * *sm*, u8 * *wpa_ie*, size_t * *wpa_ie_len*)

Generate own WPA/RSN IE from configuration.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()

***wpa_ie*** Pointer to buffer for WPA/RSN IE

***wpa_ie_len*** Pointer to the length of the wpa_ie buffer

**Returns:**
0 on success, -1 on failure

Inform WPA state machine about the WPA/RSN IE used in (Re)Association Request frame. The IE will be used to override the default value generated with wpa_sm_set_assoc_wpa_ie_default().

Definition at line 4200 of file wpa.c.

Here is the call graph for this function:


### 6.172.2.17 void wpa_sm_set_config (struct wpa_sm * *sm*, struct wpa_ssid * *config*)

Notification of current configration change.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()
***config*** Pointer to current network configuration

Notify WPA state machine that configuration has changed. config will be stored as a backpointer to network configuration. This can be NULL to clear the stored pointed.

Definition at line 4006 of file wpa.c.

Here is the call graph for this function:


### 6.172.2.18 void wpa_sm_set_eapol (struct wpa_sm * *sm*, struct eapol_sm * *eapol*)

Set EAPOL state machine pointer.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()
***eapol*** Pointer to EAPOL state machine allocated with eapol_sm_init()

Definition at line 4051 of file wpa.c.

### 6.172.2.19 void wpa_sm_set_fast_reauth (struct wpa_sm * *sm*, int *fast_reauth*)

Set fast reauthentication (EAP) enabled/disabled.

**Parameters:**
***sm*** Pointer to WPA state machine data from wpa_sm_init()
***fast_reauth*** Whether fast reauthentication (EAP) is allowed

Definition at line 3973 of file wpa.c.

#### 6.172.2.20 void wpa_sm_set_ifname (struct wpa_sm * *sm*, const char * *ifname*, const char * *bridge_ifname*)

Set network interface name.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***ifname*** Interface name

***bridge_ifname*** Optional bridge interface name (for pre-auth)

Definition at line 4035 of file wpa.c.

#### 6.172.2.21 void wpa_sm_set_own_addr (struct wpa_sm * *sm*, const u8 * *addr*)

Set own MAC address.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***addr*** Own MAC address

Definition at line 4021 of file wpa.c.

#### 6.172.2.22 int wpa_sm_set_param (struct wpa_sm * *sm*, enum wpa_sm_conf_params *param*, unsigned int *value*)

Set WPA state machine parameters.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***param*** Parameter field

***value*** Parameter value

**Returns:**

0 on success, -1 on failure

Definition at line 4066 of file wpa.c.

#### 6.172.2.23 void wpa_sm_set_pmk (struct wpa_sm * *sm*, const u8 * *pmk*, size_t *pmk_len*)

Set PMK.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***pmk*** The new PMK

***pmk_len*** The length of the new PMK in bytes

Configure the PMK for WPA state machine.

Definition at line 3934 of file wpa.c.

#### 6.172.2.24 void wpa_sm_set_pmk_from_pmksa (struct wpa_sm * *sm*)

Set PMK based on the current PMKSA.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

Take the PMK from the current PMKSA into use. If no PMKSA is active, the PMK will be cleared.

Definition at line 3952 of file wpa.c.

#### 6.172.2.25 void wpa_sm_set_scard_ctx (struct wpa_sm * *sm*, void * *scard_ctx*)

Set context pointer for smartcard callbacks.

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***scard_ctx*** Context pointer for smartcard related callback functions

Definition at line 3986 of file wpa.c.

Here is the call graph for this function:


#### 6.172.2.26 int wpa_sm_stkstart (struct wpa_sm * *sm*, const u8 * *peer*)

Send EAPOL-Key Request for STK handshake (STK M1).

**Parameters:**

***sm*** Pointer to WPA state machine data from wpa_sm_init()

***peer*** MAC address of the peer STA

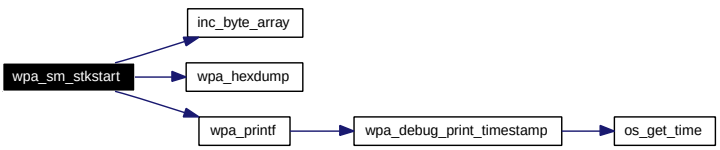
**Returns:**

0 on success, or -1 on failure

Send an EAPOL-Key Request to the current authenticator to start STK handshake with the peer.

Definition at line 1049 of file wpa.c.

Here is the call graph for this function:

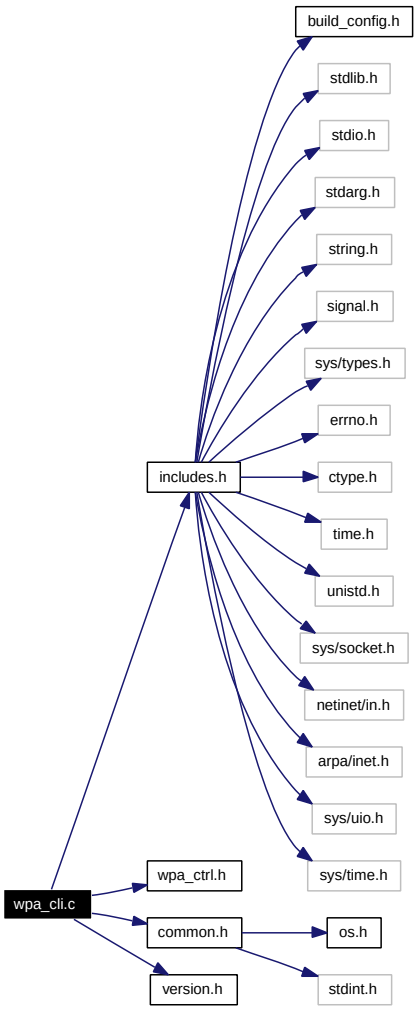


## 6.173 wpa_cli.c File Reference

WPA Supplicant - command line interface for wpa_supplicant daemon.

```
#include "includes.h"
#include "wpa_ctrl.h"
#include "common.h"
#include "version.h"
```

Include dependency graph for wpa_cli.c:




### Defines

- #define **max_args** 10

### Functions

- int **main** (int argc, char *argv[ ])

### 6.173.1 Detailed Description

WPA Supplicant - command line interface for wpa_supplicant daemon.

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

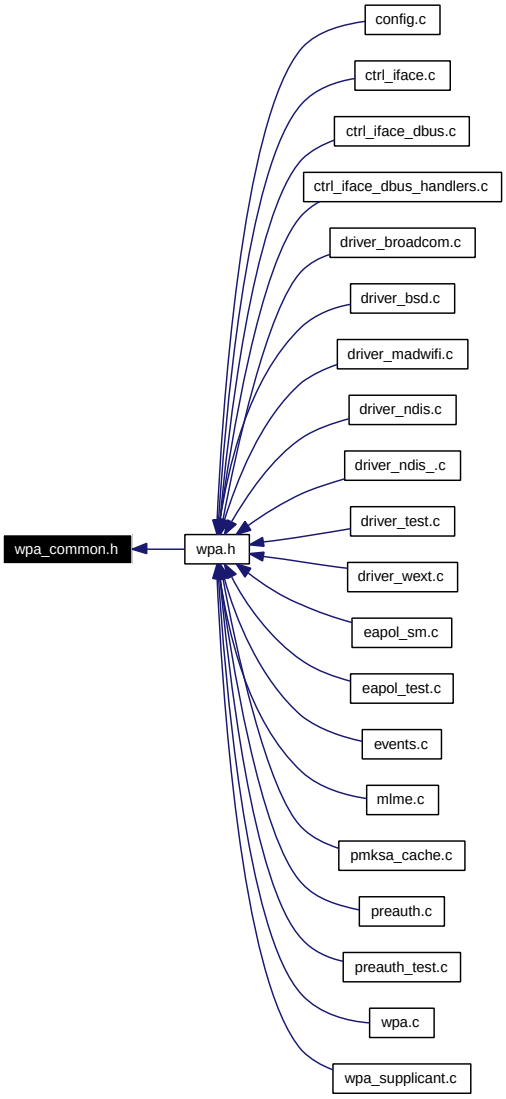
See README and COPYING for more details.

Definition in file wpa_cli.c.

## 6.174 wpa_common.h File Reference

WPA definitions shared between hostapd and wpa_supplicant.

This graph shows which files directly or indirectly include this file:




### Defines

- #define **WPA_REPLAY_COUNTER_LEN** 8
- #define **WPA_NONCE_LEN** 32
- #define **WPA_KEY_RSC_LEN** 8
- #define **EAPOL_VERSION** 2

### Enumerations

- enum {

**IEEE802_1X_TYPE_EAP_PACKET** = 0, **IEEE802_1X_TYPE_EAPOL_START** = 1, **IEEE802_1X_TYPE_EAPOL_LOGOFF** = 2, **IEEE802_1X_TYPE_EAPOL_KEY** = 3, **IEEE802_1X_TYPE_EAPOL_ENCAPSULATED_ASF_ALERT** = 4 }

- enum { **EAPOL_KEY_TYPE_RC4** = 1, **EAPOL_KEY_TYPE_RSN** = 2, **EAPOL_KEY_TYPE_WPA** = 254 }

## Variables

- ieee802_1x_hdr **STRUCT_PACKED**

### 6.174.1 Detailed Description

WPA definitions shared between hostapd and wpa_supplicant.

**Copyright**

Copyright (c) 2002-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

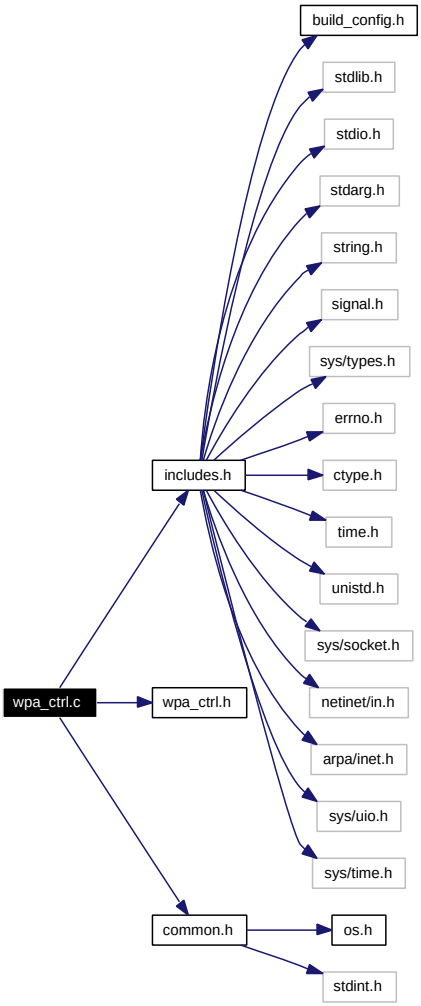
Definition in file wpa_common.h.

# 6.175 wpa_ctrl.c File Reference

wpa_supplicant/hostapd control interface library

```
#include "includes.h"
#include "wpa_ctrl.h"
#include "common.h"
```

Include dependency graph for wpa_ctrl.c:




## Functions

- int wpa_ctrl_attach (struct wpa_ctrl *ctrl)

  *Register as an event monitor for the control interface.*

- int wpa_ctrl_detach (struct wpa_ctrl *ctrl)

  *Unregister event monitor from the control interface.*

## 6.175.1 Detailed Description

wpa_supplicant/hostapd control interface library

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file wpa_ctrl.c.

## 6.175.2 Function Documentation

### 6.175.2.1 int wpa_ctrl_attach (struct wpa_ctrl * *ctrl*)

Register as an event monitor for the control interface.

**Parameters:**
***ctrl*** Control interface data from wpa_ctrl_open()

**Returns:**
0 on success, -1 on failure, -2 on timeout

This function registers the control interface connection as a monitor for wpa_supplicant/hostapd events. After a success wpa_ctrl_attach() call, the control interface connection starts receiving event messages that can be read with wpa_ctrl_recv().

Definition at line 264 of file wpa_ctrl.c.

### 6.175.2.2 int wpa_ctrl_detach (struct wpa_ctrl * *ctrl*)

Unregister event monitor from the control interface.

**Parameters:**
***ctrl*** Control interface data from wpa_ctrl_open()

**Returns:**
0 on success, -1 on failure, -2 on timeout

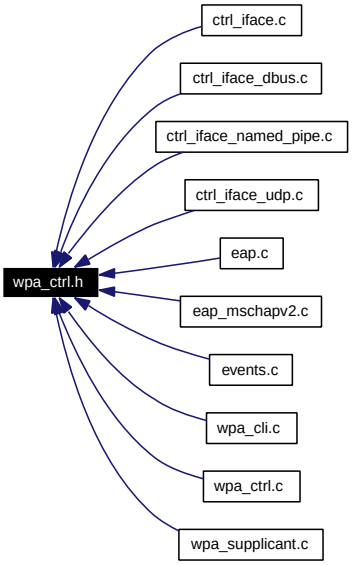
This function unregisters the control interface connection as a monitor for wpa_supplicant/hostapd events, i.e., cancels the registration done with wpa_ctrl_attach().

Definition at line 270 of file wpa_ctrl.c.

# 6.176 wpa_ctrl.h File Reference

wpa_supplicant/hostapd control interface library

This graph shows which files directly or indirectly include this file:




## Defines

- #define WPA_CTRL_REQ "CTRL-REQ-"
- #define WPA_CTRL_RSP "CTRL-RSP-"
- #define WPA_EVENT_CONNECTED "CTRL-EVENT-CONNECTED "
- #define WPA_EVENT_DISCONNECTED "CTRL-EVENT-DISCONNECTED "
- #define WPA_EVENT_TERMINATING "CTRL-EVENT-TERMINATING "
- #define WPA_EVENT_PASSWORD_CHANGED "CTRL-EVENT-PASSWORD-CHANGED "
- #define WPA_EVENT_EAP_NOTIFICATION "CTRL-EVENT-EAP-NOTIFICATION "
- #define WPA_EVENT_EAP_STARTED "CTRL-EVENT-EAP-STARTED "
- #define WPA_EVENT_EAP_METHOD "CTRL-EVENT-EAP-METHOD "
- #define WPA_EVENT_EAP_SUCCESS "CTRL-EVENT-EAP-SUCCESS "
- #define WPA_EVENT_EAP_FAILURE "CTRL-EVENT-EAP-FAILURE "

## Functions

- wpa_ctrl ∗ wpa_ctrl_open (const char ∗ctrl_path)

  *Open a control interface to wpa_supplicant/hostapd.*

- void wpa_ctrl_close (struct wpa_ctrl ∗ctrl)

  *Close a control interface to wpa_supplicant/hostapd.*

- int wpa_ctrl_request (struct wpa_ctrl ∗ctrl, const char ∗cmd, size_t cmd_len, char ∗reply, size_t ∗reply_len, void(∗msg_cb)(char ∗msg, size_t len))

  *Send a command to wpa_supplicant/hostapd.*

- int wpa_ctrl_attach (struct wpa_ctrl *ctrl)
  *Register as an event monitor for the control interface.*
- int wpa_ctrl_detach (struct wpa_ctrl *ctrl)
  *Unregister event monitor from the control interface.*
- int wpa_ctrl_recv (struct wpa_ctrl *ctrl, char *reply, size_t *reply_len)
  *Receive a pending control interface message.*
- int wpa_ctrl_pending (struct wpa_ctrl *ctrl)
  *Check whether there are pending event messages.*
- int wpa_ctrl_get_fd (struct wpa_ctrl *ctrl)
  *Get file descriptor used by the control interface.*

## 6.176.1 Detailed Description

wpa_supplicant/hostapd control interface library

**Copyright**
Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file wpa_ctrl.h.

## 6.176.2 Define Documentation

### 6.176.2.1 #define WPA_CTRL_REQ "CTRL-REQ-"

Interactive request for identity/password/pin

Definition at line 26 of file wpa_ctrl.h.

### 6.176.2.2 #define WPA_CTRL_RSP "CTRL-RSP-"

Response to identity/password/pin request

Definition at line 29 of file wpa_ctrl.h.

### 6.176.2.3 #define WPA_EVENT_CONNECTED "CTRL-EVENT-CONNECTED "

Authentication completed successfully and data connection enabled

Definition at line 33 of file wpa_ctrl.h.

#### 6.176.2.4 #define WPA_EVENT_DISCONNECTED "CTRL-EVENT-DISCONNECTED "

Disconnected, data connection is not available

Definition at line 35 of file wpa_ctrl.h.

#### 6.176.2.5 #define WPA_EVENT_EAP_FAILURE "CTRL-EVENT-EAP-FAILURE "

EAP authentication failed (EAP-Failure received)

Definition at line 49 of file wpa_ctrl.h.

#### 6.176.2.6 #define WPA_EVENT_EAP_METHOD "CTRL-EVENT-EAP-METHOD "

EAP method selected

Definition at line 45 of file wpa_ctrl.h.

#### 6.176.2.7 #define WPA_EVENT_EAP_NOTIFICATION "CTRL-EVENT-EAP-NOTIFICATION "

EAP-Request/Notification received

Definition at line 41 of file wpa_ctrl.h.

#### 6.176.2.8 #define WPA_EVENT_EAP_STARTED "CTRL-EVENT-EAP-STARTED "

EAP authentication started (EAP-Request/Identity received)

Definition at line 43 of file wpa_ctrl.h.

#### 6.176.2.9 #define WPA_EVENT_EAP_SUCCESS "CTRL-EVENT-EAP-SUCCESS "

EAP authentication completed successfully

Definition at line 47 of file wpa_ctrl.h.

#### 6.176.2.10 #define WPA_EVENT_PASSWORD_CHANGED "CTRL-EVENT-PASSWORD-CHANGED "

Password change was completed successfully

Definition at line 39 of file wpa_ctrl.h.

#### 6.176.2.11 #define WPA_EVENT_TERMINATING "CTRL-EVENT-TERMINATING "

wpa_supplicant is exiting

Definition at line 37 of file wpa_ctrl.h.

## 6.176.3 Function Documentation

### 6.176.3.1 int wpa_ctrl_attach (struct wpa_ctrl * *ctrl*)

Register as an event monitor for the control interface.

**Parameters:**
***ctrl*** Control interface data from wpa_ctrl_open()

**Returns:**
0 on success, -1 on failure, -2 on timeout

This function registers the control interface connection as a monitor for wpa_supplicant/hostapd events. After a success wpa_ctrl_attach() call, the control interface connection starts receiving event messages that can be read with wpa_ctrl_recv().

Definition at line 264 of file wpa_ctrl.c.

### 6.176.3.2 void wpa_ctrl_close (struct wpa_ctrl * *ctrl*)

Close a control interface to wpa_supplicant/hostapd.

**Parameters:**
***ctrl*** Control interface data from wpa_ctrl_open()

This function is used to close a control interface.

### 6.176.3.3 int wpa_ctrl_detach (struct wpa_ctrl * *ctrl*)

Unregister event monitor from the control interface.

**Parameters:**
***ctrl*** Control interface data from wpa_ctrl_open()

**Returns:**
0 on success, -1 on failure, -2 on timeout

This function unregisters the control interface connection as a monitor for wpa_supplicant/hostapd events, i.e., cancels the registration done with wpa_ctrl_attach().

Definition at line 270 of file wpa_ctrl.c.

### 6.176.3.4 int wpa_ctrl_get_fd (struct wpa_ctrl * *ctrl*)

Get file descriptor used by the control interface.

**Parameters:**
***ctrl*** Control interface data from wpa_ctrl_open()

**Returns:**
File descriptor used for the connection

This function can be used to get the file descriptor that is used for the control interface connection. The returned value can be used, e.g., with select() while waiting for multiple events.

The returned file descriptor must not be used directly for sending or receiving packets; instead, the library functions wpa_ctrl_request() and wpa_ctrl_recv() must be used for this.

#### 6.176.3.5 struct wpa_ctrl* wpa_ctrl_open (const char * *ctrl_path*)

Open a control interface to wpa_supplicant/hostapd.

**Parameters:**

***ctrl_path*** Path for UNIX domain sockets; ignored if UDP sockets are used.

**Returns:**

Pointer to abstract control interface data or NULL on failure

This function is used to open a control interface to wpa_supplicant/hostapd. ctrl_path is usually /var/run/wpa_supplicant or /var/run/hostapd. This path is configured in wpa_supplicant/hostapd and other programs using the control interface need to use matching path configuration.

#### 6.176.3.6 int wpa_ctrl_pending (struct wpa_ctrl * *ctrl*)

Check whether there are pending event messages.

**Parameters:**

***ctrl*** Control interface data from wpa_ctrl_open()

**Returns:**

1 if there are pending messages, 0 if no, or -1 on error

This function will check whether there are any pending control interface message available to be received with wpa_ctrl_recv(). wpa_ctrl_pending() is only used for event messages, i.e., wpa_ctrl_attach() must have been used to register the control interface as an event monitor.

#### 6.176.3.7 int wpa_ctrl_recv (struct wpa_ctrl * *ctrl*, char * *reply*, size_t * *reply_len*)

Receive a pending control interface message.

**Parameters:**

***ctrl*** Control interface data from wpa_ctrl_open()

***reply*** Buffer for the message data

***reply_len*** Length of the reply buffer

**Returns:**

0 on success, -1 on failure

This function will receive a pending control interface message. This function will block if no messages are available. The received response will be written to reply and reply_len is set to the actual length of the reply. wpa_ctrl_recv() is only used for event messages, i.e., wpa_ctrl_attach() must have been used to register the control interface as an event monitor.

#### 6.176.3.8 int wpa_ctrl_request (struct wpa_ctrl * *ctrl*, const char * *cmd*, size_t *cmd_len*, char * *reply*, size_t * *reply_len*, void(*)(char *msg, size_t len) *msg_cb*)

Send a command to wpa_supplicant/hostapd.

**Parameters:**

***ctrl*** Control interface data from wpa_ctrl_open()

***cmd*** Command; usually, ASCII text, e.g., "PING"

***cmd_len*** Length of the cmd in bytes

***reply*** Buffer for the response

***reply_len*** Reply buffer length

***msg_cb*** Callback function for unsolicited messages or NULL if not used

**Returns:**

0 on success, -1 on error (send or receive failed), -2 on timeout

This function is used to send commands to wpa_supplicant/hostapd. Received response will be written to reply and reply_len is set to the actual length of the reply. This function will block for up to two seconds while waiting for the reply. If unsolicited messages are received, the blocking time may be longer.

msg_cb can be used to register a callback function that will be called for unsolicited messages received while waiting for the command response. These messages may be received if wpa_ctrl_request() is called at the same time as wpa_supplicant/hostapd is sending such a message. This can happen only if the program has used wpa_ctrl_attach() to register itself as a monitor for event messages. Alternatively to msg_cb, programs can register two control interface connections and use one of them for commands and the other one for receiving event messages, in other words, call wpa_ctrl_attach() only for the control interface connection that will be used for event messages.

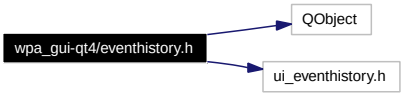
## 6.177 wpa_gui-qt4/eventhistory.h File Reference

wpa_gui - EventHistory class

```
#include <QObject>
```

```
#include "ui_eventhistory.h"
```

Include dependency graph for eventhistory.h:




### 6.177.1 Detailed Description

wpa_gui - EventHistory class

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file eventhistory.h.

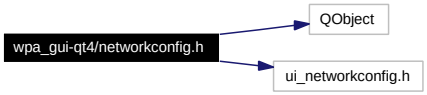
# 6.178 wpa_gui-qt4/networkconfig.h File Reference

wpa_gui - NetworkConfig class

```
#include <QObject>
```

```
#include "ui_networkconfig.h"
```

Include dependency graph for networkconfig.h:




## 6.178.1 Detailed Description

wpa_gui - NetworkConfig class

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file networkconfig.h.

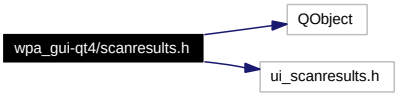
## 6.179 wpa_gui-qt4/scanresults.h File Reference

wpa_gui - ScanResults class

```
#include <QObject>
```

```
#include "ui_scanresults.h"
```

Include dependency graph for scanresults.h:




### 6.179.1 Detailed Description

wpa_gui - ScanResults class

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file scanresults.h.

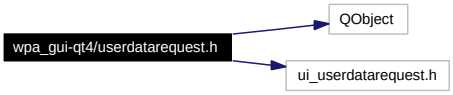
# 6.180 wpa_gui-qt4/userdatarequest.h File Reference

wpa_gui - UserDataRequest class

```
#include <QObject>
```

```
#include "ui_userdatarequest.h"
```

Include dependency graph for userdatarequest.h:




## 6.180.1 Detailed Description

wpa_gui - UserDataRequest class

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

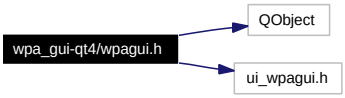
Definition in file userdatarequest.h.

## 6.181 wpa_gui-qt4/wpagui.h File Reference

wpa_gui - WpaGui class

```
#include <QObject>
#include "ui_wpagui.h"
```

Include dependency graph for wpagui.h:




### 6.181.1 Detailed Description

wpa_gui - WpaGui class

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file wpagui.h.

# 6.182 wpa_gui-qt4/wpamsg.h File Reference

wpa_gui - WpaMsg class for storing event messages

```
#include <QDateTime>
#include <QLinkedList>
```

Include dependency graph for wpamsg.h:


## Data Structures

- class **WpaMsg**

## Typedefs

- typedef QLinkedList< WpaMsg > **WpaMsgList**

### 6.182.1 Detailed Description

wpa_gui - WpaMsg class for storing event messages

**Copyright**
Copyright (c) 2005-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

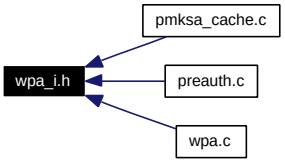
See README and COPYING for more details.

Definition in file wpamsg.h.

## 6.183 wpa_i.h File Reference

wpa_supplicant - Internal WPA state machine definitions

This graph shows which files directly or indirectly include this file:




### Variables

- wpa_ptk STRUCT_PACKED

  *WPA Pairwise Transient Key.*

### 6.183.1 Detailed Description

wpa_supplicant - Internal WPA state machine definitions

**Copyright**

Copyright (c) 2004-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file wpa_i.h.

### 6.183.2 Variable Documentation

#### 6.183.2.1 struct wpa_ptk STRUCT_PACKED

WPA Pairwise Transient Key.

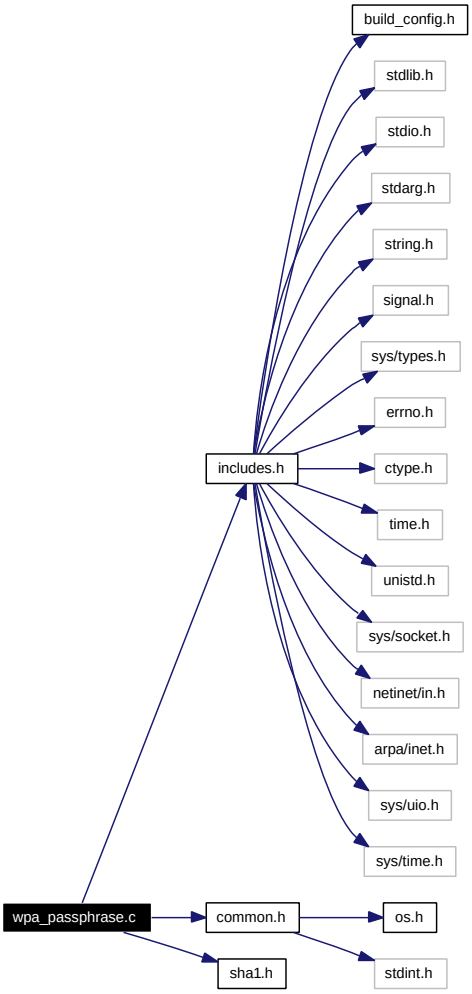
IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy

## 6.184 wpa_passphrase.c File Reference

WPA Supplicant - ASCII passphrase to WPA PSK tool.

```
#include "includes.h"
#include "common.h"
#include "sha1.h"
```

Include dependency graph for wpa_passphrase.c:




### Functions

- int **main** (int argc, char ∗argv[ ])

### 6.184.1 Detailed Description

WPA Supplicant - ASCII passphrase to WPA PSK tool.

**Copyright**



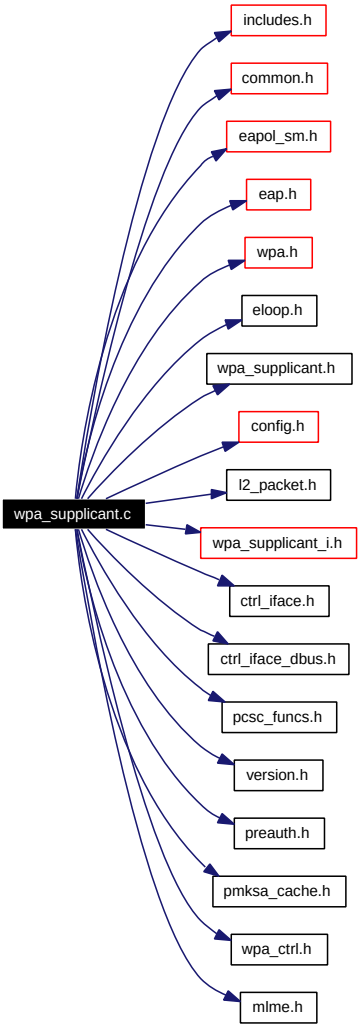
Definition in file wpa_passphrase.c.

## 6.185 wpa_supplicant.c File Reference

WPA Supplicant.

```
#include "includes.h"
#include "common.h"
#include "eapol_sm.h"
#include "eap.h"
#include "wpa.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "config.h"
#include "l2_packet.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface.h"
#include "ctrl_iface_dbus.h"
#include "pcsc_funcs.h"
#include "version.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_ctrl.h"
#include "mlme.h"
```

Include dependency graph for wpa_supplicant.c:




## Defines

- #define **SCAN_AP_LIMIT** 128

## Functions

- wpa_blacklist * wpa_blacklist_get (struct wpa_supplicant *wpa_s, const u8 *bssid)

  *Get the blacklist entry for a BSSID.*

- int wpa_blacklist_add (struct wpa_supplicant *wpa_s, const u8 *bssid)

  *Add an BSSID to the blacklist.*

- void wpa_blacklist_clear (struct wpa_supplicant *wpa_s)

  *Clear the blacklist of all entries.*

- void wpa_supplicant_req_scan (struct wpa_supplicant *wpa_s, int sec, int usec)

  *Schedule a scan for neighboring access points.*

- void wpa_supplicant_cancel_scan (struct wpa_supplicant *wpa_s)
  *Cancel a scheduled scan request.*
- void wpa_supplicant_req_auth_timeout (struct wpa_supplicant *wpa_s, int sec, int usec)
  *Schedule a timeout for authentication.*
- void wpa_supplicant_cancel_auth_timeout (struct wpa_supplicant *wpa_s)
  *Cancel authentication timeout.*
- void wpa_supplicant_initiate_eapol (struct wpa_supplicant *wpa_s)
  *Configure EAPOL state machine.*
- void wpa_supplicant_set_non_wpa_policy (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
  *Set WPA parameters to non-WPA mode.*
- void wpa_clear_keys (struct wpa_supplicant *wpa_s, const u8 *addr)
  *Clear keys configured for the driver.*
- const char * wpa_supplicant_state_txt (int state)
  *Get the connection state name as a text string.*
- void wpa_supplicant_set_state (struct wpa_supplicant *wpa_s, wpa_states state)
  *Set current connection state.*
- wpa_states wpa_supplicant_get_state (struct wpa_supplicant *wpa_s)
  *Get the connection state.*
- int wpa_supplicant_reload_configuration (struct wpa_supplicant *wpa_s)
  *Reload configuration data.*
- int wpa_supplicant_set_suites (struct wpa_supplicant *wpa_s, struct wpa_scan_result *bss, struct wpa_ssid *ssid, u8 *wpa_ie, size_t *wpa_ie_len)
  *Set authentication and encryption parameters.*
- void wpa_supplicant_associate (struct wpa_supplicant *wpa_s, struct wpa_scan_result *bss, struct wpa_ssid *ssid)
  *Request association.*
- void wpa_supplicant_disassociate (struct wpa_supplicant *wpa_s, int reason_code)
  *Disassociate the current connection.*
- void wpa_supplicant_deauthenticate (struct wpa_supplicant *wpa_s, int reason_code)
  *Deauthenticate the current connection.*
- int wpa_supplicant_get_scan_results (struct wpa_supplicant *wpa_s)
  *Get scan results.*
- wpa_ssid * wpa_supplicant_get_ssid (struct wpa_supplicant *wpa_s)
  *Get a pointer to the current network structure.*

- void wpa_supplicant_rx_eapol (void *ctx, const u8 *src_addr, const u8 *buf, size_t len)

  *Deliver a received EAPOL frame to wpa_supplicant.*

- int wpa_supplicant_driver_init (struct wpa_supplicant *wpa_s, int wait_for_interface)

  *Initialize driver interface parameters.*

- wpa_supplicant * wpa_supplicant_add_iface (struct wpa_global *global, struct wpa_interface *iface)

  *Add a new network interface.*

- int wpa_supplicant_remove_iface (struct wpa_global *global, struct wpa_supplicant *wpa_s)

  *Remove a network interface.*

- wpa_supplicant * wpa_supplicant_get_iface (struct wpa_global *global, const char *ifname)

  *Get a new network interface.*

- wpa_global * wpa_supplicant_init (struct wpa_params *params)

  *Initialize wpa_supplicant.*

- int wpa_supplicant_run (struct wpa_global *global)

  *Run the wpa_supplicant main event loop.*

- void wpa_supplicant_deinit (struct wpa_global *global)

  *Deinitialize wpa_supplicant.*

## Variables

- const char * **wpa_supplicant_version**
- const char * **wpa_supplicant_license**
- const char * **wpa_supplicant_full_license1**
- const char * **wpa_supplicant_full_license2**
- const char * **wpa_supplicant_full_license3**
- const char * **wpa_supplicant_full_license4**
- const char * **wpa_supplicant_full_license5**
- wpa_driver_ops * **wpa_supplicant_drivers** [ ]
- int **wpa_debug_use_file**
- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- int **wpa_debug_timestamp**

### 6.185.1 Detailed Description

WPA Supplicant.

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements functions for registering and unregistering wpa_supplicant interfaces. In addition, this file contains number of functions for managing network connections.

Definition in file wpa_supplicant.c.

## 6.185.2 Function Documentation

### 6.185.2.1 int wpa_blacklist_add (struct wpa_supplicant * *wpa_s*, const u8 * *bssid*)

Add an BSSID to the blacklist.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***bssid*** BSSID to be added to the blacklist

**Returns:**

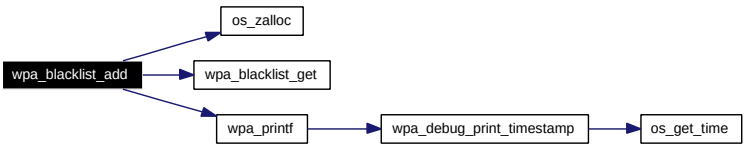
0 on success, -1 on failure

This function adds the specified BSSID to the blacklist or increases the blacklist count if the BSSID was already listed. It should be called when an association attempt fails either due to the selected BSS rejecting association or due to timeout.

This blacklist is used to force wpa_supplicant to go through all available BSSes before retrying to associate with an BSS that rejected or timed out association. It does not prevent the listed BSS from being used; it only changes the order in which they are tried.

Definition at line 418 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.2 void wpa_blacklist_clear (struct wpa_supplicant * *wpa_s*)

Clear the blacklist of all entries.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

Definition at line 474 of file wpa_supplicant.c.

Here is the call graph for this function:


#### 6.185.2.3 struct wpa_blacklist* wpa_blacklist_get (struct wpa_supplicant * *wpa_s*, const u8 * *bssid*)

Get the blacklist entry for a BSSID.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***bssid*** BSSID

**Returns:**

Matching blacklist entry for the BSSID or NULL if not found

Definition at line 385 of file wpa_supplicant.c.

#### 6.185.2.4 void wpa_clear_keys (struct wpa_supplicant * *wpa_s*, const u8 * *addr*)

Clear keys configured for the driver.

**Parameters:**

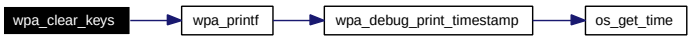
***wpa_s*** Pointer to wpa_supplicant data

***addr*** Previously used BSSID or NULL if not available

This function clears the encryption keys that has been previously configured for the driver.

Definition at line 739 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.185.2.5 struct wpa_supplicant* wpa_supplicant_add_iface (struct wpa_global * *global*, struct wpa_interface * *iface*)

Add a new network interface.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

***iface*** Interface configuration options

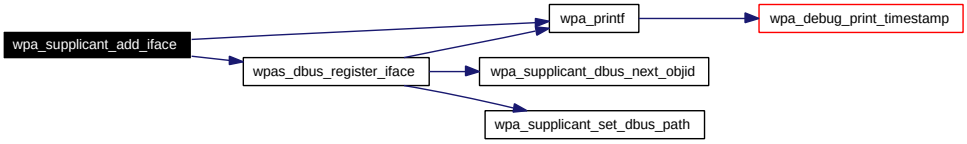
**Returns:**

Pointer to the created interface or NULL on failure

This function is used to add new network interfaces for wpa_supplicant. This can be called before wpa_supplicant_run() to add interfaces before the main event loop has been started. In addition, new interfaces can be added dynamically while wpa_supplicant is already running. This could happen, e.g., when a hotplug network adapter is inserted.

Definition at line 2386 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.185.2.6 void wpa_supplicant_associate (struct wpa_supplicant * *wpa_s*, struct wpa_scan_result * *bss*, struct wpa_ssid * *ssid*)

Request association.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

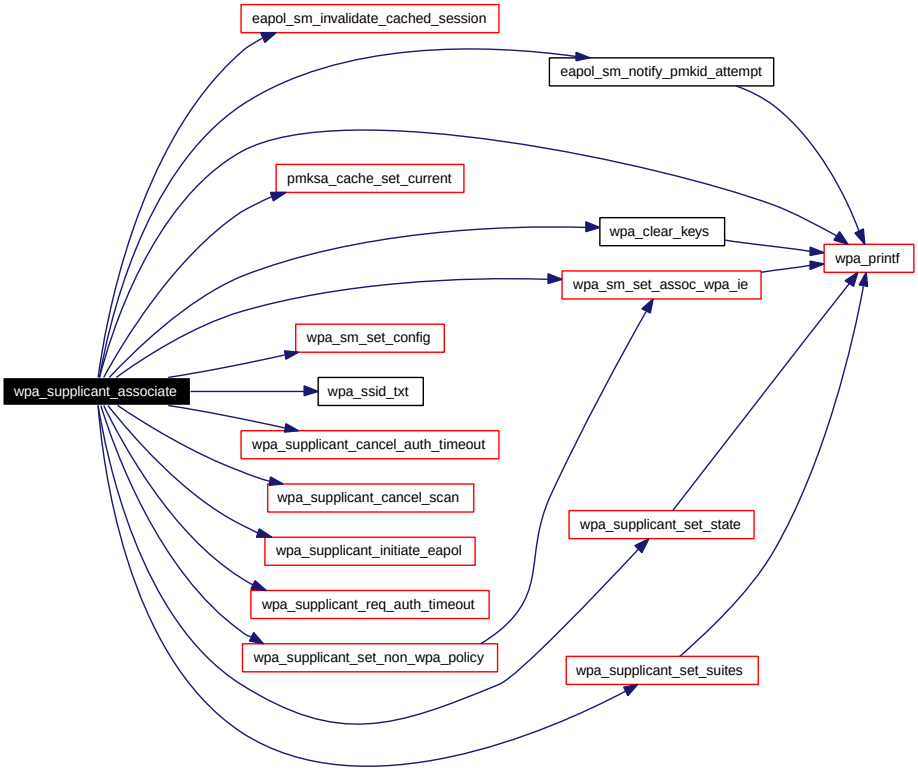
***bss*** Scan results for the selected BSS, or NULL if not available

***ssid*** Configuration data for the selected network

This function is used to request wpa_supplicant to associate with a BSS.

Definition at line 1336 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.185.2.7 void wpa_supplicant_cancel_auth_timeout (struct wpa_supplicant * *wpa_s*)

Cancel authentication timeout.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

This function is used to cancel authentication timeout scheduled with wpa_supplicant_req_auth_timeout() and it is called when authentication has been completed.

Definition at line 573 of file wpa_supplicant.c.

Here is the call graph for this function:


#### 6.185.2.8 void wpa_supplicant_cancel_scan (struct wpa_supplicant * *wpa_s*)

Cancel a scheduled scan request.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

This function is used to cancel a scan request scheduled with wpa_supplicant_req_scan().

Definition at line 517 of file wpa_supplicant.c.

Here is the call graph for this function:


#### 6.185.2.9 void wpa_supplicant_deauthenticate (struct wpa_supplicant * *wpa_s*, int *reason_code*)

Deauthenticate the current connection.

**Parameters:**

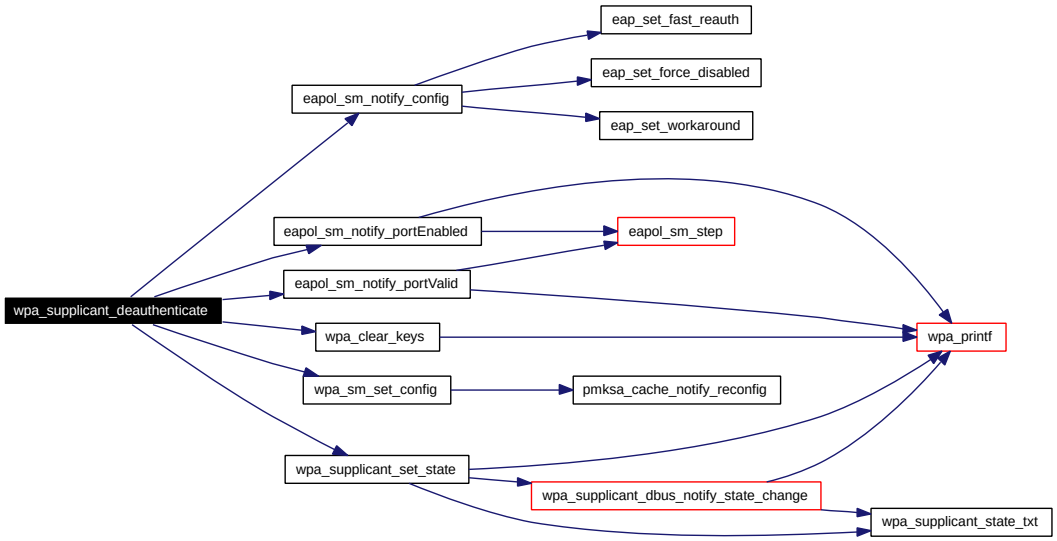
***wpa_s*** Pointer to wpa_supplicant data

***reason_code*** IEEE 802.11 reason code for the deauthenticate frame

This function is used to request wpa_supplicant to disassociate with the current AP.

Definition at line 1604 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.185.2.10 void wpa_supplicant_deinit (struct wpa_global * *global*)

Deinitialize wpa_supplicant.

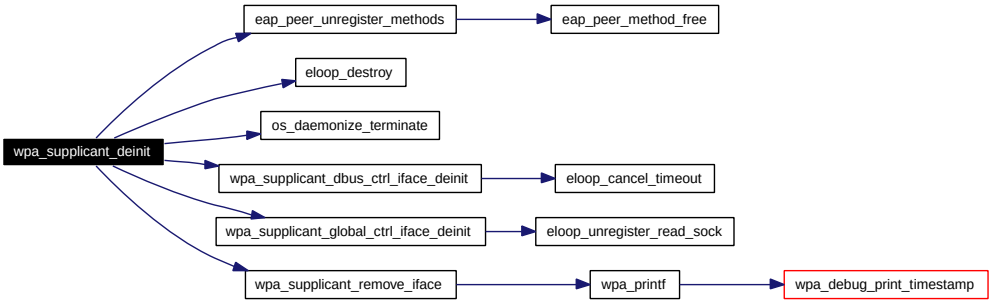
**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

This function is called to deinitialize wpa_supplicant and to free all allocated resources. Remaining network interfaces will also be removed.

Definition at line 2608 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.11 void wpa_supplicant_disassociate (struct wpa_supplicant * *wpa_s*, int *reason_code*)

Disassociate the current connection.

**Parameters:**

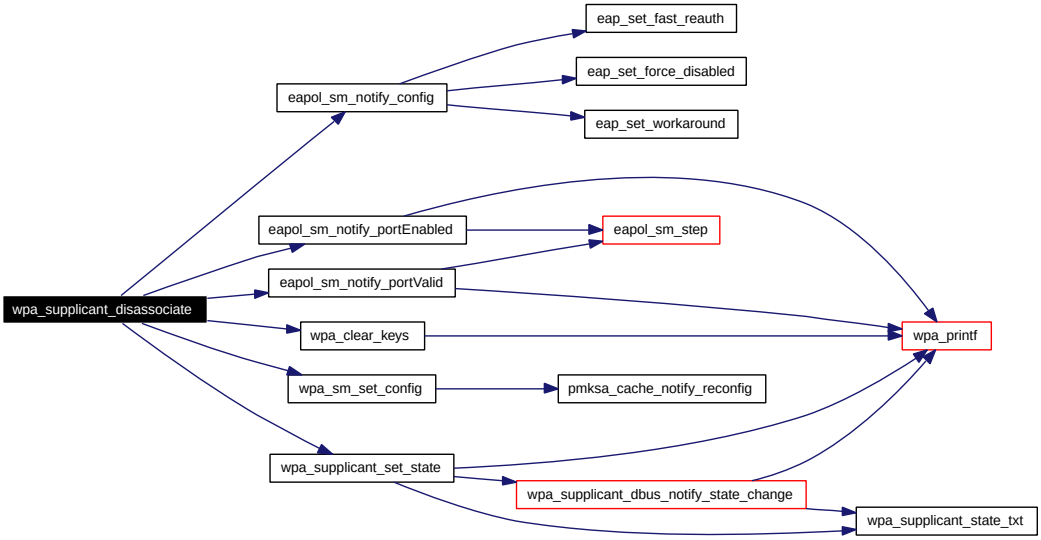
***wpa_s*** Pointer to wpa_supplicant data

***reason_code*** IEEE 802.11 reason code for the disassociate frame

This function is used to request wpa_supplicant to disassociate with the current AP.

Definition at line 1573 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.12 int wpa_supplicant_driver_init (struct wpa_supplicant * *wpa_s*, int *wait_for_interface*)

Initialize driver interface parameters.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***wait_for_interface*** 0 = do not wait for the interface (reports a failure if the interface is not present), 1 = wait until the interface is available

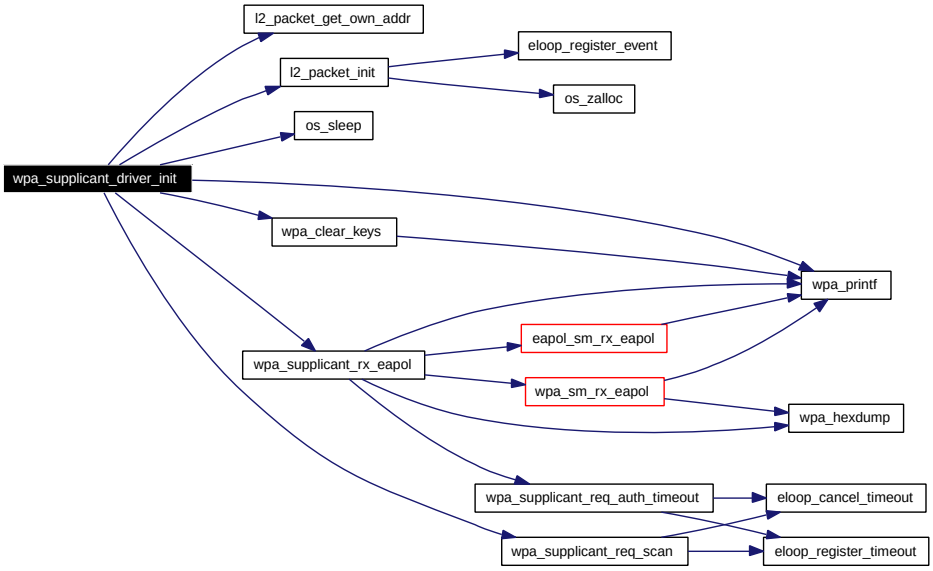
**Returns:**

0 on success, -1 on failure

This function is called to initialize driver interface parameters. wpa_drv_init() must have been called before this function to initialize the driver interface.

Definition at line 1981 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.13 struct wpa_supplicant* wpa_supplicant_get_iface (struct wpa_global * *global*, const char * *ifname*)

Get a new network interface.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

***ifname*** Interface name

**Returns:**

Pointer to the interface or NULL if not found

Definition at line 2471 of file wpa_supplicant.c.

### 6.185.2.14 int wpa_supplicant_get_scan_results (struct wpa_supplicant * *wpa_s*)

Get scan results.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

**Returns:**
0 on success, -1 on failure

This function is request the current scan results from the driver and stores a local copy of the results in wpa_s->scan_results.

Definition at line 1636 of file wpa_supplicant.c.

Here is the call graph for this function:


### 6.185.2.15 struct wpa_ssid* wpa_supplicant_get_ssid (struct wpa_supplicant * *wpa_s*)

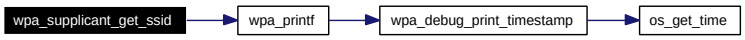
Get a pointer to the current network structure.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

**Returns:**
A pointer to the current network structure or NULL on failure

Definition at line 1743 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.16 wpa_states wpa_supplicant_get_state (struct wpa_supplicant * *wpa_s*)

Get the connection state.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

**Returns:**
The current connection state (WPA_*)

Definition at line 851 of file wpa_supplicant.c.

### 6.185.2.17 struct wpa_global* wpa_supplicant_init (struct wpa_params * *params*)

Initialize wpa_supplicant.

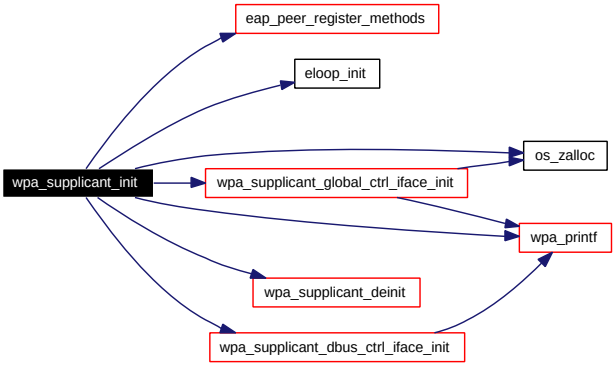
**Parameters:**
***params*** Parameters for wpa_supplicant

**Returns:**
Pointer to global wpa_supplicant data, or NULL on failure

This function is used to initialize wpa_supplicant. After successful initialization, the returned data pointer can be used to add and remove network interfaces, and eventually, to deinitialize wpa_supplicant.

Definition at line 2494 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.18 void wpa_supplicant_initiate_eapol (struct wpa_supplicant * *wpa_s*)

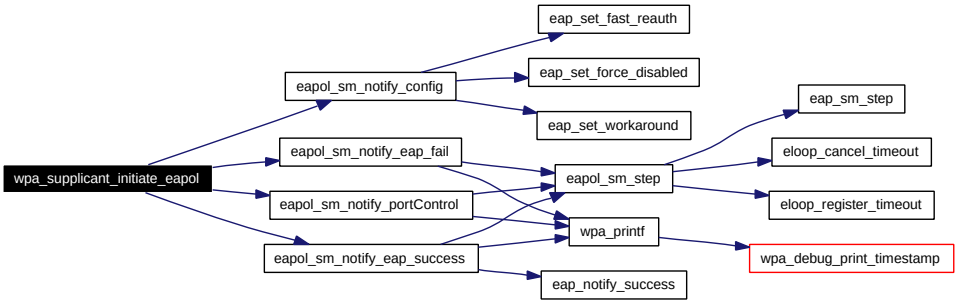
Configure EAPOL state machine.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

This function is used to configure EAPOL state machine based on the selected authentication mode.

Definition at line 589 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.19 int wpa_supplicant_reload_configuration (struct wpa_supplicant * *wpa_s*)

Reload configuration data.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

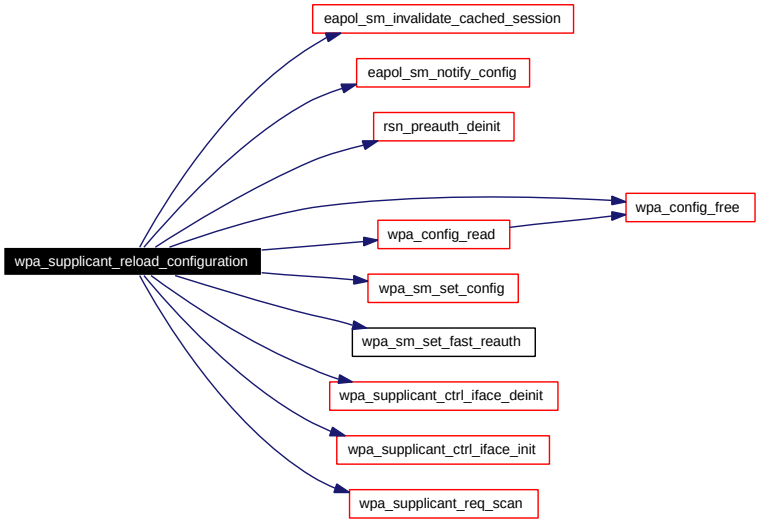
**Returns:**

0 on success or -1 if configuration parsing failed

This function can be used to request that the configuration data is reloaded (e.g., after configuration file change). This function is reloading configuration only for one interface, so this may need to be called multiple times if wpa_supplicant is controlling multiple interfaces and all interfaces need reconfiguration.

Definition at line 892 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.20 int wpa_supplicant_remove_iface (struct wpa_global * *global*, struct wpa_supplicant * *wpa_s*)

Remove a network interface.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

***wpa_s*** Pointer to the network interface to be removed

**Returns:**

0 if interface was removed, -1 if interface was not found

This function can be used to dynamically remove network interfaces from wpa_supplicant, e.g., when a hotplug network adapter is ejected. In addition, this function is used to remove all remaining interdaces when wpa_supplicant is terminated.

Definition at line 2438 of file wpa_supplicant.c.

Here is the call graph for this function:

#### 6.185.2.21 void wpa_supplicant_req_auth_timeout (struct wpa_supplicant * *wpa_s*, int *sec*, int *usec*)

Schedule a timeout for authentication.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

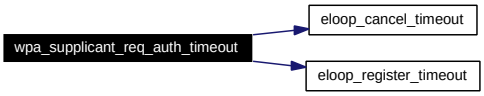
***sec*** Number of seconds after which to time out authentication

***usec*** Number of microseconds after which to time out authentication

This function is used to schedule a timeout for the current authentication attempt.

Definition at line 550 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.185.2.22 void wpa_supplicant_req_scan (struct wpa_supplicant * *wpa_s*, int *sec*, int *usec*)

Schedule a scan for neighboring access points.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

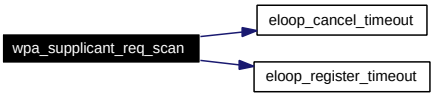
***sec*** Number of seconds after which to scan

***usec*** Number of microseconds after which to scan

This function is used to schedule a scan for neighboring access points after the specified time.

Definition at line 500 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.185.2.23 int wpa_supplicant_run (struct wpa_global * *global*)

Run the wpa_supplicant main event loop.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

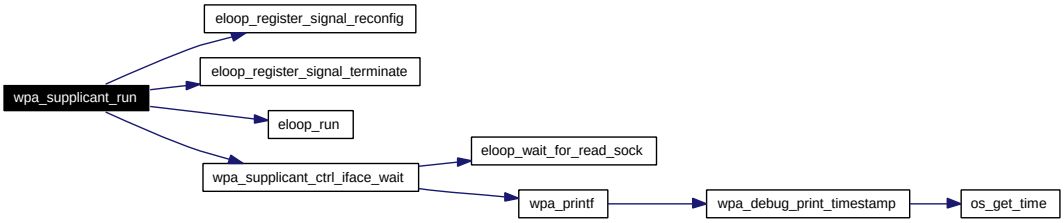
**Returns:**

0 after successful event loop run, -1 on failure

This function starts the main event loop and continues running as long as there are any remaining events. In most cases, this function is running as long as the wpa_supplicant process in still in use.

Definition at line 2576 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.24 void wpa_supplicant_rx_eapol (void * *ctx*, const u8 * *src_addr*, const u8 * *buf*, size_t *len*)

Deliver a received EAPOL frame to wpa_supplicant.

**Parameters:**

***ctx*** Context pointer (wpa_s)

***src_addr*** Source address of the EAPOL frame

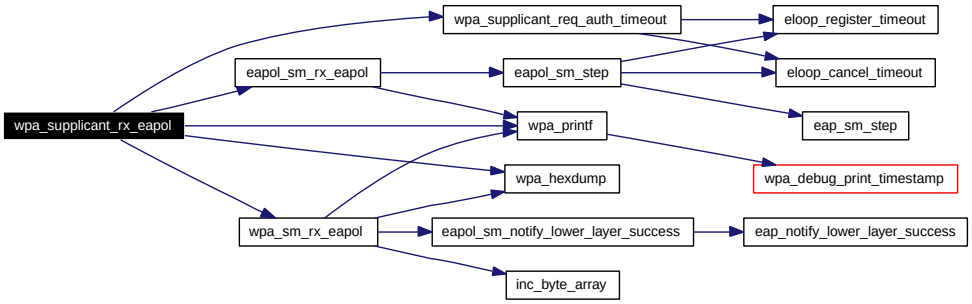
***buf*** EAPOL data starting from the EAPOL header (i.e., no Ethernet header)

***len*** Length of the EAPOL data

This function is called for each received EAPOL frame.

Definition at line 1924 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.25 void wpa_supplicant_set_non_wpa_policy (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Set WPA parameters to non-WPA mode.

**Parameters:**

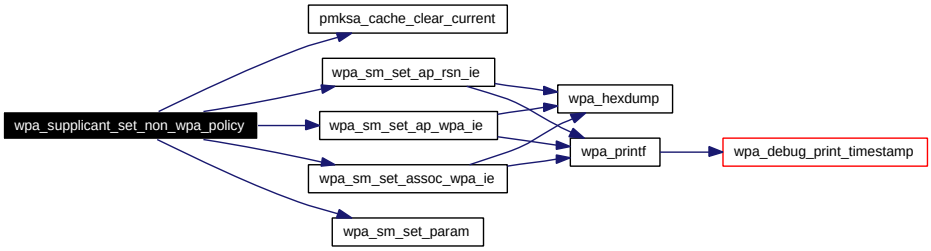
***wpa_s*** Pointer to wpa_supplicant data

***ssid*** Configuration data for the network

This function is used to configure WPA state machine and related parameters to a mode where WPA is not enabled. This is called as part of the authentication configuration when the selected network does not use WPA.

Definition at line 642 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.26 void wpa_supplicant_set_state (struct wpa_supplicant * *wpa_s*, wpa_states *state*)

Set current connection state.

**Parameters:**

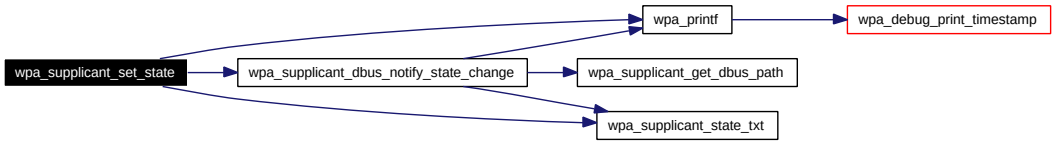
***wpa_s*** Pointer to wpa_supplicant data

***state*** The new connection state

This function is called whenever the connection state changes, e.g., association is completed for WPA/WPA2 4-Way Handshake is started.

Definition at line 814 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.185.2.27 int wpa_supplicant_set_suites (struct wpa_supplicant * *wpa_s*, struct wpa_scan_result * *bss*, struct wpa_ssid * *ssid*, u8 * *wpa_ie*, size_t * *wpa_ie_len*)

Set authentication and encryption parameters.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***bss*** Scan results for the selected BSS, or NULL if not available

***ssid*** Configuration data for the selected network

***wpa_ie*** Buffer for the WPA/RSN IE

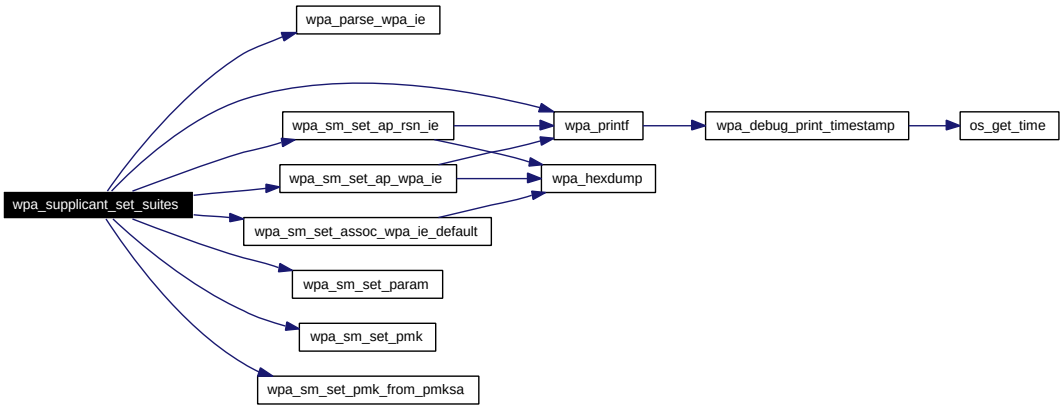
***wpa_ie_len*** Maximum wpa_ie buffer size on input. This is changed to be the used buffer length in case the functions returns success.

**Returns:**
0 on success or -1 on failure

This function is used to configure authentication and encryption parameters based on the network configuration and scan result for the selected BSS (if available).

Definition at line 1177 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.185.2.28 const char* wpa_supplicant_state_txt (int *state*)

Get the connection state name as a text string.

**Parameters:**
***state*** State (wpa_state; WPA_*)

**Returns:**
The state name as a printable text string

Definition at line 780 of file wpa_supplicant.c.

### 6.185.3 Variable Documentation

#### 6.185.3.1 const char* wpa_supplicant_full_license1

**Initial value:**

```
"This program is free software; you can redistribute it and/or modify\n"
"it under the terms of the GNU General Public License version 2 as\n"
"published by the Free Software Foundation.\n"
"\n"
"This program is distributed in the hope that it will be useful,\n"
```

```
"but WITHOUT ANY WARRANTY; without even the implied warranty of\n"
"MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the\n"
"GNU General Public License for more details.\n"
"\n"
```

Definition at line 58 of file wpa_supplicant.c.

#### 6.185.3.2 const char* wpa_supplicant_full_license2

**Initial value:**

```
"You should have received a copy of the GNU General Public License\n"
"along with this program; if not, write to the Free Software\n"
"Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA\n"
"\n"
"Alternatively, this software may be distributed under the terms of the\n"
"BSD license.\n"
"\n"
"Redistribution and use in source and binary forms, with or without\n"
"modification, are permitted provided that the following conditions are\n"
"met:\n"
"\n"
```

Definition at line 68 of file wpa_supplicant.c.

#### 6.185.3.3 const char* wpa_supplicant_full_license3

**Initial value:**

```
"1. Redistributions of source code must retain the above copyright\n"
"   notice, this list of conditions and the following disclaimer.\n"
"\n"
"2. Redistributions in binary form must reproduce the above copyright\n"
"   notice, this list of conditions and the following disclaimer in the\n"
"   documentation and/or other materials provided with the distribution.\n"
"\n"
```

Definition at line 80 of file wpa_supplicant.c.

#### 6.185.3.4 const char* wpa_supplicant_full_license4

**Initial value:**

```
"3. Neither the name(s) of the above-listed copyright holder(s) nor the\n"
"   names of its contributors may be used to endorse or promote products\n"
"   derived from this software without specific prior written permission.\n"
"\n"
"THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS\n"
"\"AS IS\" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT\n"
"LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR\n"
"A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT\n"
```

Definition at line 88 of file wpa_supplicant.c.

#### 6.185.3.5 const char* wpa_supplicant_full_license5

**Initial value:**

```
"OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,\n"
"SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT\n"
"LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,\n"
"DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY\n"
"THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT\n"
"(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE\n"
"OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.\n"
"\n"
```

Definition at line 97 of file wpa_supplicant.c.

#### 6.185.3.6 const char* wpa_supplicant_license

**Initial value:**

```
"This program is free software. You can distribute it and/or modify it\n"
"under the terms of the GNU General Public License version 2.\n"
"\n"
"Alternatively, this software may be distributed under the terms of the\n"
"BSD license. See README and COPYING for more details.\n"
```

Definition at line 44 of file wpa_supplicant.c.

#### 6.185.3.7 const char* wpa_supplicant_version

**Initial value:**

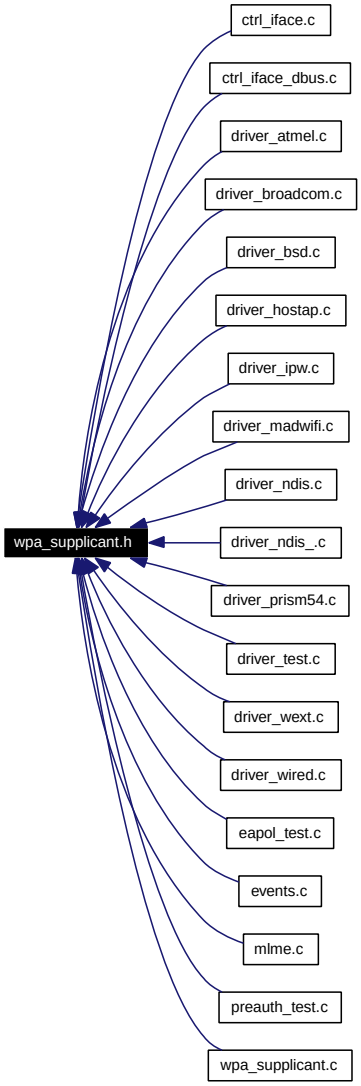
```
"wpa_supplicant v" VERSION_STR "\n"
"Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi> and contributors"
```

Definition at line 40 of file wpa_supplicant.c.

# 6.186 wpa_supplicant.h File Reference

wpa_supplicant - Exported functions for wpa_supplicant modules

This graph shows which files directly or indirectly include this file:




## Typedefs

- typedef enum wpa_event_type wpa_event_type

## Enumerations

- enum wpa_event_type {

  EVENT_ASSOC, EVENT_DISASSOC, EVENT_MICHAEL_MIC_FAILURE, EVENT_SCAN_-RESULTS,

EVENT_ASSOCINFO, EVENT_INTERFACE_STATUS, EVENT_PMKID_CANDIDATE, EVENT_STKSTART }

## Functions

- void wpa_supplicant_event (struct wpa_supplicant *wpa_s, wpa_event_type event, union wpa_event_data *data)

  *Report a driver event for wpa_supplicant.*

- void wpa_supplicant_rx_eapol (void *ctx, const u8 *src_addr, const u8 *buf, size_t len)

  *Deliver a received EAPOL frame to wpa_supplicant.*

### 6.186.1 Detailed Description

wpa_supplicant - Exported functions for wpa_supplicant modules

**Copyright**

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file wpa_supplicant.h.

### 6.186.2 Typedef Documentation

#### 6.186.2.1 typedef enum wpa_event_type wpa_event_type

enum wpa_event_type - Event type for wpa_supplicant_event() calls

### 6.186.3 Enumeration Type Documentation

#### 6.186.3.1 enum wpa_event_type

enum wpa_event_type - Event type for wpa_supplicant_event() calls

**Enumeration values:**

***EVENT_ASSOC*** Association completed.

This event needs to be delivered when the driver completes IEEE 802.11 association or reassociation successfully. wpa_driver_ops::get_bssid() is expected to provide the current BSSID after this even has been generated. In addition, optional EVENT_ASSOCINFO may be generated just before EVENT_ASSOC to provide more information about the association. If the driver interface gets both of these events at the same time, it can also include the assoc_info data in EVENT_ASSOC call.

***EVENT_DISASSOC*** Association lost.

This event should be called when association is lost either due to receiving deauthenticate or disassociate frame from the AP or when sending either of these frames to the current AP.

***EVENT_MICHAEL_MIC_FAILURE*** Michael MIC (TKIP) detected.

This event must be delivered when a Michael MIC error is detected by the local driver. Additional data is for event processing is provided with union wpa_event_data::michael_mic_failure. This information is used to request new encyption key and to initiate TKIP countermeasures if needed.

***EVENT_SCAN_RESULTS*** Scan results available.

This event must be called whenever scan results are available to be fetched with struct wpa_driver_ops::get_scan_results(). This event is expected to be used some time after struct wpa_driver_ops::scan() is called. If the driver provides an unsolicited event when the scan has been completed, this event can be used to trigger EVENT_SCAN_RESULTS call. If such event is not available from the driver, the driver wrapper code is expected to use a registered timeout to generate EVENT_SCAN_RESULTS call after the time that the scan is expected to be completed.

***EVENT_ASSOCINFO*** Report optional extra information for association.

This event can be used to report extra association information for EVENT_ASSOC processing. This extra information includes IEs from association frames and Beacon/Probe Response frames in union wpa_event_data::assoc_info. EVENT_ASSOCINFO must be send just before EVENT_ASSOC. Alternatively, the driver interface can include assoc_info data in the EVENT_ASSOC call if it has all the information available at the same point.

***EVENT_INTERFACE_STATUS*** Report interface status changes.

This optional event can be used to report changes in interface status (interface added/removed) using union wpa_event_data::interface_status. This can be used to trigger wpa_supplicant to stop and re-start processing for the interface, e.g., when a cardbus card is ejected/inserted.

***EVENT_PMKID_CANDIDATE*** Report a candidate AP for pre-authentication.

This event can be used to inform wpa_supplicant about candidates for RSN (WPA2) pre-authentication. If wpa_supplicant is not responsible for scan request (ap_scan=2 mode), this event is required for pre-authentication. If wpa_supplicant is performing scan request (ap_scan=1), this event is optional since scan results can be used to add pre-authentication candidates. union wpa_event_data::pmkid_candidate is used to report the BSSID of the candidate and priority of the candidate, e.g., based on the signal strength, in order to try to pre-authenticate first with candidates that are most likely targets for re-association.

EVENT_PMKID_CANDIDATE can be called whenever the driver has updates on the candidate list. In addition, it can be called for the current AP and APs that have existing PMKSA cache entries. wpa_supplicant will automatically skip pre-authentication in cases where a valid PMKSA exists. When more than one candidate exists, this event should be generated once for each candidate.

Driver will be notified about successful pre-authentication with struct wpa_driver_ops::add_pmkid() calls.

***EVENT_STKSTART*** Request STK handshake (MLME-STKSTART.request).

This event can be used to inform wpa_supplicant about desire to set up secure direct link connection between two stations as defined in IEEE 802.11e with a new PeerKey mechanism that replaced the original STAKey negotiation. The caller will need to set peer address for the event.

Definition at line 27 of file wpa_supplicant.h.

## 6.186.4 Function Documentation

### 6.186.4.1 void wpa_supplicant_event (struct wpa_supplicant * *wpa_s*, wpa_event_type *event*, union wpa_event_data * *data*)

Report a driver event for wpa_supplicant.

**Parameters:**

***wpa_s*** pointer to wpa_supplicant data; this is the ctx variable registered with struct wpa_driver_ops::init()

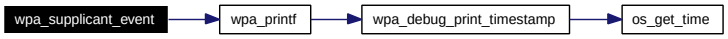
***event*** event type (defined above)

***data*** possible extra data for the event

Driver wrapper code should call this function whenever an event is received from the driver.

Definition at line 800 of file events.c.

Here is the call graph for this function:




### 6.186.4.2 void wpa_supplicant_rx_eapol (void * *ctx*, const u8 * *src_addr*, const u8 * *buf*, size_t *len*)

Deliver a received EAPOL frame to wpa_supplicant.

**Parameters:**

***ctx*** Context pointer (wpa_s)

***src_addr*** Source address of the EAPOL frame

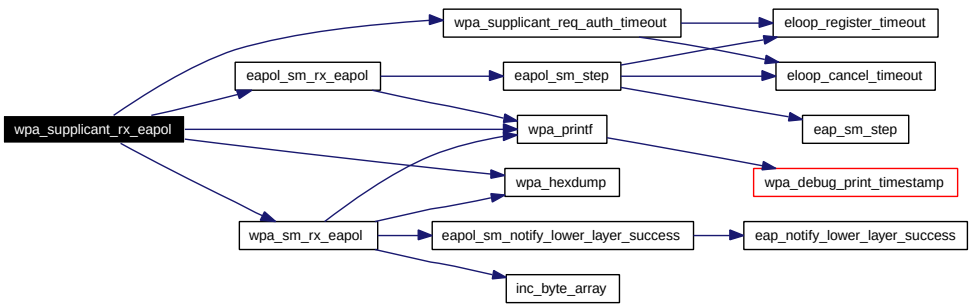
***buf*** EAPOL data starting from the EAPOL header (i.e., no Ethernet header)

***len*** Length of the EAPOL data

This function is called for each received EAPOL frame.

Definition at line 1924 of file wpa_supplicant.c.

Here is the call graph for this function:

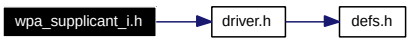


# 6.187 wpa_supplicant_i.h File Reference

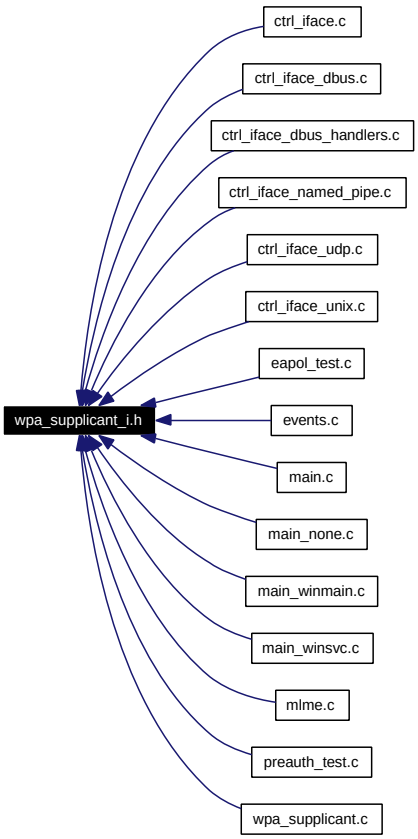
wpa_supplicant - Internal definitions

```
#include "driver.h"
```

Include dependency graph for wpa_supplicant_i.h:




This graph shows which files directly or indirectly include this file:




## Defines

- #define **BROADCAST_SSID_SCAN** ((struct wpa_ssid *) 1)

## Functions

- void wpa_supplicant_cancel_scan (struct wpa_supplicant *wpa_s)

  *Cancel a scheduled scan request.*

- int wpa_supplicant_reload_configuration (struct wpa_supplicant *wpa_s)

*Reload configuration data.*

- const char * wpa_supplicant_state_txt (int state)

  *Get the connection state name as a text string.*

- int wpa_supplicant_driver_init (struct wpa_supplicant *wpa_s, int wait_for_interface)

  *Initialize driver interface parameters.*

- wpa_blacklist * wpa_blacklist_get (struct wpa_supplicant *wpa_s, const u8 *bssid)

  *Get the blacklist entry for a BSSID.*

- int wpa_blacklist_add (struct wpa_supplicant *wpa_s, const u8 *bssid)

  *Add an BSSID to the blacklist.*

- void wpa_blacklist_clear (struct wpa_supplicant *wpa_s)

  *Clear the blacklist of all entries.*

- int wpa_supplicant_set_suites (struct wpa_supplicant *wpa_s, struct wpa_scan_result *bss, struct wpa_ssid *ssid, u8 *wpa_ie, size_t *wpa_ie_len)

  *Set authentication and encryption parameters.*

- void wpa_supplicant_associate (struct wpa_supplicant *wpa_s, struct wpa_scan_result *bss, struct wpa_ssid *ssid)

  *Request association.*

- void wpa_supplicant_set_non_wpa_policy (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  *Set WPA parameters to non-WPA mode.*

- void wpa_supplicant_initiate_eapol (struct wpa_supplicant *wpa_s)

  *Configure EAPOL state machine.*

- int wpa_supplicant_get_scan_results (struct wpa_supplicant *wpa_s)

  *Get scan results.*

- void wpa_clear_keys (struct wpa_supplicant *wpa_s, const u8 *addr)

  *Clear keys configured for the driver.*

- void wpa_supplicant_req_auth_timeout (struct wpa_supplicant *wpa_s, int sec, int usec)

  *Schedule a timeout for authentication.*

- void wpa_supplicant_set_state (struct wpa_supplicant *wpa_s, wpa_states state)

  *Set current connection state.*

- wpa_ssid * wpa_supplicant_get_ssid (struct wpa_supplicant *wpa_s)

  *Get a pointer to the current network structure.*

- void wpa_supplicant_cancel_auth_timeout (struct wpa_supplicant *wpa_s)

  *Cancel authentication timeout.*

- void wpa_supplicant_deauthenticate (struct wpa_supplicant *wpa_s, int reason_code)

*Deauthenticate the current connection.*

- void wpa_supplicant_disassociate (struct wpa_supplicant *wpa_s, int reason_code)

  *Disassociate the current connection.*

- void wpa_supplicant_req_scan (struct wpa_supplicant *wpa_s, int sec, int usec)

  *Schedule a scan for neighboring access points.*

- void **wpa_show_license** (void)
- wpa_supplicant * wpa_supplicant_add_iface (struct wpa_global *global, struct wpa_interface *iface)

  *Add a new network interface.*

- int wpa_supplicant_remove_iface (struct wpa_global *global, struct wpa_supplicant *wpa_s)

  *Remove a network interface.*

- wpa_supplicant * wpa_supplicant_get_iface (struct wpa_global *global, const char *ifname)

  *Get a new network interface.*

- wpa_global * wpa_supplicant_init (struct wpa_params *params)

  *Initialize wpa_supplicant.*

- int wpa_supplicant_run (struct wpa_global *global)

  *Run the wpa_supplicant main event loop.*

- void wpa_supplicant_deinit (struct wpa_global *global)

  *Deinitialize wpa_supplicant.*

- int wpa_supplicant_scard_init (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  *Initialize SIM/USIM access with PC/SC.*

## 6.187.1 Detailed Description

wpa_supplicant - Internal definitions

**Copyright**

Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file wpa_supplicant_i.h.

## 6.187.2 Function Documentation

### 6.187.2.1 int wpa_blacklist_add (struct wpa_supplicant * *wpa_s*, const u8 * *bssid*)

Add an BSSID to the blacklist.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***bssid*** BSSID to be added to the blacklist

**Returns:**

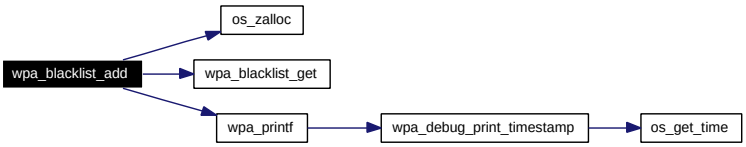
0 on success, -1 on failure

This function adds the specified BSSID to the blacklist or increases the blacklist count if the BSSID was already listed. It should be called when an association attempt fails either due to the selected BSS rejecting association or due to timeout.

This blacklist is used to force wpa_supplicant to go through all available BSSes before retrying to associate with an BSS that rejected or timed out association. It does not prevent the listed BSS from being used; it only changes the order in which they are tried.

Definition at line 418 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.187.2.2 void wpa_blacklist_clear (struct wpa_supplicant * *wpa_s*)

Clear the blacklist of all entries.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

Definition at line 474 of file wpa_supplicant.c.

Here is the call graph for this function:


### 6.187.2.3 struct wpa_blacklist* wpa_blacklist_get (struct wpa_supplicant * *wpa_s*, const u8 * *bssid*)

Get the blacklist entry for a BSSID.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***bssid*** BSSID

**Returns:**
Matching blacklist entry for the BSSID or NULL if not found

Definition at line 385 of file wpa_supplicant.c.

#### 6.187.2.4 void wpa_clear_keys (struct wpa_supplicant * *wpa_s*, const u8 * *addr*)

Clear keys configured for the driver.

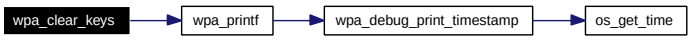
**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

***addr*** Previously used BSSID or NULL if not available

This function clears the encryption keys that has been previously configured for the driver.

Definition at line 739 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.5 struct wpa_supplicant* wpa_supplicant_add_iface (struct wpa_global * *global*, struct wpa_interface * *iface*)

Add a new network interface.

**Parameters:**
***global*** Pointer to global data from wpa_supplicant_init()

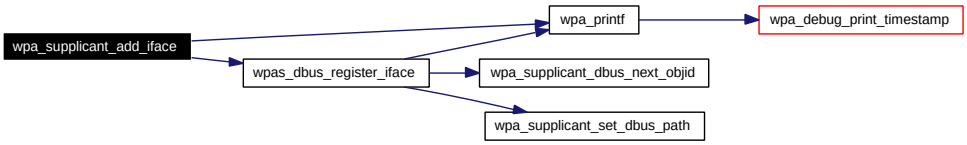
***iface*** Interface configuration options

**Returns:**
Pointer to the created interface or NULL on failure

This function is used to add new network interfaces for wpa_supplicant. This can be called before wpa_supplicant_run() to add interfaces before the main event loop has been started. In addition, new interfaces can be added dynamically while wpa_supplicant is already running. This could happen, e.g., when a hotplug network adapter is inserted.

Definition at line 2386 of file wpa_supplicant.c.

Here is the call graph for this function:



#### 6.187.2.6 void wpa_supplicant_associate (struct wpa_supplicant * *wpa_s*, struct wpa_scan_result * *bss*, struct wpa_ssid * *ssid*)

Request association.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

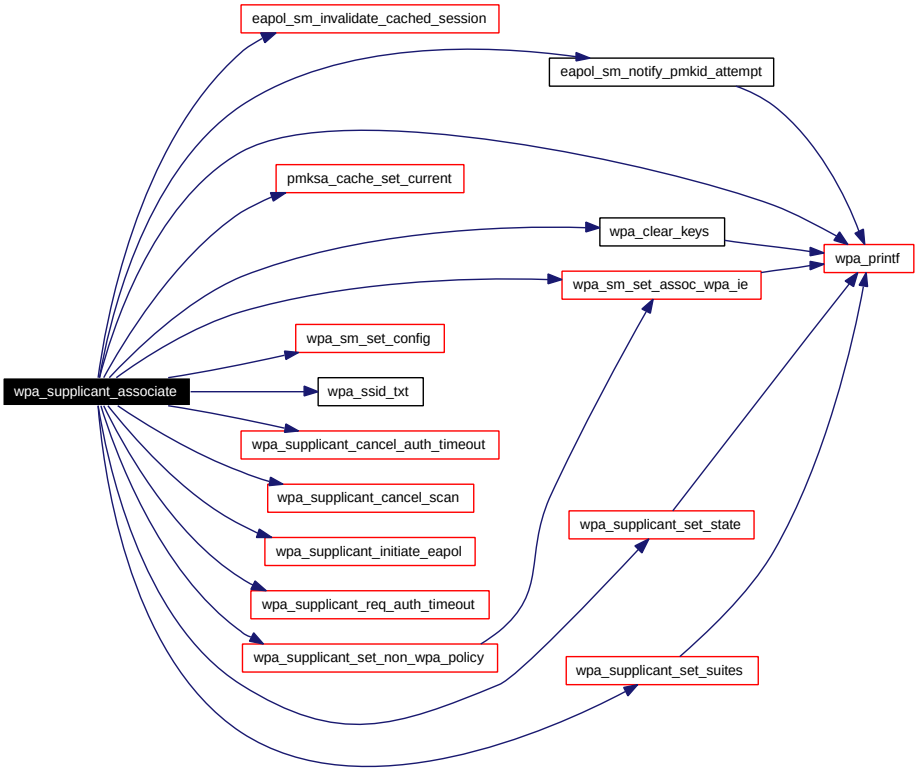
***bss*** Scan results for the selected BSS, or NULL if not available

***ssid*** Configuration data for the selected network

This function is used to request wpa_supplicant to associate with a BSS.

Definition at line 1336 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.7 void wpa_supplicant_cancel_auth_timeout (struct wpa_supplicant * *wpa_s*)

Cancel authentication timeout.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

This function is used to cancel authentication timeout scheduled with wpa_supplicant_req_auth_timeout() and it is called when authentication has been completed.

Definition at line 573 of file wpa_supplicant.c.

Here is the call graph for this function:


#### 6.187.2.8 void wpa_supplicant_cancel_scan (struct wpa_supplicant * *wpa_s*)

Cancel a scheduled scan request.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

This function is used to cancel a scan request scheduled with wpa_supplicant_req_scan().

Definition at line 517 of file wpa_supplicant.c.

Here is the call graph for this function:


#### 6.187.2.9 void wpa_supplicant_deauthenticate (struct wpa_supplicant * *wpa_s*, int *reason_code*)

Deauthenticate the current connection.

**Parameters:**

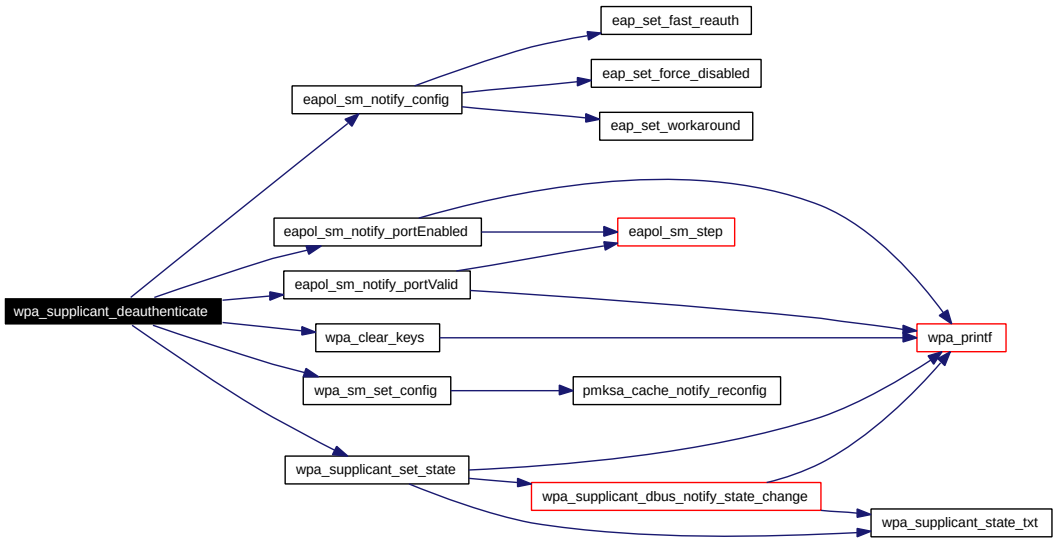
***wpa_s*** Pointer to wpa_supplicant data

***reason_code*** IEEE 802.11 reason code for the deauthenticate frame

This function is used to request wpa_supplicant to disassociate with the current AP.

Definition at line 1604 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.187.2.10 void wpa_supplicant_deinit (struct wpa_global * *global*)

Deinitialize wpa_supplicant.

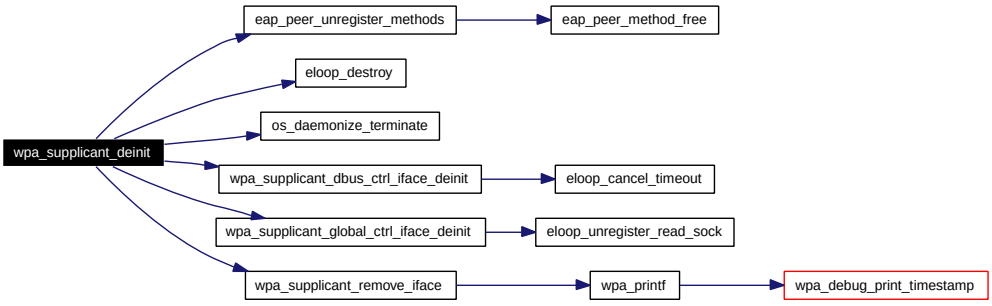
**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

This function is called to deinitialize wpa_supplicant and to free all allocated resources. Remaining network interfaces will also be removed.

Definition at line 2608 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.187.2.11 void wpa_supplicant_disassociate (struct wpa_supplicant * *wpa_s*, int *reason_code*)

Disassociate the current connection.

**Parameters:**

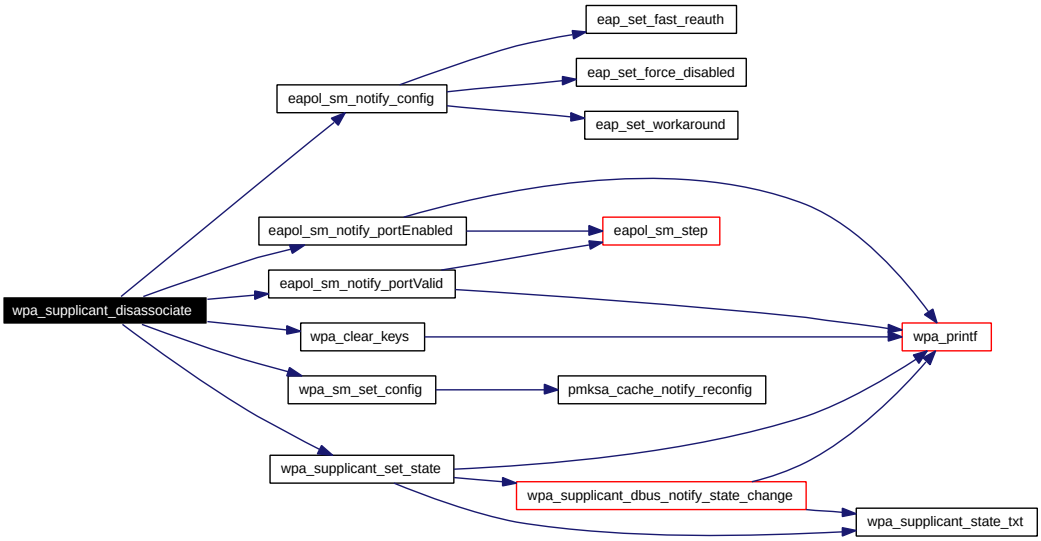
***wpa_s*** Pointer to wpa_supplicant data

***reason_code*** IEEE 802.11 reason code for the disassociate frame

This function is used to request wpa_supplicant to disassociate with the current AP.

Definition at line 1573 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.187.2.12 int wpa_supplicant_driver_init (struct wpa_supplicant * *wpa_s*, int *wait_for_interface*)

Initialize driver interface parameters.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***wait_for_interface*** 0 = do not wait for the interface (reports a failure if the interface is not present), 1 = wait until the interface is available

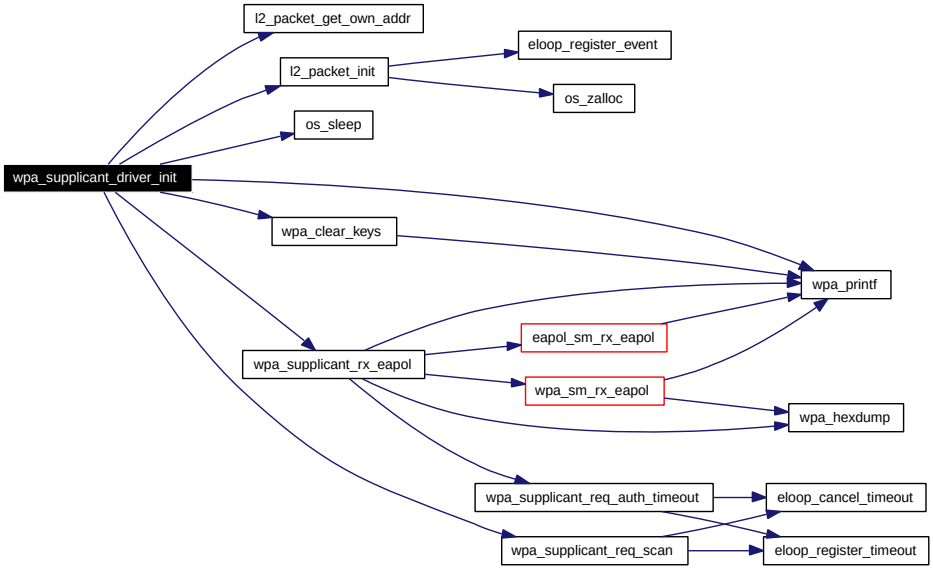
**Returns:**

0 on success, -1 on failure

This function is called to initialize driver interface parameters. wpa_drv_init() must have been called before this function to initialize the driver interface.

Definition at line 1981 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.13 struct wpa_supplicant* wpa_supplicant_get_iface (struct wpa_global * *global*, const char * *ifname*)

Get a new network interface.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

***ifname*** Interface name

**Returns:**

Pointer to the interface or NULL if not found

Definition at line 2471 of file wpa_supplicant.c.

#### 6.187.2.14 int wpa_supplicant_get_scan_results (struct wpa_supplicant * *wpa_s*)

Get scan results.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

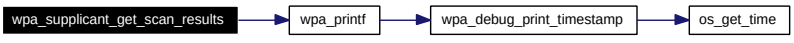
**Returns:**

0 on success, -1 on failure

This function is request the current scan results from the driver and stores a local copy of the results in wpa_s->scan_results.

Definition at line 1636 of file wpa_supplicant.c.

Here is the call graph for this function:



#### 6.187.2.15 struct wpa_ssid* wpa_supplicant_get_ssid (struct wpa_supplicant * *wpa_s*)

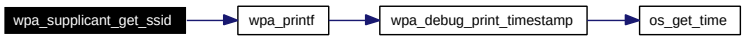
Get a pointer to the current network structure.

**Parameters:**
*wpa_s* Pointer to wpa_supplicant data

**Returns:**
A pointer to the current network structure or NULL on failure

Definition at line 1743 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.16 struct wpa_global* wpa_supplicant_init (struct wpa_params * *params*)

Initialize wpa_supplicant.

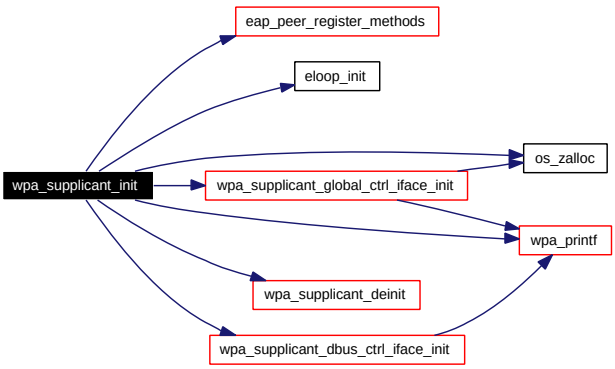
**Parameters:**
*params* Parameters for wpa_supplicant

**Returns:**
Pointer to global wpa_supplicant data, or NULL on failure

This function is used to initialize wpa_supplicant. After successful initialization, the returned data pointer can be used to add and remove network interfaces, and eventually, to deinitialize wpa_supplicant.

Definition at line 2494 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.17 void wpa_supplicant_initiate_eapol (struct wpa_supplicant * *wpa_s*)

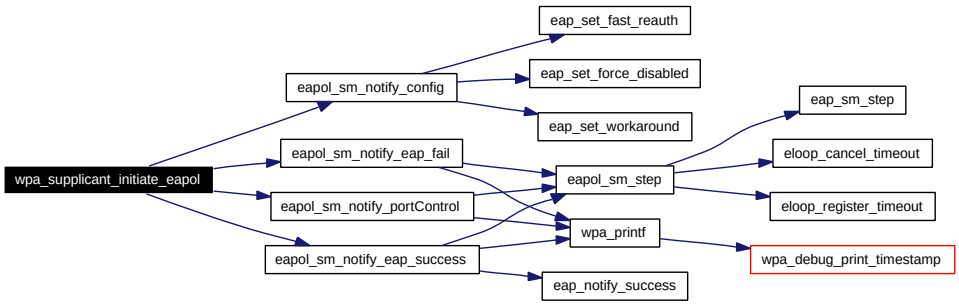
Configure EAPOL state machine.

**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

This function is used to configure EAPOL state machine based on the selected authentication mode.

Definition at line 589 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.187.2.18 int wpa_supplicant_reload_configuration (struct wpa_supplicant * *wpa_s*)

Reload configuration data.

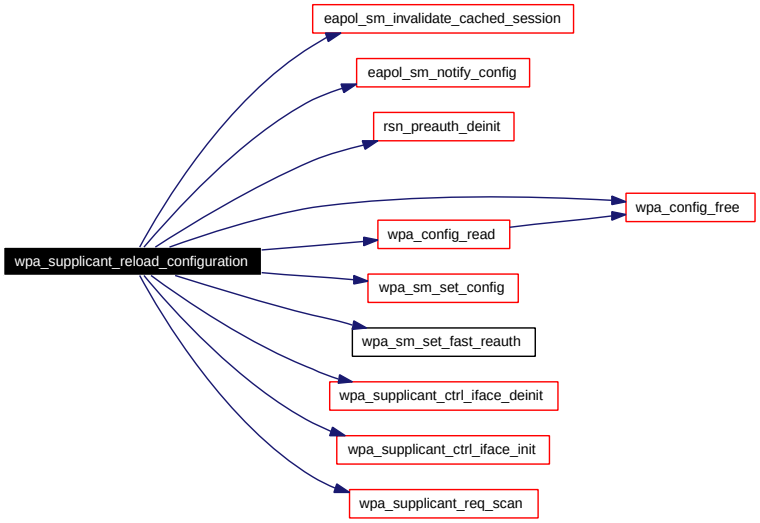
**Parameters:**
***wpa_s*** Pointer to wpa_supplicant data

**Returns:**
0 on success or -1 if configuration parsing failed

This function can be used to request that the configuration data is reloaded (e.g., after configuration file change). This function is reloading configuration only for one interface, so this may need to be called multiple times if wpa_supplicant is controlling multiple interfaces and all interfaces need reconfiguration.

Definition at line 892 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.19 int wpa_supplicant_remove_iface (struct wpa_global * *global*, struct wpa_supplicant * *wpa_s*)

Remove a network interface.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

***wpa_s*** Pointer to the network interface to be removed

**Returns:**

0 if interface was removed, -1 if interface was not found

This function can be used to dynamically remove network interfaces from wpa_supplicant, e.g., when a hotplug network adapter is ejected. In addition, this function is used to remove all remaining interdaces when wpa_supplicant is terminated.

Definition at line 2438 of file wpa_supplicant.c.

Here is the call graph for this function:


#### 6.187.2.20 void wpa_supplicant_req_auth_timeout (struct wpa_supplicant * *wpa_s*, int *sec*, int *usec*)

Schedule a timeout for authentication.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

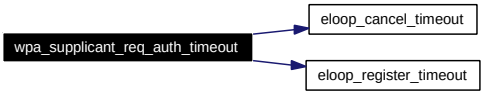
***sec*** Number of seconds after which to time out authentication

***usec*** Number of microseconds after which to time out authentication

This function is used to schedule a timeout for the current authentication attempt.

Definition at line 550 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.21 void wpa_supplicant_req_scan (struct wpa_supplicant * *wpa_s*, int *sec*, int *usec*)

Schedule a scan for neighboring access points.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

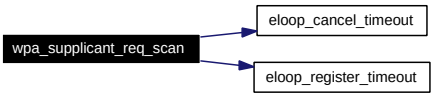
***sec*** Number of seconds after which to scan

***usec*** Number of microseconds after which to scan

This function is used to schedule a scan for neighboring access points after the specified time.

Definition at line 500 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.22 int wpa_supplicant_run (struct wpa_global * *global*)

Run the wpa_supplicant main event loop.

**Parameters:**

***global*** Pointer to global data from wpa_supplicant_init()

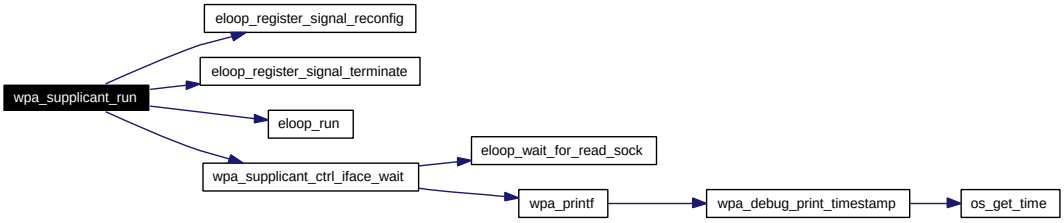
**Returns:**

0 after successful event loop run, -1 on failure

This function starts the main event loop and continues running as long as there are any remaining events. In most cases, this function is running as long as the wpa_supplicant process in still in use.

Definition at line 2576 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.23 int wpa_supplicant_scard_init (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Initialize SIM/USIM access with PC/SC.

**Parameters:**

***wpa_s*** pointer to wpa_supplicant data

***ssid*** Configuration data for the network

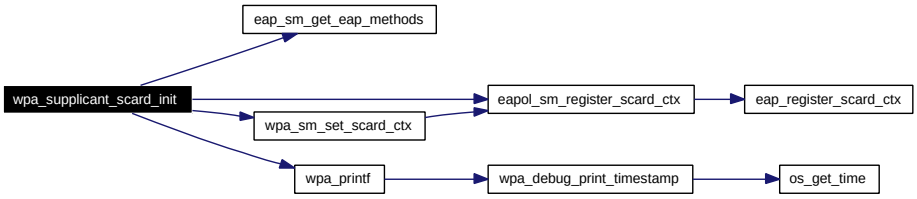
**Returns:**

0 on success, -1 on failure

This function is called when starting authentication with a network that is configured to use PC/SC for SIM/USIM access (EAP-SIM or EAP-AKA).

Definition at line 176 of file events.c.

Here is the call graph for this function:




#### 6.187.2.24 void wpa_supplicant_set_non_wpa_policy (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Set WPA parameters to non-WPA mode.

**Parameters:**

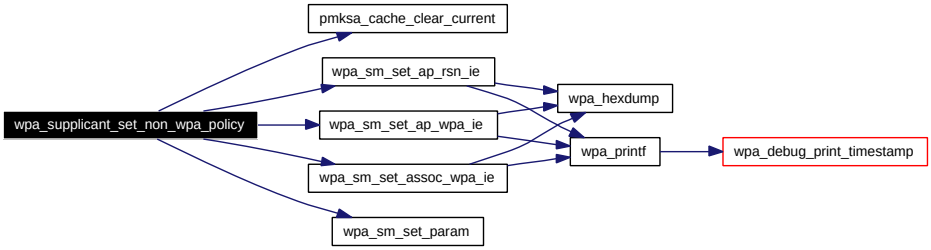
***wpa_s*** Pointer to wpa_supplicant data

***ssid*** Configuration data for the network

This function is used to configure WPA state machine and related parameters to a mode where WPA is not enabled. This is called as part of the authentication configuration when the selected network does not use WPA.

Definition at line 642 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.25 void wpa_supplicant_set_state (struct wpa_supplicant * *wpa_s*, wpa_states *state*)

Set current connection state.

**Parameters:**

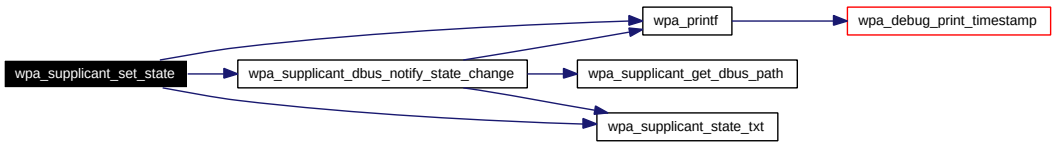
***wpa_s*** Pointer to wpa_supplicant data

***state*** The new connection state

This function is called whenever the connection state changes, e.g., association is completed for WPA/WPA2 4-Way Handshake is started.

Definition at line 814 of file wpa_supplicant.c.

Here is the call graph for this function:




#### 6.187.2.26 int wpa_supplicant_set_suites (struct wpa_supplicant * *wpa_s*, struct wpa_scan_result * *bss*, struct wpa_ssid * *ssid*, u8 * *wpa_ie*, size_t * *wpa_ie_len*)

Set authentication and encryption parameters.

**Parameters:**

***wpa_s*** Pointer to wpa_supplicant data

***bss*** Scan results for the selected BSS, or NULL if not available

***ssid*** Configuration data for the selected network

***wpa_ie*** Buffer for the WPA/RSN IE

***wpa_ie_len*** Maximum wpa_ie buffer size on input. This is changed to be the used buffer length in case the functions returns success.

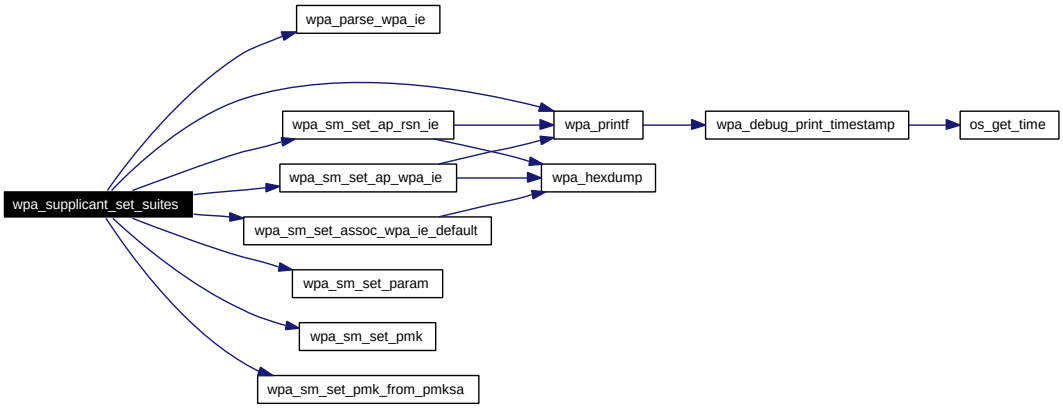
**Returns:**

0 on success or -1 on failure

This function is used to configure authentication and encryption parameters based on the network configuration and scan result for the selected BSS (if available).

Definition at line 1177 of file wpa_supplicant.c.

Here is the call graph for this function:




### 6.187.2.27 const char* wpa_supplicant_state_txt (int *state*)

Get the connection state name as a text string.

**Parameters:**

***state*** State (wpa_state; WPA_*)

**Returns:**

The state name as a printable text string

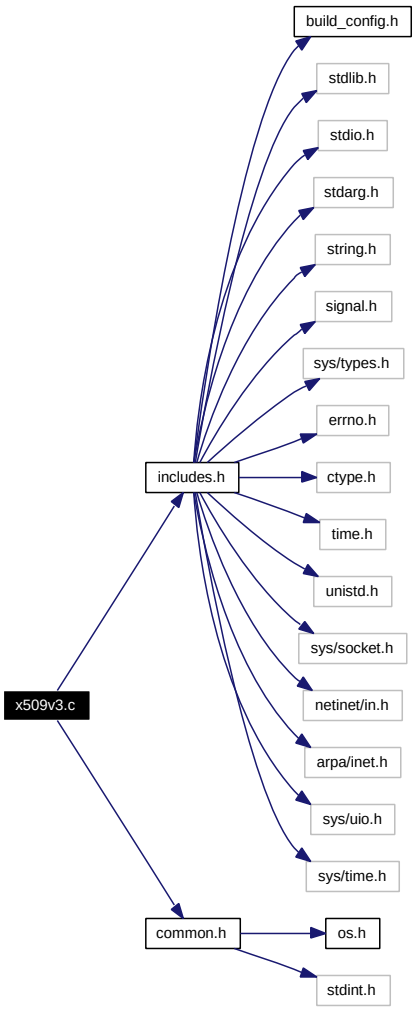
Definition at line 780 of file wpa_supplicant.c.

## 6.188 x509v3.c File Reference

X.509v3 certificate parsing and processing (RFC 3280 profile).

```
#include "includes.h"
#include "common.h"
```

Include dependency graph for x509v3.c:




### 6.188.1 Detailed Description

X.509v3 certificate parsing and processing (RFC 3280 profile).

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file x509v3.c.

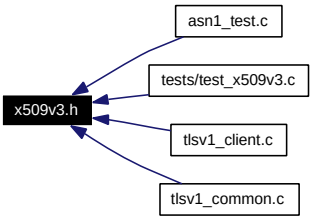
## 6.189 x509v3.h File Reference

X.509v3 certificate parsing and processing.

```
#include "asn1.h"
```

Include dependency graph for x509v3.h:


This graph shows which files directly or indirectly include this file:




### Defines

- #define **X509_EXT_BASIC_CONSTRAINTS** (1 << 0)
- #define **X509_EXT_PATH_LEN_CONSTRAINT** (1 << 1)
- #define **X509_EXT_KEY_USAGE** (1 << 2)
- #define **X509_KEY_USAGE_DIGITAL_SIGNATURE** (1 << 0)
- #define **X509_KEY_USAGE_NON_REPUDIATION** (1 << 1)
- #define **X509_KEY_USAGE_KEY_ENCIPHERMENT** (1 << 2)
- #define **X509_KEY_USAGE_DATA_ENCIPHERMENT** (1 << 3)
- #define **X509_KEY_USAGE_KEY_AGREEMENT** (1 << 4)
- #define **X509_KEY_USAGE_KEY_CERT_SIGN** (1 << 5)
- #define **X509_KEY_USAGE_CRL_SIGN** (1 << 6)
- #define **X509_KEY_USAGE_ENCIPHER_ONLY** (1 << 7)
- #define **X509_KEY_USAGE_DECIPHER_ONLY** (1 << 8)

### Enumerations

- enum {

  **X509_VALIDATE_OK**, **X509_VALIDATE_BAD_CERTIFICATE**, **X509_VALIDATE_-UNSUPPORTED_CERTIFICATE**, **X509_VALIDATE_CERTIFICATE_REVOKED**,

  **X509_VALIDATE_CERTIFICATE_EXPIRED**, **X509_VALIDATE_CERTIFICATE_-UNKNOWN**, **X509_VALIDATE_UNKNOWN_CA** }

### 6.189.1 Detailed Description

X.509v3 certificate parsing and processing.

**Copyright**

Copyright (c) 2006, Jouni Malinen <jkmaline@cc.hut.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Definition in file x509v3.h.

# Chapter 7

# wpa_supplicant Example Documentation

## 7.1 com

Semicolon separated string of entries to be matched against the alternative subject name of the authentication server certificate. If this string is set, the server sertificate is only accepted if it contains one of the entries in an alternative subject name extension.

altSubjectName string is in following format: TYPE:VALUE

Example: EMAIL:server Example: DNS:server.example.com;DNS:server2.example.com

Following types are supported: EMAIL, DNS, URI

# Chapter 8

# wpa_supplicant Page Documentation

## 8.1 Structure of the source code

[ wpa_supplicant core functionality | Generic helper functions | Cryptographic functions | Configuration | Control interface | WPA supplicant | EAP peer | EAPOL supplicant | Windows port | Test programs ]

wpa_supplicant implementation is divided into number of independent modules. Core code includes functionality for controlling the network selection, association, and configuration. Independent modules include WPA code (key handshake, PMKSA caching, pre-authentication), EAPOL state machine, and EAP state machine and methods. In addition, there are number of separate files for generic helper functions.

Both WPA and EAPOL/EAP state machines can be used separately in other programs than wpa_supplicant. As an example, the included test programs eapol_test and preauth_test are using these modules.

Driver interface API is defined in driver.h and all hardware/driver dependent functionality is implemented in driver_∗.c.

### 8.1.1 wpa_supplicant core functionality

wpa_supplicant.c Program initialization, main control loop

main.c main() for UNIX-like operating systems and MinGW (Windows); this uses command line arguments to configure wpa_supplicant

events.c Driver event processing; wpa_supplicant_event() and related functions

wpa_supplicant_i.h Internal definitions for wpa_supplicant core; should not be included into independent modules

wpa_supplicant.h Definitions for driver event data and message logging

### 8.1.2 Generic helper functions

wpa_supplicant uses generic helper functions some of which are shared with with hostapd. The following C files are currently used:

eloop.c and eloop.h Event loop (select() loop with registerable timeouts, socket read callbacks, and signal callbacks)

common.c and common.h Common helper functions

defs.h Definitions shared by multiple files

l2_packet.h, l2_packet_linux.c, and l2_packet_pcap.c Layer 2 (link) access wrapper (includes native Linux implementation and wrappers for libdnet/libpcap). A new l2_packet implementation may need to be added when porting to new operating systems that are not supported by libdnet/libpcap. Makefile can be used to select which l2_packet implementation is included. l2_packet_linux.c uses Linux packet sockets and l2_packet_pcap.c has a more portable version using libpcap and libdnet.

pcsc_funcs.c and pcsc_funcs.h Wrapper for PC/SC lite SIM and smart card readers

priv_netlink.h Private version of netlink definitions from Linux kernel header files; this could be replaced with C library header file once suitable version becomes commonly available

version.h Version number definitions

wireless_copy.h Private version of Linux wireless extensions definitions from kernel header files; this could be replaced with C library header file once suitable version becomes commonly available

### 8.1.3 Cryptographic functions

md5.c and md5.h MD5 (replaced with a crypto library if TLS support is included) HMAC-MD5 (keyed checksum for message authenticity validation)

rc4.c and rc4.h RC4 (broadcast/default key encryption)

sha1.c and sha1.h SHA-1 (replaced with a crypto library if TLS support is included) HMAC-SHA-1 (keyed checksum for message authenticity validation) PRF-SHA-1 (pseudorandom (key/nonce generation) function) PBKDF2-SHA-1 (ASCII passphrase to shared secret) T-PRF (for EAP-FAST) TLS-PRF (RFC 2246)

aes_wrap.c, aes_wrap.h, aes.c AES (replaced with a crypto library if TLS support is included), AES Key Wrap Algorithm with 128-bit KEK, RFC3394 (broadcast/default key encryption), One-Key CBC MAC (OMAC1) hash with AES-128, AES-128 CTR mode encryption, AES-128 EAX mode encryption/decryption, AES-128 CBC

crypto.h Definition of crypto library wrapper

crypto.c Wrapper functions for libcrypto (OpenSSL)

crypto_gnutls.c Wrapper functions for libgcrypt (used by GnuTLS)

ms_funcs.c and ms_funcs.h Helper functions for MSCHAPV2 and LEAP

tls.h Definition of TLS library wrapper

tls_none.c Dummy implementation of TLS library wrapper for cases where TLS functionality is not included.

tls_openssl.c TLS library wrapper for openssl

tls_gnutls.c TLS library wrapper for GnuTLS

### 8.1.4 Configuration

config_ssid.h Definition of per network configuration items

config.h Definition of the wpa_supplicant configuration

config.c Configuration parser and common functions

config_file.c Configuration backend for text files (e.g., wpa_supplicant.conf)

### 8.1.5 Control interface

wpa_supplicant has a control interface that can be used to get status information and manage operations from external programs. An example command line interface (wpa_cli) and GUI (wpa_gui) for this interface are included in the wpa_supplicant distribution.

ctrl_iface.c and ctrl_iface.h wpa_supplicant-side of the control interface

wpa_ctrl.c and wpa_ctrl.h Library functions for external programs to provide access to the wpa_supplicant control interface

wpa_cli.c Example program for using wpa_supplicant control interface

### 8.1.6 WPA supplicant

wpa.c and wpa.h WPA state machine and 4-Way/Group Key Handshake processing

preauth.c and preauth.h PMKSA caching and pre-authentication (RSN/WPA2)

wpa_i.h Internal definitions for WPA code; not to be included to other modules.

### 8.1.7 EAP peer

EAP peer implementation is a separate module that can be used by other programs than just wpa_supplicant.

eap.c and eap.h EAP state machine and method interface

eap_defs.h Common EAP definitions

eap_i.h Internal definitions for EAP state machine and EAP methods; not to be included in other modules

eap_sim_common.c and eap_sim_common.h Common code for EAP-SIM and EAP-AKA

eap_tls_common.c and eap_tls_common.h Common code for EAP-PEAP, EAP-TTLS, and EAP-FAST

eap_tlv.c and eap_tlv.h EAP-TLV code for EAP-PEAP and EAP-FAST

eap_ttls.c and eap_ttls.h EAP-TTLS

eap_pax.c, eap_pax_common.h, eap_pax_common.c EAP-PAX

eap_psk.c, eap_psk_common.h, eap_psk_common.c EAP-PSK (note: this is not needed for WPA-PSK)

eap_sake.c, eap_sake_common.h, eap_sake_common.c EAP-SAKE

eap_gpsk.c, eap_gpsk_common.h, eap_gpsk_common.c EAP-GPSK

eap_aka.c, eap_fast.c, eap_gtc.c, eap_leap.c, eap_md5.c, eap_mschapv2.c, eap_otp.c, eap_peap.c, eap_sim.c, eap_tls.c Other EAP method implementations

### 8.1.8 EAPOL supplicant

eapol_sm.c and eapol_sm.h EAPOL supplicant state machine and IEEE 802.1X processing

### 8.1.9 Windows port

ndis_events.cpp External program for receiving NdisMIndicateStatus() events and delivering them to wpa_supplicant in more easier to use form

win_if_list.c External program for listing current network interface

### 8.1.10 Test programs

radius_client.c and radius_client.h RADIUS authentication client implementation for eapol_test

radius.c and radius.h RADIUS message processing for eapol_test

config_types.h and hostapd.h Minimal version of hostapd header files for eapol_test

eapol_test.c Standalone EAP testing tool with integrated RADIUS authentication client

preauth_test.c Standalone RSN pre-authentication tool

wpa_passphrase.c WPA ASCII passphrase to PSK conversion

## 8.2 Control interface

wpa_supplicant implements a control interface that can be used by external programs to control the operations of the wpa_supplicant daemon and to get status information and event notifications. There is a small C library, in a form of a single C file, wpa_ctrl.c, that provides helper functions to facilitate the use of the control interface. External programs can link this file into them and then use the library functions documented in wpa_ctrl.h to interact with wpa_supplicant. This library can also be used with C++. wpa_cli.c and wpa_gui are example programs using this library.

There are multiple mechanisms for inter-process communication. For example, Linux version of wpa_supplicant is using UNIX domain sockets for the control interface and Windows version UDP sockets. The use of the functions defined in wpa_ctrl.h can be used to hide the details of the used IPC from external programs.

### 8.2.1 Using the control interface

External programs, e.g., a GUI or a configuration utility, that need to communicate with wpa_supplicant should link in wpa_ctrl.c. This allows them to use helper functions to open connection to the control interface with wpa_ctrl_open() and to send commands with wpa_ctrl_request().

wpa_supplicant uses the control interface for two types of communication: commands and unsolicited event messages. Commands are a pair of messages, a request from the external program and a response from wpa_supplicant. These can be executed using wpa_ctrl_request(). Unsolicited event messages are sent by wpa_supplicant to the control interface connection without specific request from the external program for receiving each message. However, the external program needs to attach to the control interface with wpa_ctrl_attach() to receive these unsolicited messages.

If the control interface connection is used both for commands and unsolicited event messages, there is potential for receiving an unsolicited message between the command request and response. wpa_ctrl_request() caller will need to supply a callback, msg_cb, for processing these messages. Often it is easier to open two control interface connections by calling wpa_ctrl_open() twice and then use one of the connections for commands and the other one for unsolicited messages. This way command request/response pairs will not be broken by unsolicited messages. wpa_cli is an example of how to use only one connection for both purposes and wpa_gui demonstrates how to use two separate connections.

Once the control interface connection is not needed anymore, it should be closed by calling wpa_ctrl_close(). If the connection was used for unsolicited event messages, it should be first detached by calling wpa_ctrl_detach().

### 8.2.2 Control interface commands

Following commands can be used with wpa_ctrl_request():

#### 8.2.2.1 PING

This command can be used to test whether wpa_supplicant is replying to the control interface commands. The expected reply is `PONG` if the connection is open and wpa_supplicant is processing commands.

#### 8.2.2.2 MIB

Request a list of MIB variables (dot1x, dot11). The output is a text block with each line in `variable=value` format. For example:

```
dot11RSNAOptionImplemented=TRUE
dot11RSNAPreauthenticationImplemented=TRUE
dot11RSNAEnabled=FALSE
dot11RSNAPreauthenticationEnabled=FALSE
dot11RSNAConfigVersion=1
dot11RSNAConfigPairwiseKeysSupported=5
dot11RSNAConfigGroupCipherSize=128
dot11RSNAConfigPMKLifetime=43200
dot11RSNAConfigPMKReauthThreshold=70
dot11RSNAConfigNumberOfPTKSAReplayCounters=1
dot11RSNAConfigSATimeout=60
dot11RSNAAuthenticationSuiteSelected=00-50-f2-2
dot11RSNAPairwiseCipherSelected=00-50-f2-4
dot11RSNAGroupCipherSelected=00-50-f2-4
dot11RSNAPMKIDUsed=
dot11RSNAAuthenticationSuiteRequested=00-50-f2-2
dot11RSNAPairwiseCipherRequested=00-50-f2-4
dot11RSNAGroupCipherRequested=00-50-f2-4
dot11RSNAConfigNumberOfGTKSAReplayCounters=0
dot11RSNA4WayHandshakeFailures=0
dot1xSuppPaeState=5
dot1xSuppHeldPeriod=60
dot1xSuppAuthPeriod=30
dot1xSuppStartPeriod=30
dot1xSuppMaxStart=3
dot1xSuppSuppControlledPortStatus=Authorized
dot1xSuppBackendPaeState=2
dot1xSuppEapolFramesRx=0
dot1xSuppEapolFramesTx=440
dot1xSuppEapolStartFramesTx=2
dot1xSuppEapolLogoffFramesTx=0
dot1xSuppEapolRespFramesTx=0
dot1xSuppEapolReqIdFramesRx=0
dot1xSuppEapolReqFramesRx=0
dot1xSuppInvalidEapolFramesRx=0
dot1xSuppEapLengthErrorFramesRx=0
dot1xSuppLastEapolFrameVersion=0
dot1xSuppLastEapolFrameSource=00:00:00:00:00:00
```

#### 8.2.2.3 STATUS

Request current WPA/EAPOL/EAP status information. The output is a text block with each line in `variable=value` format. For example:

```
bssid=02:00:01:02:03:04
ssid=test network
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA-PSK
wpa_state=COMPLETED
ip_address=192.168.1.21
Supplicant PAE state=AUTHENTICATED
suppPortStatus=Authorized
EAP state=SUCCESS
```

#### 8.2.2.4 STATUS-VERBOSE

Same as STATUS, but with more verbosity (i.e., more `variable=value` pairs).

```
bssid=02:00:01:02:03:04
ssid=test network
id=0
```

```
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA-PSK
wpa_state=COMPLETED
ip_address=192.168.1.21
Supplicant PAE state=AUTHENTICATED
suppPortStatus=Authorized
heldPeriod=60
authPeriod=30
startPeriod=30
maxStart=3
portControl=Auto
Supplicant Backend state=IDLE
EAP state=SUCCESS
reqMethod=0
methodState=NONE
decision=COND_SUCC
ClientTimeout=60
```

#### 8.2.2.5 PMKSA

Show PMKSA cache

```
Index / AA / PMKID / expiration (in seconds) / opportunistic
1 / 02:00:01:02:03:04 / 000102030405060708090a0b0c0d0e0f / 41362 / 0
2 / 02:00:01:33:55:77 / 928389281928383b34afb34ba4212345 / 362 / 1
```

#### 8.2.2.6 SET <variable> <value>

Set variables:

- EAPOL::heldPeriod
- EAPOL::authPeriod
- EAPOL::startPeriod
- EAPOL::maxStart
- dot11RSNAConfigPMKLifetime
- dot11RSNAConfigPMKReauthThreshold
- dot11RSNAConfigSATimeout

Example command:

```
SET EAPOL::heldPeriod 45
```

#### 8.2.2.7 LOGON

IEEE 802.1X EAPOL state machine logon.

#### 8.2.2.8 LOGOFF

IEEE 802.1X EAPOL state machine logoff.

#### 8.2.2.9 REASSOCIATE

Force reassociation.

#### 8.2.2.10 PREAUTH <BSSID>

Start pre-authentication with the given BSSID.

#### 8.2.2.11 ATTACH

Attach the connection as a monitor for unsolicited events. This can be done with wpa_ctrl_attach().

#### 8.2.2.12 DETACH

Detach the connection as a monitor for unsolicited events. This can be done with wpa_ctrl_detach().

#### 8.2.2.13 LEVEL <debug level>

Change debug level.

#### 8.2.2.14 RECONFIGURE

Force wpa_supplicant to re-read its configuration data.

#### 8.2.2.15 TERMINATE

Terminate wpa_supplicant process.

#### 8.2.2.16 BSSID <network id> <BSSID>

Set preferred BSSID for a network. Network id can be received from the `LIST_NETWORKS` command output.

#### 8.2.2.17 LIST_NETWORKS

List configured networks.

```
network id / ssid / bssid / flags
0 example network any [CURRENT]
```

(note: fields are separated with tabs)

#### 8.2.2.18 DISCONNECT

Disconnect and wait for `REASSOCIATE` command before connecting.

#### 8.2.2.19 SCAN

Request a new BSS scan.

#### 8.2.2.20 SCAN_RESULTS

Get the latest scan results.

```
bssid / frequency / signal level / flags / ssid
00:09:5b:95:e0:4e 2412 208 [WPA-PSK-CCMP] jkm private
02:55:24:33:77:a3 2462 187 [WPA-PSK-TKIP] testing
00:09:5b:95:e0:4f 2412 209 jkm guest
```

(note: fields are separated with tabs)

#### 8.2.2.21 SELECT_NETWORK <network id>

Select a network (disable others). Network id can be received from the LIST_NETWORKS command output.

#### 8.2.2.22 ENABLE_NETWORK <network id>

Enable a network. Network id can be received from the LIST_NETWORKS command output.

#### 8.2.2.23 DISABLE_NETWORK <network id>

Disable a network. Network id can be received from the LIST_NETWORKS command output.

#### 8.2.2.24 ADD_NETWORK

Add a new network. This command creates a new network with empty configuration. The new network is disabled and once it has been configured it can be enabled with ENABLE_NETWORK command. ADD_NETWORK returns the network id of the new network or FAIL on failure.

#### 8.2.2.25 REMOVE_NETWORK <network id>

Remove a network. Network id can be received from the LIST_NETWORKS command output.

#### 8.2.2.26 SET_NETWORK <network id> <variable> <value>

Set network variables. Network id can be received from the LIST_NETWORKS command output.

This command uses the same variables and data formats as the configuration file. See example wpa_supplicant.conf for more details.

- ssid (network name, SSID)
- psk (WPA passphrase or pre-shared key)
- key_mgmt (key management protocol)

- identity (EAP identity)
- password (EAP password)
- ...

#### 8.2.2.27 GET_NETWORK <network id> <variable>

Get network variables. Network id can be received from the `LIST_NETWORKS` command output.

#### 8.2.2.28 SAVE_CONFIG

Save the current configuration.

### 8.2.3 Interactive requests

If wpa_supplicant needs additional information during authentication (e.g., password), it will use a specific prefix, `CTRL-REQ-` (*WPA_CTRL_REQ* macro) in an unsolicited event message. An external program, e.g., a GUI, can provide such information by using `CTRL-RSP-` (*WPA_CTRL_RSP* macro) prefix in a command with matching field name.

The following fields can be requested in this way from the user:

- IDENTITY (EAP identity/user name)
- PASSWORD (EAP password)
- NEW_PASSWORD (New password if the server is requesting password change)
- PIN (PIN code for accessing a SIM or smartcard)
- OTP (one-time password; like password, but the value is used only once)
- PASSPHRASE (passphrase for a private key file)

```
CTRL-REQ-<field name>-<network id>-<human readable text>
CTRL-RSP-<field name>-<network id>-<value>
```

For example, request from wpa_supplicant:

```
CTRL-REQ-PASSWORD-1-Password needed for SSID test-network
```

And a matching reply from the GUI:

```
CTRL-RSP-PASSWORD-1-secret
```

#### 8.2.3.1 GET_CAPABILITY <option> [strict]

Get list of supported functionality (eap, pairwise, group, proto). Supported functionality is shown as space separate lists of values used in the same format as in wpa_supplicant configuration. If optional argument, 'strict', is added, only the values that the driver claims to explicitly support are included. Without this, all available capabilities are included if the driver does not provide a mechanism for querying capabilities.

Example request/reply pairs:

```
GET_CAPABILITY eap
AKA FAST GTC LEAP MD5 MSCHAPV2 OTP PAX PEAP PSK SIM TLS TTLS


GET_CAPABILITY pairwise
CCMP TKIP NONE


GET_CAPABILITY pairwise strict


GET_CAPABILITY group
CCMP TKIP WEP104 WEP40


GET_CAPABILITY key_mgmt
WPA-PSK WPA-EAP IEEE8021X NONE


GET_CAPABILITY proto
RSN WPA


GET_CAPABILITY auth_alg
OPEN SHARED LEAP
```

#### 8.2.3.2 AP_SCAN <ap_scan value>

Change ap_scan value: 0 = no scanning, 1 = wpa_supplicant requests scans and uses scan results to select the AP, 2 = wpa_supplicant does not use scanning and just requests driver to associate and take care of AP selection

#### 8.2.3.3 INTERFACES

List configured interfaces.

```
wlan0
eth0
```

## 8.3 Driver wrapper implementation (driver.h, drivers.c)

All hardware and driver dependent functionality is in separate C files that implement defined wrapper functions. Other parts of the wpa_supplicant are designed to be hardware, driver, and operating system independent.

Driver wrappers need to implement whatever calls are used in the target operating system/driver for controlling wireless LAN devices. As an example, in case of Linux, these are mostly some glue code and ioctl() calls and netlink message parsing for Linux Wireless Extensions (WE). Since features required for WPA were added only recently to Linux Wireless Extensions (in version 18), some driver specific code is used in number of driver interface implementations. These driver dependent parts can be replaced with generic code in driver_wext.c once the target driver includes full support for WE-18. After that, all Linux drivers, at least in theory, could use the same driver wrapper code.

A driver wrapper needs to implement some or all of the functions defined in driver.h. These functions are registered by filling struct wpa_driver_ops with function pointers. Hardware independent parts of wpa_supplicant will call these functions to control the driver/wlan card. In addition, support for driver events is required. The event callback function, wpa_supplicant_event(), and its parameters are documented in wpa_supplicant.h. In addition, a pointer to the 'struct wpa_driver_ops' needs to be registered in drivers.c file.

When porting to other operating systems, the driver wrapper should be modified to use the native interface of the target OS. It is possible that some extra requirements for the interface between the driver wrapper and generic wpa_supplicant code are discovered during porting to a new operating system. These will be addressed on case by case basis by modifying the interface and updating the other driver wrappers for this. The goal is to avoid changing this interface without very good reasons in order to limit the number of changes needed to other wrappers and hardware independent parts of wpa_supplicant. When changes are required, recommended way is to make them in backwards compatible way that allows existing driver interface implementations to be compiled without any modification.

Generic Linux Wireless Extensions functions are implemented in driver_wext.c. All Linux driver wrappers can use these when the kernel driver supports the generic ioctl()s and wireless events. Driver specific functions are implemented in separate C files, e.g., driver_hostap.c. These files need to define struct wpa_driver_ops entry that will be used in wpa_supplicant.c when calling driver functions. struct wpa_driver_ops entries are registered in drivers.c.

In general, it is likely to be useful to first take a look at couple of driver interface examples before starting on implementing a new one. driver_hostap.c and driver_wext.c include a complete implementation for Linux drivers that use wpa_supplicant-based control of WPA IE and roaming. driver_ndis.c (with help from driver_ndis_.c) is an example of a complete interface for Windows NDIS interface for drivers that generate WPA IE themselves and decide when to roam. These example implementations include full support for all security modes.

### 8.3.1 Driver requirements for WPA

WPA introduces new requirements for the device driver. At least some of these need to be implemented in order to provide enough support for wpa_supplicant.

#### 8.3.1.1 TKIP/CCMP

WPA requires that the pairwise cipher suite (encryption algorithm for unicast data packets) is TKIP or CCMP. These are new encryption protocols and thus, the driver will need to be modified to support them. Depending on the used wlan hardware, some parts of these may be implemented by the hardware/firmware.

Specification for both TKIP and CCMP is available from IEEE (IEEE 802.11i amendment). Fully func-

tional, hardware independent implementation of both encryption protocols is also available in Host AP driver (driver/modules/hostap_{tkip,ccmp}.c). In addition, Linux 2.6 kernel tree has generic implementations for WEP, TKIP, and CCMP that can be used in Linux drivers.

The driver will also need to provide configuration mechanism to allow user space programs to configure TKIP and CCMP. Linux Wireless Extensions v18 added support for configuring these algorithms and individual/non-default keys. If the target kernel does not include WE-18, private ioctls can be used to provide similar functionality.

#### 8.3.1.2 Roaming control and scanning support

wpa_supplicant can optionally control AP selection based on the information received from Beacon and/or Probe Response frames (ap_scan=1 mode in configuration). This means that the driver should support external control for scan process. In case of Linux, use of new Wireless Extensions scan support (i.e., 'iwlist wlan0 scan') is recommended. The current driver wrapper (driver_wext.c) uses this for scan results.

Scan results must also include the WPA information element. Support for this was added in WE-18. With older versions, a custom event can be used to provide the full WPA IE (including element id and length) as a hex string that is included in the scan results.

wpa_supplicant needs to also be able to request the driver to associate with a specific BSS. Current Host AP driver and matching driver_hostap.c wrapper uses following sequence for this request. Similar/identical mechanism should be usable also with other drivers.

- set WPA IE for AssocReq with private ioctl
- set SSID with SIOCSIWESSID
- set channel/frequency with SIOCSIWFREQ
- set BSSID with SIOCSIWAP (this last ioctl will trigger the driver to request association)

#### 8.3.1.3 WPA IE generation

wpa_supplicant selects which cipher suites and key management suites are used. Based on this information, it generates a WPA IE. This is provided to the driver interface in the associate call. This does not match with Windows NDIS drivers which generate the WPA IE themselves.

wpa_supplicant allows Windows NDIS-like behavior by providing the selected cipher and key management suites in the associate call. If the driver generates its own WPA IE and that differs from the one generated by wpa_supplicant, the driver has to inform wpa_supplicant about the used WPA IE (i.e., the one it used in (Re)Associate Request). This notification is done using EVENT_ASSOCINFO event (see wpa_supplicant.h). wpa_supplicant is normally configured to use ap_scan=2 mode with drivers that control WPA IE generation and roaming.

#### 8.3.1.4 Driver events

wpa_supplicant needs to receive event callbacks when certain events occur (association, disassociation, Michael MIC failure, scan results available, PMKSA caching candidate). These events and the callback details are defined in wpa_supplicant.h (wpa_supplicant_event() function and enum wpa_event_type).

On Linux, association and disassociation can use existing Wireless Extensions event that is reporting new AP with SIOCGIWAP event. Similarly, completion of a scan can be reported with SIOCGIWSCAN event.

Michael MIC failure event was added in WE-18. Older versions of Wireless Extensions will need to use a custom event. Host AP driver used a custom event with following contents: MLME-MICHAELMICFAILURE.indication(keyid=# broadcast/unicast addr=addr2). This is the recommended format until the driver can be moved to use WE-18 mechanism.

#### 8.3.1.5 Summary of Linux Wireless Extensions use

AP selection depends on ap_scan configuration:

ap_scan=1:

- wpa_supplicant requests scan with SIOCSIWSCAN
- driver reports scan complete with wireless event SIOCGIWSCAN
- wpa_supplicant reads scan results with SIOCGIWSCAN (multiple call if a larget buffer is needed)
- wpa_supplicant decides which AP to use based on scan results
- wpa_supplicant configures driver to associate with the selected BSS (SIOCSIWMODE, SIOCSIWGENIE, SIOCSIWAUTH, SIOCSIWFREQ, SIOCSIWESSID, SIOCSIWAP)

ap_scan=2:

- wpa_supplicant configures driver to associate with an SSID (SIOCSIWMODE, SIOCSIWGENIE, SIOCSIWAUTH, SIOCSIWESSID)

After this, both modes use similar steps:

- optionally (or required for drivers that generate WPA/RSN IE for (Re)AssocReq), driver reports association parameters (AssocReq IEs) with wireless event IWEVASSOCREQIE (and optionally IWEVASSOCRESPIE)
- driver reports association with wireless event SIOCGIWAP
- wpa_supplicant takes care of EAPOL frame handling (validating information from associnfo and if needed, from scan results if WPA/RSN IE from the Beacon frame is not reported through associnfo)

## 8.4 EAP peer implementation

Extensible Authentication Protocol (EAP) is an authentication framework defined in RFC 3748. wpa_supplicant uses a separate code module for EAP peer implementation. This module was designed to use only a minimal set of direct function calls (mainly, to debug/event functions) in order for it to be usable in other programs. The design of the EAP implementation is based loosely on RFC 4137. The state machine is defined in this RFC and so is the interface between the peer state machine and methods. As such, this RFC provides useful information for understanding the EAP peer implementation in wpa_supplicant.

Some of the terminology used in EAP state machine is referring to EAPOL (IEEE 802.1X), but there is no strict requirement on the lower layer being IEEE 802.1X if EAP module is built for other programs than wpa_supplicant. These terms should be understood to refer to the lower layer as defined in RFC 4137.

### 8.4.1 Adding EAP methods

Each EAP method is implemented as a separate module, usually as one C file named eap_<name of the method>.c, e.g., eap_md5.c. All EAP methods use the same interface between the peer state machine and method specific functions. This allows new EAP methods to be added without modifying the core EAP state machine implementation.

New EAP methods need to be registered by adding them into the build (Makefile) and the EAP method registration list in the eap_peer_register_methods() function of eap_methods.c. Each EAP method should use a build-time configuration option, e.g., EAP_TLS, in order to make it possible to select which of the methods are included in the build.

EAP methods must implement the interface defined in eap_i.h. struct eap_method defines the needed function pointers that each EAP method must provide. In addition, the EAP type and name are registered using this structure. This interface is based on section 4.4 of RFC 4137.

It is recommended that the EAP methods would use generic helper functions, eap_msg_alloc() and eap_hdr_validate() when processing messages. This allows code sharing and can avoid missing some of the needed validation steps for received packets. In addition, these functions make it easier to change between expanded and legacy EAP header, if needed.

When adding an EAP method that uses a vendor specific EAP type (Expanded Type as defined in RFC 3748, Chapter 5.7), the new method must be registered by passing vendor id instead of EAP_VENDOR_IETF to eap_peer_method_alloc(). These methods must not try to emulate expanded types by registering a legacy EAP method for type 254. See eap_vendor_test.c for an example of an EAP method implementation that is implemented as an expanded type.

## 8.5 Porting to different target boards and operating systems

wpa_supplicant was designed to be easily portable to different hardware (board, CPU) and software (OS, drivers) targets. It is already used with number of operating systems and numerous wireless card models and drivers. The main wpa_supplicant repository includes support for Linux, FreeBSD, and Windows. In addition, at least VxWorks and PalmOS are supported in separate repositories. On the hardware side, wpa_supplicant is used on various systems: desktops, laptops, PDAs, and embedded devices with CPUs including x86, PowerPC, arm/xscale, and MIPS. Both big and little endian configurations are supported.

### 8.5.1 Extra functions on top of ANSI C

wpa_supplicant is mostly using ANSI C functions that are available on most targets. However, couple of additional functions that are common on modern UNIX systems are used. Number of these are listed with prototypes in common.h (the #ifdef CONFIG_ANSI_C_EXTRA block). These functions may need to be implemented or at least defined as macros to native functions in the target OS or C library.

### 8.5.2 Driver interface

Unless the target OS and driver is already supported, most porting projects have to implement a driver wrapper. This may be done by adding a new driver interface module or modifying an existing module (driver_∗.c) if the new target is similar to one of them. Driver wrapper implementation describes the details of the driver interface and discusses the tasks involved in porting this part of wpa_supplicant.

### 8.5.3 l2_packet (link layer access)

wpa_supplicant needs to have access to sending and receiving layer 2 (link layer) packets with two Ethertypes: EAP-over-LAN (EAPOL) 0x888e and RSN pre-authentication 0x88c7. l2_packet.h defines the interfaces used for this in the core wpa_supplicant implementation.

If the target operating system supports a generic mechanism for link layer access, that is likely the best mechanism for providing the needed functionality for wpa_supplicant. Linux packet socket is an example of such a generic mechanism. If this is not available, a separate interface may need to be implemented to the network stack or driver. This is usually an intermediate or protocol driver that is operating between the device driver and the OS network stack. If such a mechanism is not feasible, the interface can also be implemented directly in the device driver.

The main wpa_supplicant repository includes l2_packet implementations for Linux using packet sockets (l2_packet_linux.c), more portable version using libpcap/libdnet libraries (l2_packet_pcap.c; this supports WinPcap, too), and FreeBSD specific version of libpcap interface (l2_packet_freebsd.c).

If the target operating system is supported by libpcap (receiving) and libdnet (sending), l2_packet_pcap.c can likely be used with minimal or no changes. If this is not a case or a proprietary interface for link layer is required, a new l2_packet module may need to be added. Alternatively, struct wpa_driver_ops::send_eapol() handler can be used to override the l2_packet library if the link layer access is integrated with the driver interface implementation.

### 8.5.4 Event loop

wpa_supplicant uses a single process/thread model and an event loop to provide callbacks on events (registered timeout, received packet, signal). eloop.h defines the event loop interface. eloop.c is an implementation of such an event loop using select() and sockets. This is suitable for most UNIX/POSIX systems.

When porting to other operating systems, it may be necessary to replace that implementation with OS specific mechanisms that provide similar functionality.

### 8.5.5 Control interface

wpa_supplicant uses a control interface to allow external processed to get status information and to control the operations. Currently, this is implemented with socket based communication; both UNIX domain sockets and UDP sockets are supported. If the target OS does not support sockets, this interface will likely need to be modified to use another mechanism like message queues. The control interface is optional component, so it is also possible to run wpa_supplicant without porting this part.

The wpa_supplicant side of the control interface is implemented in ctrl_iface.c. Matching client side is implemented as a control interface library in wpa_ctrl.c.

### 8.5.6 Program entry point

wpa_supplicant defines a set of functions that can be used to initialize main supplicant processing. Each operating system has a mechanism for starting new processing or threads. This is usually a function with a specific set of arguments and calling convention. This function is responsible on initializing wpa_-supplicant.

main.c includes an entry point for UNIX-like operating system, i.e., main() function that uses command line arguments for setting parameters for wpa_supplicant. When porting to other operating systems, similar OS-specific entry point implementation is needed. It can be implemented in a new file that is then linked with wpa_supplicant instead of main.o. main.c is also a good example on how the initialization process should be done.

The supplicant initialization functions are defined in wpa_supplicant_i.h. In most cases, the entry point function should start by fetching configuration parameters. After this, a global wpa_supplicant context is initialized with a call to wpa_supplicant_init(). After this, existing network interfaces can be added with wpa_supplicant_add_iface(). wpa_supplicant_run() is then used to start the main event loop. Once this returns at program termination time, wpa_supplicant_deinit() is used to release global context data.

wpa_supplicant_add_iface() and wpa_supplicant_remove_iface() can be used dynamically to add and remove interfaces based on when wpa_supplicant processing is needed for them. This can be done, e.g., when hotplug network adapters are being inserted and ejected. It is also possible to do this when a network interface is being enabled/disabled if it is desirable that wpa_supplicant processing for the interface is fully enabled/disabled at the same time.

### 8.5.7 Simple build example

One way to start a porting project is to begin with a very simple build of wpa_supplicant with WPA-PSK support and once that is building correctly, start adding features.

Following command can be used to build very simple version of wpa_supplicant:

```
cc -o wpa_supplicant config.c eloop.c common.c md5.c rc4.c sha1.c \
config_none.c l2_packet_none.c tls_none.c wpa.c preauth.c \
aes_wrap.c wpa_supplicant.c events.c main_none.c drivers.c
```

The end result is not really very useful since it uses empty functions for configuration parsing and layer 2 packet access and does not include a driver interface. However, this is a good starting point since the build is complete in the sense that all functions are present and this is easy to configure to a build system by just including the listed C files.

Once this version can be build successfully, the end result can be made functional by adding a proper program entry point (main∗.c), driver interface (driver_∗.c and matching CONFIG_DRIVER_∗ define for registration in drivers.c), configuration parser/writer (config_∗.c), and layer 2 packet access implementation (l2_packet_∗.c). After these components have been added, the end result should be a working WPA/WPA2-PSK enabled supplicant.

After the basic functionality has been verified to work, more features can be added by linking in more files and defining C pre-processor defines. Currently, the best source of information for what options are available and which files needs to be included is in the Makefile used for building the supplicant with make. Similar configuration will be needed for build systems that either use different type of make tool or a GUI-based project configuration.

## 8.6 Testing and development tools

[ eapol_test | preauth_test | driver_test | Unit tests ]

wpa_supplicant source tree includes number of testing and development tools that make it easier to test the programs without having to setup a full test setup with wireless cards. In addition, these tools can be used to implement automatic tests suites.

### 8.6.1 eapol_test - EAP peer and RADIUS client testing

eapol_test is a program that links together the same EAP peer implementation that wpa_supplicant is using and the RADIUS authentication client code from hostapd. In addition, it has minimal glue code to combine these two components in similar ways to IEEE 802.1X/EAPOL Authenticator state machines. In other words, it integrates IEEE 802.1X Authenticator (normally, an access point) and IEEE 802.1X Supplicant (normally, a wireless client) together to generate a single program that can be used to test EAP methods without having to setup an access point and a wireless client.

The main uses for eapol_test are in interoperability testing of EAP methods against RADIUS servers and in development testing for new EAP methods. It can be easily used to automate EAP testing for interoperability and regression since the program can be run from shell scripts without require additional test components apart from a RADIUS server. For example, the automated EAP tests described in eap_testing.txt are implemented with eapol_test. Similarly, eapol_test could be used to implement an automated regression test suite for a RADIUS authentication server.

eapol_test uses the same build time configuration file, .config, as wpa_supplicant. This file is used to select which EAP methods are included in eapol_test. This program is not built with the default Makefile target, so a separate make command needs to be used to compile the tool:

```
make eapol_test
```

The resulting eapol_test binary has following command like options:

```
usage:
eapol_test [-nWS] -c<conf> [-a<AS IP>] [-p<AS port>] [-s<AS secret>] \
           [-r<count>] [-t<timeout>] [-C<Connect-Info>] \
           [-M<client MAC address>]
eapol_test scard
eapol_test sim <PIN> <num triplets> [debug]

options:
  -c<conf> = configuration file
  -a<AS IP> = IP address of the authentication server, default 127.0.0.1
  -p<AS port> = UDP port of the authentication server, default 1812
  -s<AS secret> = shared secret with the authentication server, default 'radius'
  -r<count> = number of re-authentications
  -W = wait for a control interface monitor before starting
  -S = save configuration after authentiation
  -n = no MPPE keys expected
  -t<timeout> = sets timeout in seconds (default: 30 s)
  -C<Connect-Info> = RADIUS Connect-Info (default: CONNECT 11Mbps 802.11b)
  -M<client MAC address> = Set own MAC address (Calling-Station-Id,
                           default: 02:00:00:00:00:01)
```

As an example,

```
eapol_test -ctest.conf -a127.0.0.1 -p1812 -ssecret -r1
```

tries to complete EAP authentication based on the network configuration from test.conf against the RADIUS server running on the local host. A re-authentication is triggered to test fast re-authentication. The configuration file uses the same format for network blocks as wpa_supplicant.

### 8.6.2 preauth_test - WPA2 pre-authentication and EAP peer testing

preauth_test is similar to eapol_test in the sense that in combines EAP peer implementation with something else, in this case, with WPA2 pre-authentication. This tool can be used to test pre-authentication based on the code that wpa_supplicant is using. As such, it tests both the wpa_supplicant implementation and the functionality of an access point.

preauth_test is built with:

```
make preauth_test
```

and it uses following command line arguments:

```
usage: preauth_test <conf> <target MAC address> <ifname>
```

For example,

```
preauth_test test.conf 02:11:22:33:44:55 eth0
```

would use network configuration from test.conf to try to complete pre-authentication with AP using BSSID 02:11:22:33:44:55. The pre-authentication packets would be sent using the eth0 interface.

### 8.6.3 driver_test - driver interface for testing wpa_supplicant

wpa_supplicant was designed to support number of different ways to communicate with a network device driver. This design uses driver interface API and number of driver interface implementations. One of these is driver_test.c, i.e., a test driver interface that is actually not using any drivers. Instead, it provides a mechanism for running wpa_supplicant without having to have a device driver or wireless LAN hardware for that matter.

driver_test can be used to talk directly with hostapd's driver_test component to create a test setup where one or more clients and access points can be tested within one test host and without having to have multiple wireless cards. This makes it easier to test the core code in wpa_supplicant, and hostapd for that matter. Since driver_test uses the same driver API than any other driver interface implementation, the core code of wpa_supplicant and hostapd can be tested with the same coverage as one would get when using real wireless cards. The only area that is not tested is the driver interface implementation (driver_∗.c).

Having the possibility to use simulated network components makes it much easier to do development testing while adding new features and to reproduce reported bugs. As such, it is often easiest to just do most of the development and bug fixing without using real hardware. Once the driver_test setup has been used to implement a new feature or fix a bug, the end result can be verified with wireless LAN cards. In many cases, this may even be unnecessary, depending on what area the feature/bug is relating to. Of course, changes to driver interfaces will still require use of real hardware.

Since multiple components can be run within a single host, testing of complex network configuration, e.g., large number of clients association with an access point, becomes quite easy. All the tests can also be automated without having to resort to complex test setup using remote access to multiple computers.

driver_test can be included in the wpa_supplicant build in the same way as any other driver interface, i.e., by adding the following line into .config:

```
CONFIG_DRIVER_TEST=y
```

When running wpa_supplicant, the test interface is selected by using *-Dtest* command line argument. The interface name (*-i* argument) can be selected arbitrarily, i.e., it does not need to match with any existing network interface. The interface name is used to generate a MAC address, so when using multiple clients, each should use a different interface, e.g., *sta1*, *sta2*, and so on.

wpa_supplicant and hostapd are configured in the same way as they would be for normal use. Following example shows a simple test setup for WPA-PSK.

hostapd is configured with following psk-test.conf configuration file:

```
driver=test

interface=ap1
logger_stdout=-1
logger_stdout_level=0
debug=2
dump_file=/tmp/hostapd.dump

test_socket=/tmp/Test/ap1

ssid=jkm-test-psk

wpa=1
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
wpa_passphrase=12345678
```

and started with following command:

```
hostapd psk-test.conf
```

wpa_supplicant uses following configuration file:

```
driver_param=test_socket=/tmp/Test/ap1

network={
    ssid="jkm-test-psk"
    key_mgmt=WPA-PSK
    psk="12345678"
}
```

wpa_supplicant can then be started with following command:

```
wpa_supplicant -Dtest -cpsk-test.conf -ista1 -ddK
```

If run without debug information, i.e., with

```
wpa_supplicant -Dtest -cpsk-test.conf -ista1
```

wpa_supplicant completes authentication and prints following events:

```
Trying to associate with 02:b8:a6:62:08:5a (SSID='jkm-test-psk' freq=0 MHz)
Associated with 02:b8:a6:62:08:5a
WPA: Key negotiation completed with 02:b8:a6:62:08:5a [PTK=TKIP GTK=TKIP]
CTRL-EVENT-CONNECTED - Connection to 02:b8:a6:62:08:5a completed (auth)
```

If test setup is using multiple clients, it is possible to run multiple wpa_supplicant processes. Alternatively, the support for multiple interfaces can be used with just one process to save some resources on single-CPU systems. For example, following command runs two clients:

```
./wpa_supplicant -Dtest -cpsk-test.conf -ista1 \
-N -Dtest -cpsk-test.conf -ista2
```

This shows following event log:

```
Trying to associate with 02:b8:a6:62:08:5a (SSID='jkm-test-psk' freq=0 MHz)
Associated with 02:b8:a6:62:08:5a
WPA: Key negotiation completed with 02:b8:a6:62:08:5a [PTK=TKIP GTK=TKIP]
CTRL-EVENT-CONNECTED - Connection to 02:b8:a6:62:08:5a completed (auth)
Trying to associate with 02:b8:a6:62:08:5a (SSID='jkm-test-psk' freq=0 MHz)
Associated with 02:b8:a6:62:08:5a
WPA: Key negotiation completed with 02:b8:a6:62:08:5a [PTK=TKIP GTK=TKIP]
CTRL-EVENT-CONNECTED - Connection to 02:b8:a6:62:08:5a completed (auth)
```

hostapd shows this with following events:

```
ap1: STA 02:b5:64:63:30:63 IEEE 802.11: associated
ap1: STA 02:b5:64:63:30:63 WPA: pairwise key handshake completed (WPA)
ap1: STA 02:b5:64:63:30:63 WPA: group key handshake completed (WPA)
ap1: STA 02:2a:c4:18:5b:f3 IEEE 802.11: associated
ap1: STA 02:2a:c4:18:5b:f3 WPA: pairwise key handshake completed (WPA)
ap1: STA 02:2a:c4:18:5b:f3 WPA: group key handshake completed (WPA)
```

By default, driver_param is simulating a driver that uses the WPA/RSN IE generated by wpa_supplicant. Driver-generated IE and AssocInfo events can be tested by adding *use_associnfo=1* to the *driver_param* line in the configuration file. For example:

```
driver_param=test_socket=/tmp/Test/ap1 use_associnfo=1
```

### 8.6.4 Unit tests

Number of the components (.c files) used in wpa_supplicant define their own unit tests for automated validation of the basic functionality. Most of the tests for cryptographic algorithms are using standard test vectors to validate functionality. These tests can be useful especially when verifying port to a new CPU target.

In most cases, these tests are implemented in the end of the same file with functions that are normally commented out, but ca be included by defining a pre-processor variable when building the file separately. The details of the needed build options are included in the Makefile (test-∗ targets). All automated unit tests can be run with

```
make tests
```

This make target builds and runs each test and terminates with zero exit code if all tests were completed successfully.

# Index


_FUNC
    config.c, 164
_INT
    config.c, 164

aborted_cached
    eapol_ctx, 31
accept_802_1x_keys
    eapol_config, 28
add_pmkid
    wpa_driver_ops, 53
aes.c, 99
    aes_decrypt, 102
    aes_decrypt_deinit, 102
    aes_decrypt_init, 102
    aes_encrypt, 103
    aes_encrypt_deinit, 103
    aes_encrypt_init, 103
    GETU32, 101
    PUTU32, 101
    rijndaelKeySetupDec, 103
    rijndaelKeySetupEnc, 104
    ROUND, 101, 102
aes.h, 105
aes_128_cbc_decrypt
    aes_wrap.c, 108
    aes_wrap.h, 114
aes_128_cbc_encrypt
    aes_wrap.c, 108
    aes_wrap.h, 114
aes_128_ctr_encrypt
    aes_wrap.c, 108
    aes_wrap.h, 115
aes_128_eax_decrypt
    aes_wrap.c, 109
    aes_wrap.h, 115
aes_128_eax_encrypt
    aes_wrap.c, 109
    aes_wrap.h, 116
aes_128_encrypt_block
    aes_wrap.c, 110
    aes_wrap.h, 117
aes_decrypt
    aes.c, 102
    crypto.h, 201
aes_decrypt_deinit
    aes.c, 102
    crypto.h, 201
aes_decrypt_init
    aes.c, 102
    crypto.h, 201
aes_encrypt
    aes.c, 103
    crypto.h, 202
aes_encrypt_deinit
    aes.c, 103
    crypto.h, 202
aes_encrypt_init
    aes.c, 103
    crypto.h, 202
aes_unwrap
    aes_wrap.c, 110
    aes_wrap.h, 117
aes_wrap
    aes_wrap.c, 111
    aes_wrap.h, 118
aes_wrap.c, 106
    aes_128_cbc_decrypt, 108
    aes_128_cbc_encrypt, 108
    aes_128_ctr_encrypt, 108
    aes_128_eax_decrypt, 109
    aes_128_eax_encrypt, 109
    aes_128_encrypt_block, 110
    aes_unwrap, 110
    aes_wrap, 111
    omac1_aes_128, 111
aes_wrap.h, 113
    aes_128_cbc_decrypt, 114
    aes_128_cbc_encrypt, 114
    aes_128_ctr_encrypt, 115
    aes_128_eax_decrypt, 115
    aes_128_eax_encrypt, 116
    aes_128_encrypt_block, 117
    aes_unwrap, 117
    aes_wrap, 118
    omac1_aes_128, 118
altsubject_match2
    wpa_ssid, 86
anonymous_identity
    wpa_ssid, 86

ap_scan
    wpa_config, 41
asn1.c, 120
asn1.h, 122
asn1_test.c, 124
assoc_info
    wpa_event_data, 65
associate
    wpa_driver_ops, 53
auth_alg
    wpa_driver_associate_params, 49
    wpa_ssid, 86
AVP_PAD
    eap_ttls.h, 464

base64.c, 126
    base64_decode, 127
    base64_encode, 127
base64.h, 128
    base64_decode, 128
    base64_encode, 129
base64_decode
    base64.c, 127
    base64.h, 128
base64_encode
    base64.c, 127
    base64.h, 129
beacon_ies
    wpa_event_data::assoc_info, 66
bignum.c, 130
    bignum_add, 132
    bignum_cmp, 132
    bignum_cmp_d, 132
    bignum_deinit, 132
    bignum_exptmod, 133
    bignum_get_unsigned_bin, 133
    bignum_get_unsigned_bin_len, 133
    bignum_init, 134
    bignum_mul, 134
    bignum_mulmod, 134
    bignum_set_unsigned_bin, 135
    bignum_sub, 135
bignum.h, 137
    bignum_add, 138
    bignum_cmp, 138
    bignum_cmp_d, 138
    bignum_deinit, 139
    bignum_exptmod, 139
    bignum_get_unsigned_bin, 139
    bignum_get_unsigned_bin_len, 140
    bignum_init, 140
    bignum_mul, 140
    bignum_mulmod, 141
    bignum_set_unsigned_bin, 141
    bignum_sub, 141
bignum_add
    bignum.c, 132
    bignum.h, 138
bignum_cmp
    bignum.c, 132
    bignum.h, 138
bignum_cmp_d
    bignum.c, 132
    bignum.h, 138
bignum_deinit
    bignum.c, 132
    bignum.h, 139
bignum_exptmod
    bignum.c, 133
    bignum.h, 139
bignum_get_unsigned_bin
    bignum.c, 133
    bignum.h, 139
bignum_get_unsigned_bin_len
    bignum.c, 133
    bignum.h, 140
bignum_init
    bignum.c, 134
    bignum.h, 140
bignum_mul
    bignum.c, 134
    bignum.h, 140
bignum_mulmod
    bignum.c, 134
    bignum.h, 141
bignum_set_unsigned_bin
    bignum.c, 135
    bignum.h, 141
bignum_sub
    bignum.c, 135
    bignum.h, 141
bridge_ifname
    wpa_interface, 73
bssid
    wpa_driver_associate_params, 49
    wpa_event_data::pmkid_candidate, 70
    wpa_ssid, 86
build_config.h, 143

ca_cert
    wpa_ssid, 86
ca_cert2
    wpa_ssid, 86
ca_path
    wpa_ssid, 87
ca_path2
    wpa_ssid, 87
cb

eapol_ctx, 31
challenge_response
ms_funcs.c, 592
ms_funcs.h, 599
client_cert
wpa_ssid, 87
client_cert2
wpa_ssid, 87
common.c, 144
hexstr2bin, 146
hwaddr_aton, 146
inc_byte_array, 146
wpa_debug_print_timestamp, 146
wpa_hexdump, 147
wpa_hexdump_ascii, 147
wpa_hexdump_ascii_key, 147
wpa_hexdump_key, 148
wpa_msg_register_cb, 148
wpa_printf, 148
wpa_snprintf_hex, 149
wpa_snprintf_hex_uppercase, 149
wpa_ssid_txt, 149
common.h, 151
hexstr2bin, 156
hwaddr_aton, 157
inc_byte_array, 157
wpa_debug_print_timestamp, 157
WPA_GET_BE24, 155
WPA_GET_BE32, 155
wpa_hexdump, 157
wpa_hexdump_ascii, 158
wpa_hexdump_ascii_key, 158
wpa_hexdump_key, 158
wpa_msg, 159
wpa_msg_register_cb, 159
wpa_printf, 159
WPA_PUT_BE16, 155
WPA_PUT_BE24, 155
WPA_PUT_BE32, 156
WPA_PUT_BE64, 156
WPA_PUT_LE16, 156
wpa_snprintf_hex, 160
wpa_snprintf_hex_uppercase, 160
wpa_ssid_txt, 160
config.c, 162
_FUNC, 164
_INT, 164
wpa_config_add_network, 165
wpa_config_add_prio_network, 165
wpa_config_alloc_empty, 165
wpa_config_allowed_eap_method, 166
wpa_config_debug_dump_networks, 166
wpa_config_free, 166
wpa_config_free_blob, 167
wpa_config_free_ssid, 167
wpa_config_get, 167
wpa_config_get_blob, 167
wpa_config_get_network, 168
wpa_config_get_no_key, 168
wpa_config_remove_blob, 168
wpa_config_remove_network, 169
wpa_config_set, 169
wpa_config_set_blob, 170
wpa_config_set_network_defaults, 170
wpa_config_update_psk, 170
config.h, 171
wpa_config_add_network, 173
wpa_config_add_prio_network, 173
wpa_config_alloc_empty, 174
wpa_config_debug_dump_networks, 174
wpa_config_free, 174
wpa_config_free_blob, 175
wpa_config_free_ssid, 175
wpa_config_get, 175
wpa_config_get_blob, 175
wpa_config_get_network, 176
wpa_config_get_no_key, 176
wpa_config_read, 176
wpa_config_remove_blob, 177
wpa_config_remove_network, 177
wpa_config_set, 178
wpa_config_set_blob, 178
wpa_config_set_network_defaults, 179
wpa_config_update_psk, 179
wpa_config_write, 179
config_file.c, 181
wpa_config_read, 182
wpa_config_write, 183
config_none.c, 184
wpa_config_read, 185
wpa_config_write, 185
config_ssid.h, 187
DEFAULT_EAPOL_FLAGS, 188
DEFAULT_GROUP, 188
wpa_config_allowed_eap_method, 189
config_types.h, 190
config_winreg.c, 191
wpa_config_read, 192
wpa_config_write, 193
confname
wpa_interface, 73
crypto.c, 194
des_encrypt, 195
md4_vector, 195
crypto.h, 197
aes_decrypt, 201
aes_decrypt_deinit, 201
aes_decrypt_init, 201

aes_encrypt, 202
aes_encrypt_deinit, 202
aes_encrypt_init, 202
crypto_cipher_decrypt, 203
crypto_cipher_deinit, 203
crypto_cipher_encrypt, 203
crypto_cipher_init, 203
crypto_global_deinit, 204
crypto_global_init, 204
crypto_hash_finish, 204
crypto_hash_init, 204
crypto_hash_update, 205
crypto_mod_exp, 205
crypto_private_key_free, 205
crypto_private_key_import, 206
crypto_private_key_sign_pkcs1, 206
crypto_public_key_decrypt_pkcs1, 206
crypto_public_key_encrypt_pkcs1_v15, 207
crypto_public_key_free, 207
crypto_public_key_from_cert, 207
crypto_public_key_import, 208
des_encrypt, 208
fips186_2_prf, 208
md4_vector, 208
md5_vector, 209
sha1_vector, 209
sha256_vector, 209
crypto_cipher_decrypt
crypto.h, 203
crypto_cipher_deinit
crypto.h, 203
crypto_cipher_encrypt
crypto.h, 203
crypto_cipher_init
crypto.h, 203
crypto_cryptoapi.c, 210
des_encrypt, 211
md4_vector, 211
crypto_global_deinit
crypto.h, 204
crypto_global_init
crypto.h, 204
crypto_gnutls.c, 213
des_encrypt, 214
md4_vector, 214
crypto_hash_finish
crypto.h, 204
crypto_hash_init
crypto.h, 204
crypto_hash_update
crypto.h, 205
crypto_internal.c, 215
crypto_libtomcrypt.c, 216
des_encrypt, 217
md4_vector, 217
crypto_mod_exp
crypto.h, 205
crypto_none.c, 219
des_encrypt, 220
md4_vector, 220
crypto_private_key_free
crypto.h, 205
crypto_private_key_import
crypto.h, 206
crypto_private_key_sign_pkcs1
crypto.h, 206
crypto_public_key_decrypt_pkcs1
crypto.h, 206
crypto_public_key_encrypt_pkcs1_v15
crypto.h, 207
crypto_public_key_free
crypto.h, 207
crypto_public_key_from_cert
crypto.h, 207
crypto_public_key_import
crypto.h, 208
crypto_rsa_exptmod
rsa.c, 691
rsa.h, 694
crypto_rsa_free
rsa.c, 691
rsa.h, 695
crypto_rsa_get_modulus_len
rsa.c, 692
rsa.h, 695
crypto_rsa_import_private_key
rsa.c, 692
rsa.h, 696
crypto_rsa_import_public_key
rsa.c, 693
rsa.h, 696
ctrl_iface.c, 221
wpa_supplicant_ctrl_iface_process, 222
wpa_supplicant_global_ctrl_iface_process, 223
ctrl_iface.h, 225
wpa_supplicant_ctrl_iface_deinit, 226
wpa_supplicant_ctrl_iface_init, 226
wpa_supplicant_ctrl_iface_process, 227
wpa_supplicant_ctrl_iface_wait, 228
wpa_supplicant_global_ctrl_iface_deinit, 228
wpa_supplicant_global_ctrl_iface_init, 229
wpa_supplicant_global_ctrl_iface_process, 229
ctrl_iface_dbus.c, 231
wpa_supplicant_dbus_ctrl_iface_deinit, 233
wpa_supplicant_dbus_ctrl_iface_init, 233
wpa_supplicant_dbus_next_objid, 234

wpa_supplicant_dbus_notify_scan_results, 234
wpa_supplicant_dbus_notify_state_change, 235
wpa_supplicant_get_dbus_path, 235
wpa_supplicant_get_iface_by_dbus_path, 235
wpa_supplicant_set_dbus_path, 236
wpas_dbus_decompose_object_path, 236
wpas_dbus_new_invalid_iface_error, 236
wpas_dbus_new_invalid_network_error, 236
wpas_dbus_register_iface, 237
wpas_dbus_unregister_iface, 237
ctrl_iface_dbus.h, 239
ctrl_iface_dbus_handlers.c, 240
wpas_dbus_bssid_properties, 242
wpas_dbus_global_add_interface, 242
wpas_dbus_global_get_interface, 243
wpas_dbus_global_remove_interface, 244
wpas_dbus_iface_add_network, 244
wpas_dbus_iface_capabilities, 245
wpas_dbus_iface_disable_network, 245
wpas_dbus_iface_disconnect, 246
wpas_dbus_iface_enable_network, 246
wpas_dbus_iface_get_state, 247
wpas_dbus_iface_remove_network, 247
wpas_dbus_iface_scan, 248
wpas_dbus_iface_scan_results, 248
wpas_dbus_iface_select_network, 249
wpas_dbus_iface_set_ap_scan, 250
wpas_dbus_iface_set_network, 250
ctrl_iface_dbus_handlers.h, 252
ctrl_iface_named_pipe.c, 253
wpa_supplicant_ctrl_iface_deinit, 254
wpa_supplicant_ctrl_iface_init, 254
wpa_supplicant_ctrl_iface_wait, 255
wpa_supplicant_global_ctrl_iface_deinit, 255
wpa_supplicant_global_ctrl_iface_init, 255
ctrl_iface_udp.c, 257
wpa_supplicant_ctrl_iface_deinit, 258
wpa_supplicant_ctrl_iface_init, 258
wpa_supplicant_ctrl_iface_wait, 259
wpa_supplicant_global_ctrl_iface_deinit, 259
wpa_supplicant_global_ctrl_iface_init, 260
ctrl_iface_unix.c, 261
wpa_supplicant_ctrl_iface_deinit, 262
wpa_supplicant_ctrl_iface_init, 262
wpa_supplicant_ctrl_iface_wait, 263
wpa_supplicant_global_ctrl_iface_deinit, 263
wpa_supplicant_global_ctrl_iface_init, 264
ctrl_interface
wpa_config, 41
wpa_interface, 73
ctrl_interface_group
wpa_config, 42
dbus_dict_helpers.c, 265
wpa_dbus_dict_append_bool, 266
wpa_dbus_dict_append_byte, 266
wpa_dbus_dict_append_byte_array, 267
wpa_dbus_dict_append_double, 267
wpa_dbus_dict_append_int16, 267
wpa_dbus_dict_append_int32, 268
wpa_dbus_dict_append_int64, 268
wpa_dbus_dict_append_object_path, 268
wpa_dbus_dict_append_string, 269
wpa_dbus_dict_append_string_array, 269
wpa_dbus_dict_append_uint16, 270
wpa_dbus_dict_append_uint32, 270
wpa_dbus_dict_append_uint64, 270
wpa_dbus_dict_begin_string_array, 270
wpa_dbus_dict_close_write, 271
wpa_dbus_dict_end_string_array, 271
wpa_dbus_dict_entry_clear, 271
wpa_dbus_dict_get_entry, 272
wpa_dbus_dict_has_dict_entry, 272
wpa_dbus_dict_open_read, 272
wpa_dbus_dict_open_write, 273
wpa_dbus_dict_string_array_add_element, 273
dbus_dict_helpers.h, 274
wpa_dbus_dict_append_bool, 275
wpa_dbus_dict_append_byte, 275
wpa_dbus_dict_append_byte_array, 275
wpa_dbus_dict_append_double, 276
wpa_dbus_dict_append_int16, 276
wpa_dbus_dict_append_int32, 276
wpa_dbus_dict_append_int64, 277
wpa_dbus_dict_append_object_path, 277
wpa_dbus_dict_append_string, 277
wpa_dbus_dict_append_string_array, 278
wpa_dbus_dict_append_uint16, 278
wpa_dbus_dict_append_uint32, 279
wpa_dbus_dict_append_uint64, 279
wpa_dbus_dict_begin_string_array, 279
wpa_dbus_dict_close_write, 280
wpa_dbus_dict_end_string_array, 280
wpa_dbus_dict_entry_clear, 280
wpa_dbus_dict_get_entry, 280
wpa_dbus_dict_has_dict_entry, 281
wpa_dbus_dict_open_read, 281
wpa_dbus_dict_open_write, 281
wpa_dbus_dict_string_array_add_element, 282
deauthenticate
wpa_driver_ops, 53
DEFAULT_EAPOL_FLAGS
config_ssid.h, 188
DEFAULT_GROUP
config_ssid.h, 188

defs.h, 283
WPA_4WAY_HANDSHAKE, 284
WPA_ASSOCIATED, 284
WPA_ASSOCIATING, 284
WPA_COMPLETED, 284
WPA_DISCONNECTED, 284
WPA_GROUP_HANDSHAKE, 284
WPA_INACTIVE, 284
WPA_SCANNING, 284
wpa_states, 284
deinit
eap_method, 16
wpa_driver_ops, 54
deinit_for_reauth
eap_method, 16
des.c, 286
des_encrypt
crypto.c, 195
crypto.h, 208
crypto_cryptoapi.c, 211
crypto_gnutls.c, 214
crypto_libtomcrypt.c, 217
crypto_none.c, 220
desc
wpa_driver_ops, 54
dh_file
wpa_ssid, 88
dh_file2
wpa_ssid, 88
disabled
wpa_ssid, 88
disassociate
wpa_driver_ops, 54
dot11RSNAConfigPMKLifetime
wpa_config, 42
dot11RSNAConfigPMKReauthThreshold
wpa_config, 42
dot11RSNAConfigSATimeout
wpa_config, 43
driver.h, 288
driver_atmel.c, 290
wpa_driver_atmel_ops, 291
driver_broadcom.c, 292
wpa_driver_broadcom_ops, 293
driver_bsd.c, 295
LE_READ_4, 297
wpa_driver_bsd_ops, 297
driver_hostap.c, 298
wpa_driver_hostap_ops, 299
driver_hostap.h, 300
driver_ipw.c, 302
wpa_driver_ipw_ops, 303
driver_madwifi.c, 304
wpa_driver_madwifi_ops, 305
driver_ndis.c, 306
driver_ndis.h, 310
driver_ndis_.c, 311
driver_ndiswrapper.c, 313
wpa_driver_ndiswrapper_ops, 314
driver_param
wpa_config, 43
wpa_interface, 74
driver_prism54.c, 315
wpa_driver_prism54_ops, 316
driver_test.c, 317
driver_wext.c, 319
wpa_driver_wext_deinit, 321
wpa_driver_wext_get_bssid, 321
wpa_driver_wext_get_ifflags, 322
wpa_driver_wext_get_scan_results, 322
wpa_driver_wext_get_ssid, 322
wpa_driver_wext_init, 323
wpa_driver_wext_scan, 323
wpa_driver_wext_scan_timeout, 324
wpa_driver_wext_set_bssid, 324
wpa_driver_wext_set_freq, 324
wpa_driver_wext_set_ifflags, 324
wpa_driver_wext_set_key, 325
wpa_driver_wext_set_mode, 325
wpa_driver_wext_set_ssid, 326
driver_wext.h, 327
wpa_driver_wext_deinit, 328
wpa_driver_wext_get_bssid, 329
wpa_driver_wext_get_ifflags, 329
wpa_driver_wext_get_scan_results, 329
wpa_driver_wext_get_ssid, 330
wpa_driver_wext_init, 330
wpa_driver_wext_scan, 330
wpa_driver_wext_scan_timeout, 331
wpa_driver_wext_set_bssid, 331
wpa_driver_wext_set_freq, 332
wpa_driver_wext_set_ifflags, 332
wpa_driver_wext_set_key, 332
wpa_driver_wext_set_mode, 333
wpa_driver_wext_set_ssid, 333
driver_wired.c, 334
wpa_driver_wired_ops, 335
drivers.c, 336

eap.c, 338
eap_clear_config_otp, 342
eap_get_config, 342
eap_get_config_blob, 342
eap_get_config_identity, 342
eap_get_config_new_password, 343
eap_get_config_otp, 343
eap_get_config_password, 344
eap_get_eapKeyData, 344

eap_get_eapRespData, 344
eap_get_phase2_type, 345
eap_get_phase2_types, 345
eap_hdr_validate, 345
eap_invalidate_cached_session, 346
eap_key_available, 346
eap_msg_alloc, 346
eap_notify_lower_layer_success, 347
eap_notify_pending, 347
eap_notify_success, 347
eap_register_scard_ctx, 348
eap_set_config_blob, 348
eap_set_fast_reauth, 348
eap_set_force_disabled, 348
eap_set_workaround, 349
eap_sm_abort, 349
eap_sm_buildIdentity, 349
eap_sm_deinit, 350
eap_sm_get_status, 350
eap_sm_init, 351
eap_sm_notify_ctrl_attached, 351
eap_sm_request_identity, 352
eap_sm_request_new_password, 352
eap_sm_request_otp, 352
eap_sm_request_passphrase, 352
eap_sm_request_password, 353
eap_sm_request_pin, 353
eap_sm_step, 353
eap.h, 354
eap_get_eapKeyData, 357
eap_get_eapRespData, 358
eap_get_phase2_type, 358
eap_get_phase2_types, 358
eap_invalidate_cached_session, 359
eap_key_available, 359
eap_notify_lower_layer_success, 359
eap_notify_success, 359
eap_register_scard_ctx, 360
eap_set_fast_reauth, 360
eap_set_force_disabled, 360
eap_set_workaround, 360
eap_sm_abort, 361
eap_sm_buildIdentity, 361
eap_sm_deinit, 361
eap_sm_get_status, 362
eap_sm_init, 362
eap_sm_notify_ctrl_attached, 363
eap_sm_request_identity, 363
eap_sm_request_new_password, 364
eap_sm_request_otp, 364
eap_sm_request_passphrase, 364
eap_sm_request_password, 364
eap_sm_request_pin, 365
eap_sm_step, 365
EAPOL_altAccept, 357
EAPOL_altReject, 357
eapol_bool_var, 356
EAPOL_eapFail, 357
EAPOL_eapNoResp, 357
EAPOL_eapReq, 357
EAPOL_eapResp, 357
EAPOL_eapRestart, 356
EAPOL_eapSuccess, 356
EAPOL_idleWhile, 357
eapol_int_var, 357
EAPOL_portEnabled, 357
eap_aka.c, 366
eap_clear_config_otp
eap.c, 342
eap_i.h, 386
eap_config, 13
opensc_engine_path, 13
pkcs11_engine_path, 13
pkcs11_module_path, 14
eap_defs.h, 367
eap_fast.c, 369
eap_get_config
eap.c, 342
eap_i.h, 386
eap_get_config_blob
eap.c, 342
eap_i.h, 386
eap_get_config_identity
eap.c, 342
eap_i.h, 387
eap_get_config_new_password
eap.c, 343
eap_i.h, 387
eap_get_config_otp
eap.c, 343
eap_i.h, 387
eap_get_config_password
eap.c, 344
eap_i.h, 388
eap_get_eapKeyData
eap.c, 344
eap.h, 357
eap_get_eapRespData
eap.c, 344
eap.h, 358
eap_get_name
eap_methods.c, 396
eap_methods.h, 402
eap_get_names
eap_methods.c, 397
eap_methods.h, 402
eap_get_names_as_string_array
eap_methods.c, 397

eap_methods.h, 403
eap_get_phase2_type
eap.c, 345
eap.h, 358
eap_get_phase2_types
eap.c, 345
eap.h, 358
eap_get_type
eap_methods.c, 397
eap_methods.h, 403
eap_gpsk.c, 371
eap_gpsk_common.c, 373
eap_gpsk_compute_mic, 374
eap_gpsk_derive_keys, 374
eap_gpsk_mic_len, 375
eap_gpsk_supported_ciphersuite, 375
eap_gpsk_common.h, 377
eap_gpsk_compute_mic, 378
eap_gpsk_derive_keys, 378
eap_gpsk_mic_len, 379
eap_gpsk_supported_ciphersuite, 379
eap_gpsk_compute_mic
eap_gpsk_common.c, 374
eap_gpsk_common.h, 378
eap_gpsk_derive_keys
eap_gpsk_common.c, 374
eap_gpsk_common.h, 378
eap_gpsk_mic_len
eap_gpsk_common.c, 375
eap_gpsk_common.h, 379
eap_gpsk_supported_ciphersuite
eap_gpsk_common.c, 375
eap_gpsk_common.h, 379
eap_gtc.c, 381
eap_hdr_validate
eap.c, 345
eap_i.h, 388
eap_i.h, 383
eap_clear_config_otp, 386
eap_get_config, 386
eap_get_config_blob, 386
eap_get_config_identity, 387
eap_get_config_new_password, 387
eap_get_config_otp, 387
eap_get_config_password, 388
eap_hdr_validate, 388
eap_msg_alloc, 389
eap_notify_pending, 389
eap_set_config_blob, 389
eap_invalidate_cached_session
eap.c, 346
eap.h, 359
eap_key_available
eap.c, 346
eap.h, 359
eap_leap.c, 391
eap_md5.c, 393
eap_method, 15
deinit, 16
deinit_for_reauth, 16
free, 16
get_emsk, 17
get_identity, 17
get_status, 17
getKey, 18
has_reauth_data, 18
init, 18
init_for_reauth, 19
isKeyAvailable, 19
next, 19
process, 19
version, 20
eap_method_ret, 21
eap_methods
wpa_ssid, 88
eap_methods.c, 395
eap_get_name, 396
eap_get_names, 397
eap_get_names_as_string_array, 397
eap_get_type, 397
eap_peer_get_methods, 398
eap_peer_method_alloc, 398
eap_peer_method_free, 398
eap_peer_method_register, 399
eap_peer_register_methods, 399
eap_peer_unregister_methods, 399
eap_sm_get_eap_methods, 400
eap_methods.h, 401
eap_get_name, 402
eap_get_names, 402
eap_get_names_as_string_array, 403
eap_get_type, 403
eap_peer_get_methods, 403
eap_peer_method_alloc, 404
eap_peer_method_free, 404
eap_peer_method_register, 404
eap_peer_register_methods, 405
eap_peer_unregister_methods, 405
eap_sm_get_eap_methods, 405
eap_mschapv2.c, 407
eap_peer_mschapv2_register, 408
eap_msg_alloc
eap.c, 346
eap_i.h, 389
eap_notify_lower_layer_success
eap.c, 347
eap.h, 359
eap_notify_pending

eap.c, 347
eap_i.h, 389
eap_notify_success
eap.c, 347
eap.h, 359
eap_otp.c, 409
eap_pax.c, 411
eap_pax_common.c, 412
eap_pax_initial_key_derivation, 413
eap_pax_kdf, 413
eap_pax_mac, 414
eap_pax_common.h, 416
eap_pax_initial_key_derivation, 417
eap_pax_kdf, 418
eap_pax_mac, 418
eap_pax_initial_key_derivation
eap_pax_common.c, 413
eap_pax_common.h, 417
eap_pax_kdf
eap_pax_common.c, 413
eap_pax_common.h, 418
eap_pax_mac
eap_pax_common.c, 414
eap_pax_common.h, 418
eap_peap.c, 420
eap_peer_get_methods
eap_methods.c, 398
eap_methods.h, 403
eap_peer_method_alloc
eap_methods.c, 398
eap_methods.h, 404
eap_peer_method_free
eap_methods.c, 398
eap_methods.h, 404
eap_peer_method_register
eap_methods.c, 399
eap_methods.h, 404
eap_peer_mschapv2_register
eap_mschapv2.c, 408
eap_peer_register_methods
eap_methods.c, 399
eap_methods.h, 405
eap_peer_unregister_methods
eap_methods.c, 399
eap_methods.h, 405
eap_psk.c, 421
eap_psk_common.c, 422
eap_psk_common.h, 424
eap_register_scard_ctx
eap.c, 348
eap.h, 360
eap_sake.c, 425
eap_sake_common.c, 427
eap_sake_compute_mic, 428
eap_sake_derive_keys, 428
eap_sake_parse_attributes, 429
eap_sake_common.h, 430
eap_sake_compute_mic, 431
eap_sake_derive_keys, 431
eap_sake_parse_attributes, 432
eap_sake_compute_mic
eap_sake_common.c, 428
eap_sake_common.h, 431
eap_sake_derive_keys
eap_sake_common.c, 428
eap_sake_common.h, 431
eap_sake_parse_attributes
eap_sake_common.c, 429
eap_sake_common.h, 432
eap_set_config_blob
eap.c, 348
eap_i.h, 389
eap_set_fast_reauth
eap.c, 348
eap.h, 360
eap_set_force_disabled
eap.c, 348
eap.h, 360
eap_set_workaround
eap.c, 349
eap.h, 360
eap_sim.c, 433
eap_sim_common.c, 435
eap_sim_common.h, 437
eap_sm, 22
eap_sm_abort
eap.c, 349
eap.h, 361
eap_sm_buildIdentity
eap.c, 349
eap.h, 361
eap_sm_deinit
eap.c, 350
eap.h, 361
eap_sm_get_eap_methods
eap_methods.c, 400
eap_methods.h, 405
eap_sm_get_status
eap.c, 350
eap.h, 362
eap_sm_init
eap.c, 351
eap.h, 362
eap_sm_notify_ctrl_attached
eap.c, 351
eap.h, 363
eap_sm_request_identity
eap.c, 352

eap.h, 363
eap_sm_request_new_password
eap.c, 352
eap.h, 364
eap_sm_request_otp
eap.c, 352
eap.h, 364
eap_sm_request_passphrase
eap.c, 352
eap.h, 364
eap_sm_request_password
eap.c, 353
eap.h, 364
eap_sm_request_pin
eap.c, 353
eap.h, 365
eap_sm_step
eap.c, 353
eap.h, 365
eap_tls.c, 440
eap_tls_build_ack
eap_tls_common.c, 442
eap_tls_common.h, 449
eap_tls_common.c, 441
eap_tls_build_ack, 442
eap_tls_data_reassemble, 443
eap_tls_derive_key, 443
eap_tls_process_helper, 444
eap_tls_process_init, 445
eap_tls_reauth_init, 445
eap_tls_ssl_deinit, 446
eap_tls_ssl_init, 446
eap_tls_status, 446
eap_tls_common.h, 448
eap_tls_build_ack, 449
eap_tls_data_reassemble, 449
eap_tls_derive_key, 450
eap_tls_process_helper, 451
eap_tls_process_init, 451
eap_tls_reauth_init, 452
eap_tls_ssl_deinit, 452
eap_tls_ssl_init, 453
eap_tls_status, 453
eap_tls_data_reassemble
eap_tls_common.c, 443
eap_tls_common.h, 449
eap_tls_derive_key
eap_tls_common.c, 443
eap_tls_common.h, 450
eap_tls_process_helper
eap_tls_common.c, 444
eap_tls_common.h, 451
eap_tls_process_init
eap_tls_common.c, 445
eap_tls_common.h, 451
eap_tls_reauth_init
eap_tls_common.c, 445
eap_tls_common.h, 452
eap_tls_ssl_deinit
eap_tls_common.c, 446
eap_tls_common.h, 452
eap_tls_ssl_init
eap_tls_common.c, 446
eap_tls_common.h, 453
eap_tls_status
eap_tls_common.c, 446
eap_tls_common.h, 453
eap_tlv.c, 455
eap_tlv_build_nak, 456
eap_tlv_build_result, 456
eap_tlv_process, 457
eap_tlv.h, 458
eap_tlv_build_nak, 459
eap_tlv_build_result, 459
eap_tlv_process, 460
eap_tlv_build_nak
eap_tlv.c, 456
eap_tlv.h, 459
eap_tlv_build_result
eap_tlv.c, 456
eap_tlv.h, 459
eap_tlv_process
eap_tlv.c, 457
eap_tlv.h, 460
eap_ttls.c, 461
eap_ttls.h, 463
AVP_PAD, 464
eap_vendor_test.c, 465
eap_workaround
wpa_ssid, 88
EAPOL_altAccept
eap.h, 357
EAPOL_altReject
eap.h, 357
eapol_bool_var
eap.h, 356
eapol_callbacks, 24
get_bool, 25
get_config, 25
get_config_blob, 25
get_eapReqData, 25
get_int, 25
notify_pending, 26
set_bool, 26
set_config_blob, 26
set_int, 26
eapol_config, 28
accept_802_1x_keys, 28

required_keys, 28
eapol_ctx, 30
aborted_cached, 31
cb, 31
eapol_done_cb, 31
eapol_send, 32
get_config_blob, 32
opensc_engine_path, 32
pkcs11_engine_path, 32
pkcs11_module_path, 32
preauth, 32
scard_ctx, 33
set_config_blob, 33
set_wep_key, 33
eapol_done_cb
eapol_ctx, 31
EAPOL_eapFail
eap.h, 357
EAPOL_eapNoResp
eap.h, 357
EAPOL_eapReq
eap.h, 357
EAPOL_eapResp
eap.h, 357
EAPOL_eapRestart
eap.h, 356
EAPOL_eapSuccess
eap.h, 356
EAPOL_idleWhile
eap.h, 357
eapol_int_var
eap.h, 357
EAPOL_portEnabled
eap.h, 357
eapol_send
eapol_ctx, 32
eapol_sm, 34
eapol_sm.c, 467
eapol_sm_configure, 470
eapol_sm_deinit, 470
eapol_sm_get_key, 471
eapol_sm_get_mib, 471
eapol_sm_get_status, 471
eapol_sm_init, 472
eapol_sm_invalidate_cached_session, 472
eapol_sm_notify_cached, 473
eapol_sm_notify_config, 473
eapol_sm_notify_ctrl_attached, 474
eapol_sm_notify_ctrl_response, 474
eapol_sm_notify_eap_fail, 475
eapol_sm_notify_eap_success, 475
eapol_sm_notify_logoff, 476
eapol_sm_notify_lower_layer_success, 476
eapol_sm_notify_pmkid_attempt, 476
eapol_sm_notify_portControl, 477
eapol_sm_notify_portEnabled, 477
eapol_sm_notify_portValid, 478
eapol_sm_notify_tx_eapol_key, 478
eapol_sm_register_scard_ctx, 478
eapol_sm_request_reauth, 479
eapol_sm_rx_eapol, 479
eapol_sm_step, 479
eapol_sm.h, 481
eapol_sm_configure, 483
eapol_sm_deinit, 484
eapol_sm_get_key, 484
eapol_sm_get_mib, 484
eapol_sm_get_status, 485
eapol_sm_init, 485
eapol_sm_invalidate_cached_session, 486
eapol_sm_notify_cached, 486
eapol_sm_notify_config, 486
eapol_sm_notify_ctrl_attached, 487
eapol_sm_notify_ctrl_response, 487
eapol_sm_notify_eap_fail, 488
eapol_sm_notify_eap_success, 488
eapol_sm_notify_logoff, 489
eapol_sm_notify_lower_layer_success, 489
eapol_sm_notify_pmkid_attempt, 490
eapol_sm_notify_portControl, 490
eapol_sm_notify_portEnabled, 490
eapol_sm_notify_portValid, 491
eapol_sm_notify_tx_eapol_key, 491
eapol_sm_register_scard_ctx, 491
eapol_sm_request_reauth, 492
eapol_sm_rx_eapol, 492
eapol_sm_step, 493
eapol_sm_configure
eapol_sm.c, 470
eapol_sm.h, 483
eapol_sm_deinit
eapol_sm.c, 470
eapol_sm.h, 484
eapol_sm_get_key
eapol_sm.c, 471
eapol_sm.h, 484
eapol_sm_get_mib
eapol_sm.c, 471
eapol_sm.h, 484
eapol_sm_get_status
eapol_sm.c, 471
eapol_sm.h, 485
eapol_sm_init
eapol_sm.c, 472
eapol_sm.h, 485
eapol_sm_invalidate_cached_session
eapol_sm.c, 472
eapol_sm.h, 486

eapol_sm_notify_cached
eapol_sm.c, 473
eapol_sm.h, 486
eapol_sm_notify_config
eapol_sm.c, 473
eapol_sm.h, 486
eapol_sm_notify_ctrl_attached
eapol_sm.c, 474
eapol_sm.h, 487
eapol_sm_notify_ctrl_response
eapol_sm.c, 474
eapol_sm.h, 487
eapol_sm_notify_eap_fail
eapol_sm.c, 475
eapol_sm.h, 488
eapol_sm_notify_eap_success
eapol_sm.c, 475
eapol_sm.h, 488
eapol_sm_notify_logoff
eapol_sm.c, 476
eapol_sm.h, 489
eapol_sm_notify_lower_layer_success
eapol_sm.c, 476
eapol_sm.h, 489
eapol_sm_notify_pmkid_attempt
eapol_sm.c, 476
eapol_sm.h, 490
eapol_sm_notify_portControl
eapol_sm.c, 477
eapol_sm.h, 490
eapol_sm_notify_portEnabled
eapol_sm.c, 477
eapol_sm.h, 490
eapol_sm_notify_portValid
eapol_sm.c, 478
eapol_sm.h, 491
eapol_sm_notify_tx_eapol_key
eapol_sm.c, 478
eapol_sm.h, 491
eapol_sm_register_scard_ctx
eapol_sm.c, 478
eapol_sm.h, 491
eapol_sm_request_reauth
eapol_sm.c, 479
eapol_sm.h, 492
eapol_sm_rx_eapol
eapol_sm.c, 479
eapol_sm.h, 492
eapol_sm_step
eapol_sm.c, 479
eapol_sm.h, 493
eapol_test.c, 494
eapol_version
wpa_config, 43
eappsk_len
wpa_ssid, 89
eloop.c, 497
eloop_cancel_timeout, 499
eloop_destroy, 499
eloop_get_user_data, 499
eloop_init, 499
eloop_register_read_sock, 500
eloop_register_signal, 500
eloop_register_signal_reconfig, 500
eloop_register_signal_terminate, 501
eloop_register_sock, 501
eloop_register_timeout, 502
eloop_run, 502
eloop_terminate, 502
eloop_terminated, 502
eloop_unregister_read_sock, 503
eloop_unregister_sock, 503
eloop_wait_for_read_sock, 503
eloop.h, 504
eloop_cancel_timeout, 509
eloop_destroy, 509
eloop_event_handler, 508
eloop_event_type, 509
eloop_get_user_data, 509
eloop_init, 510
eloop_register_event, 510
eloop_register_read_sock, 510
eloop_register_signal, 511
eloop_register_signal_reconfig, 511
eloop_register_signal_terminate, 512
eloop_register_sock, 512
eloop_register_timeout, 513
eloop_run, 513
eloop_signal_handler, 508
eloop_sock_handler, 508
eloop_terminate, 514
eloop_terminated, 514
eloop_timeout_handler, 508
eloop_unregister_event, 514
eloop_unregister_read_sock, 514
eloop_unregister_sock, 515
eloop_wait_for_read_sock, 515
eloop_cancel_timeout
eloop.c, 499
eloop.h, 509
eloop_win.c, 522
eloop_destroy
eloop.c, 499
eloop.h, 509
eloop_none.c, 518
eloop_win.c, 522
eloop_event_handler
eloop.h, 508

eloop_event_type
eloop.h, 509
eloop_get_user_data
eloop.c, 499
eloop.h, 509
eloop_none.c, 518
eloop_win.c, 522
eloop_init
eloop.c, 499
eloop.h, 510
eloop_none.c, 518
eloop_win.c, 523
eloop_none.c, 516
eloop_destroy, 518
eloop_get_user_data, 518
eloop_init, 518
eloop_run, 518
eloop_terminate, 518
eloop_terminated, 519
eloop_unregister_read_sock, 519
eloop_wait_for_read_sock, 519
eloop_register_event
eloop.h, 510
eloop_win.c, 523
eloop_register_read_sock
eloop.c, 500
eloop.h, 510
eloop_win.c, 523
eloop_register_signal
eloop.c, 500
eloop.h, 511
eloop_win.c, 524
eloop_register_signal_reconfig
eloop.c, 500
eloop.h, 511
eloop_win.c, 524
eloop_register_signal_terminate
eloop.c, 501
eloop.h, 512
eloop_win.c, 525
eloop_register_sock
eloop.c, 501
eloop.h, 512
eloop_register_timeout
eloop.c, 502
eloop.h, 513
eloop_win.c, 525
eloop_run
eloop.c, 502
eloop.h, 513
eloop_none.c, 518
eloop_win.c, 526
eloop_signal_handler
eloop.h, 508
eloop_sock_handler
eloop.h, 508
eloop_terminate
eloop.c, 502
eloop.h, 514
eloop_none.c, 518
eloop_win.c, 526
eloop_terminated
eloop.c, 502
eloop.h, 514
eloop_none.c, 519
eloop_win.c, 526
eloop_timeout_handler
eloop.h, 508
eloop_unregister_event
eloop.h, 514
eloop_win.c, 526
eloop_unregister_read_sock
eloop.c, 503
eloop.h, 514
eloop_none.c, 519
eloop_win.c, 527
eloop_unregister_sock
eloop.c, 503
eloop.h, 515
eloop_wait_for_read_sock
eloop.c, 503
eloop.h, 515
eloop_none.c, 519
eloop_win.c, 527
eloop_win.c, 520
eloop_cancel_timeout, 522
eloop_destroy, 522
eloop_get_user_data, 522
eloop_init, 523
eloop_register_event, 523
eloop_register_read_sock, 523
eloop_register_signal, 524
eloop_register_signal_reconfig, 524
eloop_register_signal_terminate, 525
eloop_register_timeout, 525
eloop_run, 526
eloop_terminate, 526
eloop_terminated, 526
eloop_unregister_event, 526
eloop_unregister_read_sock, 527
eloop_wait_for_read_sock, 527
engine
wpa_ssid, 89
engine_id
wpa_ssid, 89
EVENT_ASSOC
wpa_supplicant.h, 916
EVENT_ASSOCINFO

wpa_supplicant.h, 917
EVENT_DISASSOC
wpa_supplicant.h, 916
EVENT_INTERFACE_STATUS
wpa_supplicant.h, 917
EVENT_MICHAEL_MIC_FAILURE
wpa_supplicant.h, 917
EVENT_PMKID_CANDIDATE
wpa_supplicant.h, 917
EVENT_SCAN_RESULTS
wpa_supplicant.h, 917
EVENT_STKSTART
wpa_supplicant.h, 917
events.c, 528
wpa_supplicant_event, 530
wpa_supplicant_scard_init, 530

fast_reauth
wpa_config, 43
fips186_2_prf
crypto.h, 208
flush_pmkid
wpa_driver_ops, 54
fragment_size
wpa_ssid, 89
free
eap_method, 16
freq
wpa_driver_associate_params, 49

generate_authenticator_response
ms_funcs.c, 592
ms_funcs.h, 599
generate_authenticator_response_pwhash
ms_funcs.c, 592
ms_funcs.h, 600
generate_nt_response
ms_funcs.c, 593
ms_funcs.h, 601
generate_nt_response_pwhash
ms_funcs.c, 593
ms_funcs.h, 601
get_asymetric_start_key
ms_funcs.c, 594
ms_funcs.h, 601
get_bool
eapol_callbacks, 25
get_bssid
wpa_driver_ops, 55
get_capa
wpa_driver_ops, 55
get_config
eapol_callbacks, 25
get_config_blob
eapol_callbacks, 25
eapol_ctx, 32
get_eapReqData
eapol_callbacks, 25
get_emsk
eap_method, 17
get_hw_feature_data
wpa_driver_ops, 55
get_identity
eap_method, 17
get_ifname
wpa_driver_ops, 55
get_int
eapol_callbacks, 25
get_mac_addr
wpa_driver_ops, 56
get_master_key
ms_funcs.c, 594
ms_funcs.h, 602
get_scan_results
wpa_driver_ops, 56
get_ssid
wpa_driver_ops, 56
get_status
eap_method, 17
getKey
eap_method, 18
GETU32
aes.c, 101

has_reauth_data
eap_method, 18
hash_nt_password_hash
ms_funcs.c, 595
ms_funcs.h, 602
hexstr2bin
common.c, 146
common.h, 156
hmac_md5
md5.c, 581
md5.h, 584
hmac_md5_vector
md5.c, 581
md5.h, 584
hmac_sha1
sha1.c, 699
sha1.h, 704
hmac_sha1_vector
sha1.c, 700
sha1.h, 704
hmac_sha256
sha256.c, 709
sha256.h, 711
hmac_sha256_vector

sha256.c, 709
sha256.h, 712
hwaddr_aton
common.c, 146
common.h, 157

id
wpa_ssid, 89
id_str
wpa_ssid, 89
inc_byte_array
common.c, 146
common.h, 157
includes.h, 531
index
wpa_event_data::pmkid_candidate, 70
init
eap_method, 18
wpa_driver_ops, 57
init_for_reauth
eap_method, 19
IOCTL_NDISUIO_SET_ETHER_TYPE
l2_packet_ndis.c, 551
isKeyAvailable
eap_method, 19

key_id
wpa_ssid, 90
key_mgmt
wpa_ssid, 90

l2_packet.h, 535
l2_packet_deinit, 538
l2_packet_get_ip_addr, 538
l2_packet_get_own_addr, 538
l2_packet_init, 539
l2_packet_notify_auth_start, 539
l2_packet_send, 539
l2_packet_deinit
l2_packet.h, 538
l2_packet_freebsd.c, 542
l2_packet_linux.c, 546
l2_packet_ndis.c, 551
l2_packet_none.c, 555
l2_packet_pcap.c, 559
l2_packet_winpcap.c, 565
l2_packet_freebsd.c, 541
l2_packet_deinit, 542
l2_packet_get_ip_addr, 542
l2_packet_get_own_addr, 543
l2_packet_init, 543
l2_packet_notify_auth_start, 544
l2_packet_send, 544
l2_packet_get_ip_addr
l2_packet.h, 538
l2_packet_freebsd.c, 542
l2_packet_linux.c, 547
l2_packet_ndis.c, 551
l2_packet_none.c, 556
l2_packet_pcap.c, 560
l2_packet_winpcap.c, 565
l2_packet_get_own_addr
l2_packet.h, 538
l2_packet_freebsd.c, 543
l2_packet_linux.c, 547
l2_packet_ndis.c, 551
l2_packet_none.c, 556
l2_packet_pcap.c, 560
l2_packet_winpcap.c, 565
l2_packet_init
l2_packet.h, 539
l2_packet_freebsd.c, 543
l2_packet_linux.c, 547
l2_packet_ndis.c, 552
l2_packet_none.c, 556
l2_packet_pcap.c, 560
l2_packet_winpcap.c, 565
l2_packet_linux.c, 545
l2_packet_deinit, 546
l2_packet_get_ip_addr, 547
l2_packet_get_own_addr, 547
l2_packet_init, 547
l2_packet_notify_auth_start, 548
l2_packet_send, 548
l2_packet_ndis.c, 549
IOCTL_NDISUIO_SET_ETHER_TYPE, 551
l2_packet_deinit, 551
l2_packet_get_ip_addr, 551
l2_packet_get_own_addr, 551
l2_packet_init, 552
l2_packet_notify_auth_start, 552
l2_packet_send, 553
l2_packet_none.c, 554
l2_packet_deinit, 555
l2_packet_get_ip_addr, 556
l2_packet_get_own_addr, 556
l2_packet_init, 556
l2_packet_notify_auth_start, 557
l2_packet_send, 557
l2_packet_notify_auth_start
l2_packet.h, 539
l2_packet_freebsd.c, 544
l2_packet_linux.c, 548
l2_packet_ndis.c, 552
l2_packet_none.c, 557
l2_packet_pcap.c, 561
l2_packet_winpcap.c, 566
l2_packet_pcap.c, 558

l2_packet_deinit, 559
l2_packet_get_ip_addr, 560
l2_packet_get_own_addr, 560
l2_packet_init, 560
l2_packet_notify_auth_start, 561
l2_packet_send, 561
l2_packet_send
l2_packet.h, 539
l2_packet_freebsd.c, 544
l2_packet_linux.c, 548
l2_packet_ndis.c, 553
l2_packet_none.c, 557
l2_packet_pcap.c, 561
l2_packet_winpcap.c, 566
l2_packet_winpcap.c, 563
l2_packet_deinit, 565
l2_packet_get_ip_addr, 565
l2_packet_get_own_addr, 565
l2_packet_init, 565
l2_packet_notify_auth_start, 566
l2_packet_send, 566
LE_READ_4
driver_bsd.c, 297
leap
wpa_ssid, 90
libtommath.c, 568

main.c, 570
main_none.c, 572
main_winmain.c, 574
main_winsvc.c, 576
md4.c, 578
md4_vector
crypto.c, 195
crypto.h, 208
crypto_cryptoapi.c, 211
crypto_gnutls.c, 214
crypto_libtomcrypt.c, 217
crypto_none.c, 220
md5.c, 580
hmac_md5, 581
hmac_md5_vector, 581
md5.h, 583
hmac_md5, 584
hmac_md5_vector, 584
md5_vector
crypto.h, 209
mlme.c, 585
mlme.h, 589
mlme_add_sta
wpa_driver_ops, 57
mlme_remove_sta
wpa_driver_ops, 57
mlme_setprotection
wpa_driver_ops, 58
mode
wpa_ssid, 90
ms_funcs.c, 590
challenge_response, 592
generate_authenticator_response, 592
generate_authenticator_response_pwhash, 592
generate_nt_response, 593
generate_nt_response_pwhash, 593
get_asymetric_start_key, 594
get_master_key, 594
hash_nt_password_hash, 595
new_password_encrypted_with_old_nt_-
password_hash, 595
nt_challenge_response, 595
nt_password_hash, 596
old_nt_password_hash_encrypted_with_-
new_nt_password_hash, 596
ms_funcs.h, 598
challenge_response, 599
generate_authenticator_response, 599
generate_authenticator_response_pwhash, 600
generate_nt_response, 601
generate_nt_response_pwhash, 601
get_asymetric_start_key, 601
get_master_key, 602
hash_nt_password_hash, 602
new_password_encrypted_with_old_nt_-
password_hash, 602
nt_challenge_response, 603
nt_password_hash, 603
old_nt_password_hash_encrypted_with_-
new_nt_password_hash, 604
mschapv2_retry
wpa_ssid, 90

name
wpa_driver_ops, 58
ndis_events.c, 605
new_password
wpa_ssid, 90
new_password_encrypted_with_old_nt_-
password_hash
ms_funcs.c, 595
ms_funcs.h, 602
next
eap_method, 19
wpa_ssid, 90
non_leap
wpa_ssid, 91
notify_pending
eapol_callbacks, 26
nt_challenge_response
ms_funcs.c, 595

ms_funcs.h, 603
nt_password_hash
ms_funcs.c, 596
ms_funcs.h, 603
num_prio
wpa_config, 43

old_nt_password_hash_encrypted_with_new_nt_-password_hash
ms_funcs.c, 596
ms_funcs.h, 604
omac1_aes_128
aes_wrap.c, 111
aes_wrap.h, 118
opensc_engine_path
eap_config, 13
eapol_ctx, 32
wpa_config, 43
os.h, 607
os_daemonize, 609
os_daemonize_terminate, 609
os_get_random, 610
os_get_time, 610
os_mktime, 610
os_program_deinit, 610
os_program_init, 611
os_random, 611
os_readfile, 611
os_rel2abs_path, 611
os_setenv, 612
os_sleep, 612
os_time_before, 609
os_time_sub, 609
os_unsetenv, 612
os_zalloc, 612
os_daemonize
os.h, 609
os_internal.c, 616
os_none.c, 622
os_unix.c, 628
os_win32.c, 634
os_daemonize_terminate
os.h, 609
os_internal.c, 616
os_none.c, 622
os_unix.c, 628
os_win32.c, 634
os_get_random
os.h, 610
os_internal.c, 616
os_none.c, 622
os_unix.c, 628
os_win32.c, 634
os_get_time
os.h, 610
os_internal.c, 617
os_none.c, 622
os_unix.c, 628
os_win32.c, 634
os_internal.c, 614
os_daemonize, 616
os_daemonize_terminate, 616
os_get_random, 616
os_get_time, 617
os_mktime, 617
os_program_deinit, 617
os_program_init, 617
os_random, 618
os_readfile, 618
os_rel2abs_path, 618
os_setenv, 618
os_sleep, 619
os_unsetenv, 619
os_zalloc, 619
os_mktime
os.h, 610
os_internal.c, 617
os_none.c, 622
os_unix.c, 628
os_win32.c, 634
os_none.c, 620
os_daemonize, 622
os_daemonize_terminate, 622
os_get_random, 622
os_get_time, 622
os_mktime, 622
os_program_deinit, 623
os_program_init, 623
os_random, 623
os_readfile, 623
os_rel2abs_path, 624
os_setenv, 624
os_sleep, 624
os_unsetenv, 625
os_zalloc, 625
os_program_deinit
os.h, 610
os_internal.c, 617
os_none.c, 623
os_unix.c, 629
os_win32.c, 635
os_program_init
os.h, 611
os_internal.c, 617
os_none.c, 623
os_unix.c, 629
os_win32.c, 635
os_random

os.h, 611
os_internal.c, 618
os_none.c, 623
os_unix.c, 629
os_win32.c, 635
os_readfile
os.h, 611
os_internal.c, 618
os_none.c, 623
os_unix.c, 629
os_win32.c, 635
os_rel2abs_path
os.h, 611
os_internal.c, 618
os_none.c, 624
os_unix.c, 630
os_win32.c, 636
os_setenv
os.h, 612
os_internal.c, 618
os_none.c, 624
os_unix.c, 630
os_win32.c, 636
os_sleep
os.h, 612
os_internal.c, 619
os_none.c, 624
os_unix.c, 630
os_win32.c, 636
os_time_before
os.h, 609
os_time_sub
os.h, 609
os_unix.c, 626
os_daemonize, 628
os_daemonize_terminate, 628
os_get_random, 628
os_get_time, 628
os_mktime, 628
os_program_deinit, 629
os_program_init, 629
os_random, 629
os_readfile, 629
os_rel2abs_path, 630
os_setenv, 630
os_sleep, 630
os_unsetenv, 631
os_zalloc, 631
os_unsetenv
os.h, 612
os_internal.c, 619
os_none.c, 625
os_unix.c, 631
os_win32.c, 637
os_win32.c, 632
os_daemonize, 634
os_daemonize_terminate, 634
os_get_random, 634
os_get_time, 634
os_mktime, 634
os_program_deinit, 635
os_program_init, 635
os_random, 635
os_readfile, 635
os_rel2abs_path, 636
os_setenv, 636
os_sleep, 636
os_unsetenv, 637
os_zalloc, 637
os_zalloc
os.h, 612
os_internal.c, 619
os_none.c, 625
os_unix.c, 631
os_win32.c, 637
otp
wpa_ssid, 91

pac_file
wpa_ssid, 91
passphrase
wpa_ssid, 91
pbkdf2_sha1
sha1.c, 700
sha1.h, 705
pcsc
wpa_ssid, 91
pcsc_funcs.c, 638
scard_deinit, 640
scard_get_imsi, 640
scard_gsm_auth, 641
scard_init, 641
scard_set_pin, 642
scard_umts_auth, 642
pcsc_funcs.h, 644
peerkey
wpa_ssid, 91
pending_req_identity
wpa_ssid, 92
pending_req_new_password
wpa_ssid, 92
pending_req_otp
wpa_ssid, 92
pending_req_passphrase
wpa_ssid, 92
pending_req_password
wpa_ssid, 92
pending_req_pin

wpa_ssid, 92
phase1
wpa_ssid, 93
phase2
wpa_ssid, 93
pid_file
wpa_params, 75
pin
wpa_ssid, 93
pkcs11_engine_path
eap_config, 13
eapol_ctx, 32
wpa_config, 44
pkcs11_module_path
eap_config, 14
eapol_ctx, 32
wpa_config, 44
pmksa_cache.c, 646
pmksa_cache_add, 647
pmksa_cache_clear_current, 648
pmksa_cache_deinit, 648
pmksa_cache_get, 648
pmksa_cache_get_current, 649
pmksa_cache_get_opportunistic, 649
pmksa_cache_init, 649
pmksa_cache_list, 650
pmksa_cache_notify_reconfig, 650
pmksa_cache_set_current, 650
pmksa_cache.h, 652
pmksa_cache_add, 653
pmksa_cache_clear_current, 654
pmksa_cache_deinit, 654
pmksa_cache_get, 654
pmksa_cache_get_current, 654
pmksa_cache_get_opportunistic, 654
pmksa_cache_init, 655
pmksa_cache_list, 655
pmksa_cache_notify_reconfig, 656
pmksa_cache_set_current, 656
pmksa_cache_add
pmksa_cache.c, 647
pmksa_cache.h, 653
pmksa_cache_clear_current
pmksa_cache.c, 648
pmksa_cache.h, 654
pmksa_cache_deinit
pmksa_cache.c, 648
pmksa_cache.h, 654
pmksa_cache_get
pmksa_cache.c, 648
pmksa_cache.h, 654
pmksa_cache_get_current
pmksa_cache.c, 649
pmksa_cache.h, 654
pmksa_cache_get_opportunistic
pmksa_cache.c, 649
pmksa_cache.h, 654
pmksa_cache_init
pmksa_cache.c, 649
pmksa_cache.h, 655
pmksa_cache_list
pmksa_cache.c, 650
pmksa_cache.h, 655
pmksa_cache_notify_reconfig
pmksa_cache.c, 650
pmksa_cache.h, 656
pmksa_cache_set_current
pmksa_cache.c, 650
pmksa_cache.h, 656
pmksa_candidate_add
preauth.c, 658
preauth.h, 664
pmksa_candidate_free
preauth.c, 659
preauth.h, 664
pnext
wpa_ssid, 93
poll
wpa_driver_ops, 58
preauth
eapol_ctx, 32
wpa_event_data::pmkid_candidate, 70
preauth.c, 657
pmksa_candidate_add, 658
pmksa_candidate_free, 659
rsn_preauth_candidate_process, 659
rsn_preauth_deinit, 660
rsn_preauth_get_status, 660
rsn_preauth_in_progress, 661
rsn_preauth_init, 661
rsn_preauth_scan_results, 662
preauth.h, 663
pmksa_candidate_add, 664
pmksa_candidate_free, 664
rsn_preauth_candidate_process, 664
rsn_preauth_deinit, 665
rsn_preauth_get_status, 665
rsn_preauth_in_progress, 666
rsn_preauth_init, 666
rsn_preauth_scan_results, 667
preauth_test.c, 669
priority
wpa_ssid, 94
priv_netlink.h, 672
RTA_NEXT, 673
RTA_OK, 673
private_key
wpa_ssid, 94

private_key2
wpa_ssid, 94
private_key2_passwd
wpa_ssid, 94
private_key_passwd
wpa_ssid, 95
proactive_key_caching
wpa_ssid, 95
process
eap_method, 19
PUTU32
aes.c, 101

radius.c, 674
radius_msg_get_vlanid, 676
radius.h, 677
radius_msg_get_vlanid, 680
radius_client.c, 681
radius_client.h, 683
radius_msg_get_vlanid
radius.c, 676
radius.h, 680
rc4
rc4.c, 686
rc4.h, 688
rc4.c, 685
rc4, 686
rc4_skip, 686
rc4.h, 688
rc4, 688
rc4_skip, 689
rc4_skip
rc4.c, 686
rc4.h, 689
remove_pmkid
wpa_driver_ops, 58
req_ies
wpa_event_data::assoc_info, 66
required_keys
eapol_config, 28
resp_ies
wpa_event_data::assoc_info, 67
rijndaelKeySetupDec
aes.c, 103
rijndaelKeySetupEnc
aes.c, 104
ROUND
aes.c, 101, 102
rsa.c, 690
crypto_rsa_exptmod, 691
crypto_rsa_free, 691
crypto_rsa_get_modulus_len, 692
crypto_rsa_import_private_key, 692
crypto_rsa_import_public_key, 693
rsa.h, 694
crypto_rsa_exptmod, 694
crypto_rsa_free, 695
crypto_rsa_get_modulus_len, 695
crypto_rsa_import_private_key, 696
crypto_rsa_import_public_key, 696
rsn_pmksa_cache_entry, 37
rsn_preauth_candidate_process
preauth.c, 659
preauth.h, 664
rsn_preauth_deinit
preauth.c, 660
preauth.h, 665
rsn_preauth_get_status
preauth.c, 660
preauth.h, 665
rsn_preauth_in_progress
preauth.c, 661
preauth.h, 666
rsn_preauth_init
preauth.c, 661
preauth.h, 666
rsn_preauth_scan_results
preauth.c, 662
preauth.h, 667
RTA_NEXT
priv_netlink.h, 673
RTA_OK
priv_netlink.h, 673

scan
wpa_driver_ops, 59
scan_ssid
wpa_ssid, 95
scard_ctx
eapol_ctx, 33
scard_deinit
pcsc_funcs.c, 640
scard_get_imsi
pcsc_funcs.c, 640
scard_gsm_auth
pcsc_funcs.c, 641
scard_init
pcsc_funcs.c, 641
scard_set_pin
pcsc_funcs.c, 642
scard_umts_auth
pcsc_funcs.c, 642
send_eapol
wpa_driver_ops, 59
send_mlme
wpa_driver_ops, 60
set_auth_alg
wpa_driver_ops, 60

set_bool
eapol_callbacks, 26
set_bssid
wpa_driver_ops, 60
set_channel
wpa_driver_ops, 60
set_config_blob
eapol_callbacks, 26
eapol_ctx, 33
set_countermeasures
wpa_driver_ops, 61
set_drop_unencrypted
wpa_driver_ops, 61
set_int
eapol_callbacks, 26
set_key
wpa_driver_ops, 61
set_operstate
wpa_driver_ops, 62
set_param
wpa_driver_ops, 62
set_ssid
wpa_driver_ops, 63
set_wep_key
eapol_ctx, 33
set_wpa
wpa_driver_ops, 63
sha1.c, 698
hmac_sha1, 699
hmac_sha1_vector, 700
pbkdf2_sha1, 700
sha1_prf, 700
sha1_t_prf, 701
tls_prf, 701
sha1.h, 703
hmac_sha1, 704
hmac_sha1_vector, 704
pbkdf2_sha1, 705
sha1_prf, 705
sha1_t_prf, 706
tls_prf, 706
sha1_prf
sha1.c, 700
sha1.h, 705
sha1_t_prf
sha1.c, 701
sha1.h, 706
sha1_vector
crypto.h, 209
sha256.c, 708
hmac_sha256, 709
hmac_sha256_vector, 709
sha256_prf, 710
sha256.h, 711
hmac_sha256, 711
hmac_sha256_vector, 712
sha256_prf, 712
sha256_prf
sha256.c, 710
sha256.h, 712
sha256_vector
crypto.h, 209
SM_ENTER
state_machine.h, 715
SM_ENTER_GLOBAL
state_machine.h, 715
SM_ENTRY
state_machine.h, 715
SM_ENTRY_M
state_machine.h, 716
SM_ENTRY_MA
state_machine.h, 716
SM_STATE
state_machine.h, 716
SM_STEP
state_machine.h, 717
SM_STEP_RUN
state_machine.h, 717
ssid
wpa_config, 44
wpa_ssid, 95
state_machine.h, 714
SM_ENTER, 715
SM_ENTER_GLOBAL, 715
SM_ENTRY, 715
SM_ENTRY_M, 716
SM_ENTRY_MA, 716
SM_STATE, 716
SM_STEP, 717
SM_STEP_RUN, 717
STRUCT_PACKED
wpa_i.h, 892
subject_match
wpa_ssid, 95
subject_match2
wpa_ssid, 96

tests/test_aes.c, 718
tests/test_eap_sim_common.c, 720
tests/test_md4.c, 721
tests/test_md5.c, 723
tests/test_ms_funcs.c, 725
tests/test_sha1.c, 726
tests/test_sha256.c, 728
tests/test_x509v3.c, 730
tls.h, 732
tls_capabilities, 735
tls_connection_client_hello_ext, 735

tls_connection_decrypt, 735
tls_connection_deinit, 736
tls_connection_enable_workaround, 736
tls_connection_encrypt, 736
tls_connection_established, 737
tls_connection_get_failed, 737
tls_connection_get_keyblock_size, 737
tls_connection_get_keys, 738
tls_connection_get_read_alerts, 738
tls_connection_get_write_alerts, 738
tls_connection_handshake, 738
tls_connection_ia_final_phase_finished, 739
tls_connection_ia_permute_inner_secret, 739
tls_connection_ia_send_phase_finished, 740
tls_connection_init, 740
tls_connection_prf, 740
tls_connection_resumed, 741
tls_connection_server_handshake, 741
tls_connection_set_cipher_list, 742
tls_connection_set_ia, 742
tls_connection_set_master_key, 742
tls_connection_set_params, 742
tls_connection_set_verify, 743
tls_connection_shutdown, 743
tls_deinit, 743
tls_get_cipher, 744
tls_get_errors, 744
tls_global_set_params, 744
tls_global_set_verify, 745
tls_init, 745
tls_capabilities
tls.h, 735
tls_gnutls.c, 749
tls_internal.c, 765
tls_openssl.c, 785
tls_schannel.c, 799
tls_connection_client_hello_ext
tls.h, 735
tls_gnutls.c, 749
tls_internal.c, 765
tls_schannel.c, 799
tls_connection_decrypt
tls.h, 735
tls_gnutls.c, 750
tls_internal.c, 766
tls_openssl.c, 785
tls_schannel.c, 800
tls_connection_deinit
tls.h, 736
tls_gnutls.c, 750
tls_internal.c, 766
tls_openssl.c, 786
tls_schannel.c, 800
tls_connection_enable_workaround
tls.h, 736
tls_gnutls.c, 751
tls_internal.c, 767
tls_openssl.c, 786
tls_schannel.c, 801
tls_connection_encrypt
tls.h, 736
tls_gnutls.c, 751
tls_internal.c, 767
tls_openssl.c, 786
tls_schannel.c, 801
tls_connection_established
tls.h, 737
tls_gnutls.c, 751
tls_internal.c, 768
tls_openssl.c, 786
tls_schannel.c, 802
tls_connection_get_failed
tls.h, 737
tls_gnutls.c, 752
tls_internal.c, 768
tls_openssl.c, 787
tls_schannel.c, 802
tls_connection_get_keyblock_size
tls.h, 737
tls_gnutls.c, 752
tls_internal.c, 769
tls_openssl.c, 787
tls_connection_get_keys
tls.h, 738
tls_gnutls.c, 752
tls_internal.c, 769
tls_openssl.c, 787
tls_schannel.c, 802
tls_connection_get_read_alerts
tls.h, 738
tls_gnutls.c, 752
tls_internal.c, 769
tls_openssl.c, 787
tls_schannel.c, 802
tls_connection_get_write_alerts
tls.h, 738
tls_gnutls.c, 753
tls_internal.c, 770
tls_openssl.c, 788
tls_schannel.c, 803
tls_connection_handshake
tls.h, 738
tls_gnutls.c, 753
tls_internal.c, 770
tls_openssl.c, 788
tls_schannel.c, 803
tls_connection_ia_final_phase_finished
tls.h, 739

tls_gnutls.c, 754
tls_internal.c, 771
tls_openssl.c, 789
tls_schannel.c, 804
tls_connection_ia_permute_inner_secret
tls.h, 739
tls_gnutls.c, 754
tls_internal.c, 771
tls_openssl.c, 789
tls_schannel.c, 804
tls_connection_ia_send_phase_finished
tls.h, 740
tls_gnutls.c, 754
tls_internal.c, 771
tls_openssl.c, 789
tls_schannel.c, 804
tls_connection_init
tls.h, 740
tls_gnutls.c, 755
tls_internal.c, 772
tls_openssl.c, 790
tls_schannel.c, 805
tls_connection_params, 38
tls_connection_prf
tls.h, 740
tls_gnutls.c, 755
tls_internal.c, 772
tls_openssl.c, 790
tls_schannel.c, 805
tls_connection_resumed
tls.h, 741
tls_gnutls.c, 756
tls_internal.c, 773
tls_openssl.c, 791
tls_schannel.c, 806
tls_connection_server_handshake
tls.h, 741
tls_gnutls.c, 756
tls_internal.c, 773
tls_openssl.c, 791
tls_schannel.c, 806
tls_connection_set_cipher_list
tls.h, 742
tls_gnutls.c, 756
tls_internal.c, 774
tls_openssl.c, 791
tls_schannel.c, 806
tls_connection_set_ia
tls.h, 742
tls_gnutls.c, 757
tls_internal.c, 774
tls_openssl.c, 792
tls_schannel.c, 807
tls_connection_set_master_key
tls.h, 742
tls_gnutls.c, 757
tls_internal.c, 774
tls_schannel.c, 807
tls_connection_set_params
tls.h, 742
tls_gnutls.c, 757
tls_internal.c, 775
tls_openssl.c, 792
tls_schannel.c, 807
tls_connection_set_verify
tls.h, 743
tls_gnutls.c, 758
tls_internal.c, 775
tls_openssl.c, 793
tls_schannel.c, 808
tls_connection_shutdown
tls.h, 743
tls_gnutls.c, 758
tls_internal.c, 776
tls_openssl.c, 793
tls_schannel.c, 808
tls_deinit
tls.h, 743
tls_gnutls.c, 759
tls_internal.c, 776
tls_none.c, 781
tls_openssl.c, 793
tls_schannel.c, 808
tls_get_cipher
tls.h, 744
tls_gnutls.c, 759
tls_internal.c, 777
tls_openssl.c, 793
tls_schannel.c, 809
tls_get_cipher_suite
tlsv1_common.c, 833
tlsv1_common.h, 841
tls_get_errors
tls.h, 744
tls_gnutls.c, 759
tls_internal.c, 777
tls_openssl.c, 794
tls_schannel.c, 809
tls_global_set_params
tls.h, 744
tls_gnutls.c, 760
tls_internal.c, 777
tls_openssl.c, 794
tls_schannel.c, 809
tls_global_set_verify
tls.h, 745
tls_gnutls.c, 760
tls_internal.c, 778

tls_openssl.c, 795
tls_schannel.c, 810
tls_gnutls.c, 746
tls_capabilities, 749
tls_connection_client_hello_ext, 749
tls_connection_decrypt, 750
tls_connection_deinit, 750
tls_connection_enable_workaround, 751
tls_connection_encrypt, 751
tls_connection_established, 751
tls_connection_get_failed, 752
tls_connection_get_keyblock_size, 752
tls_connection_get_keys, 752
tls_connection_get_read_alerts, 752
tls_connection_get_write_alerts, 753
tls_connection_handshake, 753
tls_connection_ia_final_phase_finished, 754
tls_connection_ia_permute_inner_secret, 754
tls_connection_ia_send_phase_finished, 754
tls_connection_init, 755
tls_connection_prf, 755
tls_connection_resumed, 756
tls_connection_server_handshake, 756
tls_connection_set_cipher_list, 756
tls_connection_set_ia, 757
tls_connection_set_master_key, 757
tls_connection_set_params, 757
tls_connection_set_verify, 758
tls_connection_shutdown, 758
tls_deinit, 759
tls_get_cipher, 759
tls_get_errors, 759
tls_global_set_params, 760
tls_global_set_verify, 760
tls_init, 760
tls_init
tls.h, 745
tls_gnutls.c, 760
tls_internal.c, 778
tls_none.c, 781
tls_openssl.c, 795
tls_schannel.c, 810
tls_internal.c, 762
tls_capabilities, 765
tls_connection_client_hello_ext, 765
tls_connection_decrypt, 766
tls_connection_deinit, 766
tls_connection_enable_workaround, 767
tls_connection_encrypt, 767
tls_connection_established, 768
tls_connection_get_failed, 768
tls_connection_get_keyblock_size, 769
tls_connection_get_keys, 769
tls_connection_get_read_alerts, 769
tls_connection_get_write_alerts, 770
tls_connection_handshake, 770
tls_connection_ia_final_phase_finished, 771
tls_connection_ia_permute_inner_secret, 771
tls_connection_ia_send_phase_finished, 771
tls_connection_init, 772
tls_connection_prf, 772
tls_connection_resumed, 773
tls_connection_server_handshake, 773
tls_connection_set_cipher_list, 774
tls_connection_set_ia, 774
tls_connection_set_master_key, 774
tls_connection_set_params, 775
tls_connection_set_verify, 775
tls_connection_shutdown, 776
tls_deinit, 776
tls_get_cipher, 777
tls_get_errors, 777
tls_global_set_params, 777
tls_global_set_verify, 778
tls_init, 778
TLS_MAX_KEY_BLOCK_LEN
tlsv1_common.h, 841
tls_none.c, 780
tls_deinit, 781
tls_init, 781
tls_openssl.c, 782
tls_capabilities, 785
tls_connection_decrypt, 785
tls_connection_deinit, 786
tls_connection_enable_workaround, 786
tls_connection_encrypt, 786
tls_connection_established, 786
tls_connection_get_failed, 787
tls_connection_get_keyblock_size, 787
tls_connection_get_keys, 787
tls_connection_get_read_alerts, 787
tls_connection_get_write_alerts, 788
tls_connection_handshake, 788
tls_connection_ia_final_phase_finished, 789
tls_connection_ia_permute_inner_secret, 789
tls_connection_ia_send_phase_finished, 789
tls_connection_init, 790
tls_connection_prf, 790
tls_connection_resumed, 791
tls_connection_server_handshake, 791
tls_connection_set_cipher_list, 791
tls_connection_set_ia, 792
tls_connection_set_params, 792
tls_connection_set_verify, 793
tls_connection_shutdown, 793
tls_deinit, 793
tls_get_cipher, 793
tls_get_errors, 794

tls_global_set_params, 794
tls_global_set_verify, 795
tls_init, 795
tls_parse_cert
tlsv1_common.c, 833
tlsv1_common.h, 841
tls_prf
sha1.c, 701
sha1.h, 706
tls_schannel.c, 796
tls_capabilities, 799
tls_connection_client_hello_ext, 799
tls_connection_decrypt, 800
tls_connection_deinit, 800
tls_connection_enable_workaround, 801
tls_connection_encrypt, 801
tls_connection_established, 802
tls_connection_get_failed, 802
tls_connection_get_keys, 802
tls_connection_get_read_alerts, 802
tls_connection_get_write_alerts, 803
tls_connection_handshake, 803
tls_connection_ia_final_phase_finished, 804
tls_connection_ia_permute_inner_secret, 804
tls_connection_ia_send_phase_finished, 804
tls_connection_init, 805
tls_connection_prf, 805
tls_connection_resumed, 806
tls_connection_server_handshake, 806
tls_connection_set_cipher_list, 806
tls_connection_set_ia, 807
tls_connection_set_master_key, 807
tls_connection_set_params, 807
tls_connection_set_verify, 808
tls_connection_shutdown, 808
tls_deinit, 808
tls_get_cipher, 809
tls_get_errors, 809
tls_global_set_params, 809
tls_global_set_verify, 810
tls_init, 810
tlsv1_client.c, 811
tlsv1_client_decrypt, 813
tlsv1_client_deinit, 814
tlsv1_client_encrypt, 814
tlsv1_client_established, 815
tlsv1_client_get_cipher, 815
tlsv1_client_get_keyblock_size, 815
tlsv1_client_get_keys, 816
tlsv1_client_global_deinit, 816
tlsv1_client_global_init, 816
tlsv1_client_handshake, 817
tlsv1_client_hello_ext, 817
tlsv1_client_init, 818
tlsv1_client_prf, 818
tlsv1_client_resumed, 818
tlsv1_client_set_ca_cert, 819
tlsv1_client_set_cipher_list, 819
tlsv1_client_set_client_cert, 819
tlsv1_client_set_master_key, 820
tlsv1_client_set_private_key, 820
tlsv1_client_shutdown, 821
tlsv1_client.h, 822
tlsv1_client_decrypt, 823
tlsv1_client_deinit, 824
tlsv1_client_encrypt, 824
tlsv1_client_established, 825
tlsv1_client_get_cipher, 825
tlsv1_client_get_keyblock_size, 826
tlsv1_client_get_keys, 826
tlsv1_client_global_deinit, 826
tlsv1_client_global_init, 826
tlsv1_client_handshake, 827
tlsv1_client_hello_ext, 827
tlsv1_client_init, 828
tlsv1_client_prf, 828
tlsv1_client_resumed, 829
tlsv1_client_set_ca_cert, 829
tlsv1_client_set_cipher_list, 829
tlsv1_client_set_client_cert, 830
tlsv1_client_set_master_key, 830
tlsv1_client_set_private_key, 830
tlsv1_client_shutdown, 831
tlsv1_client_decrypt
tlsv1_client.c, 813
tlsv1_client.h, 823
tlsv1_client_deinit
tlsv1_client.c, 814
tlsv1_client.h, 824
tlsv1_client_encrypt
tlsv1_client.c, 814
tlsv1_client.h, 824
tlsv1_client_established
tlsv1_client.c, 815
tlsv1_client.h, 825
tlsv1_client_get_cipher
tlsv1_client.c, 815
tlsv1_client.h, 825
tlsv1_client_get_keyblock_size
tlsv1_client.c, 815
tlsv1_client.h, 826
tlsv1_client_get_keys
tlsv1_client.c, 816
tlsv1_client.h, 826
tlsv1_client_global_deinit
tlsv1_client.c, 816
tlsv1_client.h, 826
tlsv1_client_global_init

tlsv1_client.c, 816
tlsv1_client.h, 826
tlsv1_client_handshake
tlsv1_client.c, 817
tlsv1_client.h, 827
tlsv1_client_hello_ext
tlsv1_client.c, 817
tlsv1_client.h, 827
tlsv1_client_init
tlsv1_client.c, 818
tlsv1_client.h, 828
tlsv1_client_prf
tlsv1_client.c, 818
tlsv1_client.h, 828
tlsv1_client_resumed
tlsv1_client.c, 818
tlsv1_client.h, 829
tlsv1_client_set_ca_cert
tlsv1_client.c, 819
tlsv1_client.h, 829
tlsv1_client_set_cipher_list
tlsv1_client.c, 819
tlsv1_client.h, 829
tlsv1_client_set_client_cert
tlsv1_client.c, 819
tlsv1_client.h, 830
tlsv1_client_set_master_key
tlsv1_client.c, 820
tlsv1_client.h, 830
tlsv1_client_set_private_key
tlsv1_client.c, 820
tlsv1_client.h, 830
tlsv1_client_shutdown
tlsv1_client.c, 821
tlsv1_client.h, 831
tlsv1_common.c, 832
tls_get_cipher_suite, 833
tls_parse_cert, 833
tlsv1_record_change_read_cipher, 834
tlsv1_record_change_write_cipher, 834
tlsv1_record_receive, 835
tlsv1_record_send, 835
tlsv1_record_set_cipher_suite, 836
tlsv1_common.h, 838
tls_get_cipher_suite, 841
TLS_MAX_KEY_BLOCK_LEN, 841
tls_parse_cert, 841
tlsv1_record_change_read_cipher, 842
tlsv1_record_change_write_cipher, 842
tlsv1_record_receive, 843
tlsv1_record_send, 844
tlsv1_record_set_cipher_suite, 844
tlsv1_record_change_read_cipher
tlsv1_common.c, 834
tlsv1_common.h, 842
tlsv1_record_change_write_cipher
tlsv1_common.c, 834
tlsv1_common.h, 842
tlsv1_record_receive
tlsv1_common.c, 835
tlsv1_common.h, 843
tlsv1_record_send
tlsv1_common.c, 835
tlsv1_common.h, 844
tlsv1_record_set_cipher_suite
tlsv1_common.c, 836
tlsv1_common.h, 844

update_config
wpa_config, 44

version
eap_method, 20

wait_for_interface
wpa_params, 76
win_if_list.c, 846
wpa.c, 848
wpa_parse_wpa_ie, 851
wpa_sm_aborted_cached, 851
wpa_sm_deinit, 851
wpa_sm_get_mib, 852
wpa_sm_get_param, 852
wpa_sm_get_status, 852
wpa_sm_init, 853
wpa_sm_key_request, 853
wpa_sm_notify_assoc, 854
wpa_sm_notify_disassoc, 854
wpa_sm_parse_own_wpa_ie, 855
wpa_sm_rx_eapol, 855
wpa_sm_set_ap_rsn_ie, 856
wpa_sm_set_ap_wpa_ie, 856
wpa_sm_set_assoc_wpa_ie, 857
wpa_sm_set_assoc_wpa_ie_default, 857
wpa_sm_set_config, 857
wpa_sm_set_eapol, 858
wpa_sm_set_fast_reauth, 858
wpa_sm_set_ifname, 858
wpa_sm_set_own_addr, 858
wpa_sm_set_param, 859
wpa_sm_set_pmk, 859
wpa_sm_set_pmk_from_pmksa, 859
wpa_sm_set_scard_ctx, 859
wpa_sm_stkstart, 860
wpa.h, 861
wpa_parse_wpa_ie, 864
wpa_sm_aborted_cached, 864
wpa_sm_deinit, 865

wpa_sm_get_mib, 865
wpa_sm_get_param, 865
wpa_sm_get_status, 866
wpa_sm_init, 866
wpa_sm_key_request, 866
wpa_sm_notify_assoc, 867
wpa_sm_notify_disassoc, 867
wpa_sm_parse_own_wpa_ie, 868
wpa_sm_rx_eapol, 868
wpa_sm_set_ap_rsn_ie, 869
wpa_sm_set_ap_wpa_ie, 869
wpa_sm_set_assoc_wpa_ie, 870
wpa_sm_set_assoc_wpa_ie_default, 870
wpa_sm_set_config, 871
wpa_sm_set_eapol, 871
wpa_sm_set_fast_reauth, 871
wpa_sm_set_ifname, 871
wpa_sm_set_own_addr, 872
wpa_sm_set_param, 872
wpa_sm_set_pmk, 872
wpa_sm_set_pmk_from_pmksa, 872
wpa_sm_set_scard_ctx, 873
wpa_sm_stkstart, 873
WPA_4WAY_HANDSHAKE
defs.h, 284
WPA_ASSOCIATED
defs.h, 284
WPA_ASSOCIATING
defs.h, 284
wpa_blacklist_add
wpa_supplicant.c, 899
wpa_supplicant_i.h, 922
wpa_blacklist_clear
wpa_supplicant.c, 899
wpa_supplicant_i.h, 922
wpa_blacklist_get
wpa_supplicant.c, 900
wpa_supplicant_i.h, 922
wpa_clear_keys
wpa_supplicant.c, 900
wpa_supplicant_i.h, 923
wpa_cli.c, 874
wpa_common.h, 876
WPA_COMPLETED
defs.h, 284
wpa_config, 40
ap_scan, 41
ctrl_interface, 41
ctrl_interface_group, 42
dot11RSNAConfigPMKLifetime, 42
dot11RSNAConfigPMKReauthThreshold, 42
dot11RSNAConfigSATimeout, 43
driver_param, 43
eapol_version, 43
fast_reauth, 43
num_prio, 43
opensc_engine_path, 43
pkcs11_engine_path, 44
pkcs11_module_path, 44
ssid, 44
update_config, 44
wpa_config_add_network
config.c, 165
config.h, 173
wpa_config_add_prio_network
config.c, 165
config.h, 173
wpa_config_alloc_empty
config.c, 165
config.h, 174
wpa_config_allowed_eap_method
config.c, 166
config_ssid.h, 189
wpa_config_blob, 45
wpa_config_debug_dump_networks
config.c, 166
config.h, 174
wpa_config_free
config.c, 166
config.h, 174
wpa_config_free_blob
config.c, 167
config.h, 175
wpa_config_free_ssid
config.c, 167
config.h, 175
wpa_config_get
config.c, 167
config.h, 175
wpa_config_get_blob
config.c, 167
config.h, 175
wpa_config_get_network
config.c, 168
config.h, 176
wpa_config_get_no_key
config.c, 168
config.h, 176
wpa_config_read
config.h, 176
config_file.c, 182
config_none.c, 185
config_winreg.c, 192
wpa_config_remove_blob
config.c, 168
config.h, 177
wpa_config_remove_network
config.c, 169

config.h, 177
wpa_config_set
config.c, 169
config.h, 178
wpa_config_set_blob
config.c, 170
config.h, 178
wpa_config_set_network_defaults
config.c, 170
config.h, 179
wpa_config_update_psk
config.c, 170
config.h, 179
wpa_config_write
config.h, 179
config_file.c, 183
config_none.c, 185
config_winreg.c, 193
wpa_ctrl, 46
wpa_ctrl.c, 878
wpa_ctrl_attach, 879
wpa_ctrl_detach, 879
wpa_ctrl.h, 880
wpa_ctrl_attach, 883
wpa_ctrl_close, 883
wpa_ctrl_detach, 883
wpa_ctrl_get_fd, 883
wpa_ctrl_open, 884
wpa_ctrl_pending, 884
wpa_ctrl_recv, 884
WPA_CTRL_REQ, 881
wpa_ctrl_request, 884
WPA_CTRL_RSP, 881
WPA_EVENT_CONNECTED, 881
WPA_EVENT_DISCONNECTED, 881
WPA_EVENT_EAP_FAILURE, 882
WPA_EVENT_EAP_METHOD, 882
WPA_EVENT_EAP_NOTIFICATION, 882
WPA_EVENT_EAP_STARTED, 882
WPA_EVENT_EAP_SUCCESS, 882
WPA_EVENT_PASSWORD_CHANGED, 882
WPA_EVENT_TERMINATING, 882
wpa_ctrl_attach
wpa_ctrl.c, 879
wpa_ctrl.h, 883
wpa_ctrl_close
wpa_ctrl.h, 883
wpa_ctrl_detach
wpa_ctrl.c, 879
wpa_ctrl.h, 883
wpa_ctrl_dst, 47
wpa_ctrl_get_fd
wpa_ctrl.h, 883
wpa_ctrl_open
wpa_ctrl.h, 884
wpa_ctrl_pending
wpa_ctrl.h, 884
wpa_ctrl_recv
wpa_ctrl.h, 884
WPA_CTRL_REQ
wpa_ctrl.h, 881
wpa_ctrl_request
wpa_ctrl.h, 884
WPA_CTRL_RSP
wpa_ctrl.h, 881
wpa_dbus_dict_append_bool
dbus_dict_helpers.c, 266
dbus_dict_helpers.h, 275
wpa_dbus_dict_append_byte
dbus_dict_helpers.c, 266
dbus_dict_helpers.h, 275
wpa_dbus_dict_append_byte_array
dbus_dict_helpers.c, 267
dbus_dict_helpers.h, 275
wpa_dbus_dict_append_double
dbus_dict_helpers.c, 267
dbus_dict_helpers.h, 276
wpa_dbus_dict_append_int16
dbus_dict_helpers.c, 267
dbus_dict_helpers.h, 276
wpa_dbus_dict_append_int32
dbus_dict_helpers.c, 268
dbus_dict_helpers.h, 276
wpa_dbus_dict_append_int64
dbus_dict_helpers.c, 268
dbus_dict_helpers.h, 277
wpa_dbus_dict_append_object_path
dbus_dict_helpers.c, 268
dbus_dict_helpers.h, 277
wpa_dbus_dict_append_string
dbus_dict_helpers.c, 269
dbus_dict_helpers.h, 277
wpa_dbus_dict_append_string_array
dbus_dict_helpers.c, 269
dbus_dict_helpers.h, 278
wpa_dbus_dict_append_uint16
dbus_dict_helpers.c, 270
dbus_dict_helpers.h, 278
wpa_dbus_dict_append_uint32
dbus_dict_helpers.c, 270
dbus_dict_helpers.h, 279
wpa_dbus_dict_append_uint64
dbus_dict_helpers.c, 270
dbus_dict_helpers.h, 279
wpa_dbus_dict_begin_string_array
dbus_dict_helpers.c, 270
dbus_dict_helpers.h, 279

wpa_dbus_dict_close_write
dbus_dict_helpers.c, 271
dbus_dict_helpers.h, 280
wpa_dbus_dict_end_string_array
dbus_dict_helpers.c, 271
dbus_dict_helpers.h, 280
wpa_dbus_dict_entry_clear
dbus_dict_helpers.c, 271
dbus_dict_helpers.h, 280
wpa_dbus_dict_get_entry
dbus_dict_helpers.c, 272
dbus_dict_helpers.h, 280
wpa_dbus_dict_has_dict_entry
dbus_dict_helpers.c, 272
dbus_dict_helpers.h, 281
wpa_dbus_dict_open_read
dbus_dict_helpers.c, 272
dbus_dict_helpers.h, 281
wpa_dbus_dict_open_write
dbus_dict_helpers.c, 273
dbus_dict_helpers.h, 281
wpa_dbus_dict_string_array_add_element
dbus_dict_helpers.c, 273
dbus_dict_helpers.h, 282
wpa_debug_print_timestamp
common.c, 146
common.h, 157
wpa_debug_show_keys
wpa_params, 76
WPA_DISCONNECTED
defs.h, 284
wpa_driver_associate_params, 48
auth_alg, 49
bssid, 49
freq, 49
wpa_ie, 49
wpa_driver_atmel_ops
driver_atmel.c, 291
wpa_driver_broadcom_ops
driver_broadcom.c, 293
wpa_driver_bsd_ops
driver_bsd.c, 297
wpa_driver_capa, 50
wpa_driver_hostap_ops
driver_hostap.c, 299
wpa_driver_ipw_ops
driver_ipw.c, 303
wpa_driver_madwifi_ops
driver_madwifi.c, 305
wpa_driver_ndiswrapper_ops
driver_ndiswrapper.c, 314
wpa_driver_ops, 51
add_pmkid, 53
associate, 53
deauthenticate, 53
deinit, 54
desc, 54
disassociate, 54
flush_pmkid, 54
get_bssid, 55
get_capa, 55
get_hw_feature_data, 55
get_ifname, 55
get_mac_addr, 56
get_scan_results, 56
get_ssid, 56
init, 57
mlme_add_sta, 57
mlme_remove_sta, 57
mlme_setprotection, 58
name, 58
poll, 58
remove_pmkid, 58
scan, 59
send_eapol, 59
send_mlme, 60
set_auth_alg, 60
set_bssid, 60
set_channel, 60
set_countermeasures, 61
set_drop_unencrypted, 61
set_key, 61
set_operstate, 62
set_param, 62
set_ssid, 63
set_wpa, 63
wpa_driver_prism54_ops
driver_prism54.c, 316
wpa_driver_wext_deinit
driver_wext.c, 321
driver_wext.h, 328
wpa_driver_wext_get_bssid
driver_wext.c, 321
driver_wext.h, 329
wpa_driver_wext_get_ifflags
driver_wext.c, 322
driver_wext.h, 329
wpa_driver_wext_get_scan_results
driver_wext.c, 322
driver_wext.h, 329
wpa_driver_wext_get_ssid
driver_wext.c, 322
driver_wext.h, 330
wpa_driver_wext_init
driver_wext.c, 323
driver_wext.h, 330
wpa_driver_wext_scan
driver_wext.c, 323

driver_wext.h, 330
wpa_driver_wext_scan_timeout
driver_wext.c, 324
driver_wext.h, 331
wpa_driver_wext_set_bssid
driver_wext.c, 324
driver_wext.h, 331
wpa_driver_wext_set_freq
driver_wext.c, 324
driver_wext.h, 332
wpa_driver_wext_set_ifflags
driver_wext.c, 324
driver_wext.h, 332
wpa_driver_wext_set_key
driver_wext.c, 325
driver_wext.h, 332
wpa_driver_wext_set_mode
driver_wext.c, 325
driver_wext.h, 333
wpa_driver_wext_set_ssid
driver_wext.c, 326
driver_wext.h, 333
wpa_driver_wired_ops
driver_wired.c, 335
WPA_EVENT_CONNECTED
wpa_ctrl.h, 881
wpa_event_data, 64
assoc_info, 65
wpa_event_data::assoc_info, 66
beacon_ies, 66
req_ies, 66
resp_ies, 67
wpa_event_data::interface_status, 68
wpa_event_data::michael_mic_failure, 69
wpa_event_data::pmkid_candidate, 70
bssid, 70
index, 70
preauth, 70
wpa_event_data::stkstart, 71
WPA_EVENT_DISCONNECTED
wpa_ctrl.h, 881
WPA_EVENT_EAP_FAILURE
wpa_ctrl.h, 882
WPA_EVENT_EAP_METHOD
wpa_ctrl.h, 882
WPA_EVENT_EAP_NOTIFICATION
wpa_ctrl.h, 882
WPA_EVENT_EAP_STARTED
wpa_ctrl.h, 882
WPA_EVENT_EAP_SUCCESS
wpa_ctrl.h, 882
WPA_EVENT_PASSWORD_CHANGED
wpa_ctrl.h, 882
WPA_EVENT_TERMINATING
wpa_ctrl.h, 882
wpa_event_type
wpa_supplicant.h, 916
WPA_GET_BE24
common.h, 155
WPA_GET_BE32
common.h, 155
wpa_global, 72
WPA_GROUP_HANDSHAKE
defs.h, 284
wpa_gui-qt4/eventhistory.h, 886
wpa_gui-qt4/networkconfig.h, 887
wpa_gui-qt4/scanresults.h, 888
wpa_gui-qt4/userdatarequest.h, 889
wpa_gui-qt4/wpagui.h, 890
wpa_gui-qt4/wpamsg.h, 891
wpa_hexdump
common.c, 147
common.h, 157
wpa_hexdump_ascii
common.c, 147
common.h, 158
wpa_hexdump_ascii_key
common.c, 147
common.h, 158
wpa_hexdump_key
common.c, 148
common.h, 158
wpa_i.h, 892
STRUCT_PACKED, 892
wpa_ie
wpa_driver_associate_params, 49
WPA_INACTIVE
defs.h, 284
wpa_interface, 73
bridge_ifname, 73
confname, 73
ctrl_interface, 73
driver_param, 74
wpa_msg
common.h, 159
wpa_msg_register_cb
common.c, 148
common.h, 159
wpa_params, 75
pid_file, 75
wait_for_interface, 76
wpa_debug_show_keys, 76
wpa_parse_wpa_ie
wpa.c, 851
wpa.h, 864
wpa_passphrase.c, 893
wpa_printf
common.c, 148

common.h, 159
wpa_ptk, 77
WPA_PUT_BE16
common.h, 155
WPA_PUT_BE24
common.h, 155
WPA_PUT_BE32
common.h, 156
WPA_PUT_BE64
common.h, 156
WPA_PUT_LE16
common.h, 156
wpa_scan_result, 78
WPA_SCANNING
defs.h, 284
wpa_sm, 79
wpa_sm_aborted_cached
wpa.c, 851
wpa.h, 864
wpa_sm_deinit
wpa.c, 851
wpa.h, 865
wpa_sm_get_mib
wpa.c, 852
wpa.h, 865
wpa_sm_get_param
wpa.c, 852
wpa.h, 865
wpa_sm_get_status
wpa.c, 852
wpa.h, 866
wpa_sm_init
wpa.c, 853
wpa.h, 866
wpa_sm_key_request
wpa.c, 853
wpa.h, 866
wpa_sm_notify_assoc
wpa.c, 854
wpa.h, 867
wpa_sm_notify_disassoc
wpa.c, 854
wpa.h, 867
wpa_sm_parse_own_wpa_ie
wpa.c, 855
wpa.h, 868
wpa_sm_rx_eapol
wpa.c, 855
wpa.h, 868
wpa_sm_set_ap_rsn_ie
wpa.c, 856
wpa.h, 869
wpa_sm_set_ap_wpa_ie
wpa.c, 856
wpa.h, 869
wpa_sm_set_assoc_wpa_ie
wpa.c, 857
wpa.h, 870
wpa_sm_set_assoc_wpa_ie_default
wpa.c, 857
wpa.h, 870
wpa_sm_set_config
wpa.c, 857
wpa.h, 871
wpa_sm_set_eapol
wpa.c, 858
wpa.h, 871
wpa_sm_set_fast_reauth
wpa.c, 858
wpa.h, 871
wpa_sm_set_ifname
wpa.c, 858
wpa.h, 871
wpa_sm_set_own_addr
wpa.c, 858
wpa.h, 872
wpa_sm_set_param
wpa.c, 859
wpa.h, 872
wpa_sm_set_pmk
wpa.c, 859
wpa.h, 872
wpa_sm_set_pmk_from_pmksa
wpa.c, 859
wpa.h, 872
wpa_sm_set_scard_ctx
wpa.c, 859
wpa.h, 873
wpa_sm_stkstart
wpa.c, 860
wpa.h, 873
wpa_snprintf_hex
common.c, 149
common.h, 160
wpa_snprintf_hex_uppercase
common.c, 149
common.h, 160
wpa_ssid, 81
altsubject_match2, 86
anonymous_identity, 86
auth_alg, 86
bssid, 86
ca_cert, 86
ca_cert2, 86
ca_path, 87
ca_path2, 87
client_cert, 87
client_cert2, 87

dh_file, 88
dh_file2, 88
disabled, 88
eap_methods, 88
eap_workaround, 88
eappsk_len, 89
engine, 89
engine_id, 89
fragment_size, 89
id, 89
id_str, 89
key_id, 90
key_mgmt, 90
leap, 90
mode, 90
mschapv2_retry, 90
new_password, 90
next, 90
non_leap, 91
otp, 91
pac_file, 91
passphrase, 91
pcsc, 91
peerkey, 91
pending_req_identity, 92
pending_req_new_password, 92
pending_req_otp, 92
pending_req_passphrase, 92
pending_req_password, 92
pending_req_pin, 92
phase1, 93
phase2, 93
pin, 93
pnext, 93
priority, 94
private_key, 94
private_key2, 94
private_key2_passwd, 94
private_key_passwd, 95
proactive_key_caching, 95
scan_ssid, 95
ssid, 95
subject_match, 95
subject_match2, 96
wpa_ssid_txt
common.c, 149
common.h, 160
wpa_states
defs.h, 284
wpa_supplicant, 97
wpa_supplicant.c, 895
wpa_blacklist_add, 899
wpa_blacklist_clear, 899
wpa_blacklist_get, 900
wpa_clear_keys, 900
wpa_supplicant_add_iface, 900
wpa_supplicant_associate, 901
wpa_supplicant_cancel_auth_timeout, 902
wpa_supplicant_cancel_scan, 902
wpa_supplicant_deauthenticate, 903
wpa_supplicant_deinit, 903
wpa_supplicant_disassociate, 904
wpa_supplicant_driver_init, 904
wpa_supplicant_full_license1, 912
wpa_supplicant_full_license2, 913
wpa_supplicant_full_license3, 913
wpa_supplicant_full_license4, 913
wpa_supplicant_full_license5, 913
wpa_supplicant_get_iface, 905
wpa_supplicant_get_scan_results, 905
wpa_supplicant_get_ssid, 906
wpa_supplicant_get_state, 906
wpa_supplicant_init, 906
wpa_supplicant_initiate_eapol, 907
wpa_supplicant_license, 914
wpa_supplicant_reload_configuration, 907
wpa_supplicant_remove_iface, 908
wpa_supplicant_req_auth_timeout, 908
wpa_supplicant_req_scan, 909
wpa_supplicant_run, 909
wpa_supplicant_rx_eapol, 910
wpa_supplicant_set_non_wpa_policy, 910
wpa_supplicant_set_state, 911
wpa_supplicant_set_suites, 911
wpa_supplicant_state_txt, 912
wpa_supplicant_version, 914
wpa_supplicant.h
EVENT_ASSOC, 916
EVENT_ASSOCINFO, 917
EVENT_DISASSOC, 916
EVENT_INTERFACE_STATUS, 917
EVENT_MICHAEL_MIC_FAILURE, 917
EVENT_PMKID_CANDIDATE, 917
EVENT_SCAN_RESULTS, 917
EVENT_STKSTART, 917
wpa_supplicant.h, 915
wpa_event_type, 916
wpa_supplicant_event, 918
wpa_supplicant_rx_eapol, 918
wpa_supplicant_add_iface
wpa_supplicant.c, 900
wpa_supplicant_i.h, 923
wpa_supplicant_associate
wpa_supplicant.c, 901
wpa_supplicant_i.h, 923
wpa_supplicant_cancel_auth_timeout
wpa_supplicant.c, 902
wpa_supplicant_i.h, 924

wpa_supplicant_cancel_scan
wpa_supplicant.c, 902
wpa_supplicant_i.h, 925
wpa_supplicant_ctrl_iface_deinit
ctrl_iface.h, 226
ctrl_iface_named_pipe.c, 254
ctrl_iface_udp.c, 258
ctrl_iface_unix.c, 262
wpa_supplicant_ctrl_iface_init
ctrl_iface.h, 226
ctrl_iface_named_pipe.c, 254
ctrl_iface_udp.c, 258
ctrl_iface_unix.c, 262
wpa_supplicant_ctrl_iface_process
ctrl_iface.c, 222
ctrl_iface.h, 227
wpa_supplicant_ctrl_iface_wait
ctrl_iface.h, 228
ctrl_iface_named_pipe.c, 255
ctrl_iface_udp.c, 259
ctrl_iface_unix.c, 263
wpa_supplicant_dbus_ctrl_iface_deinit
ctrl_iface_dbus.c, 233
wpa_supplicant_dbus_ctrl_iface_init
ctrl_iface_dbus.c, 233
wpa_supplicant_dbus_next_objid
ctrl_iface_dbus.c, 234
wpa_supplicant_dbus_notify_scan_results
ctrl_iface_dbus.c, 234
wpa_supplicant_dbus_notify_state_change
ctrl_iface_dbus.c, 235
wpa_supplicant_deauthenticate
wpa_supplicant.c, 903
wpa_supplicant_i.h, 925
wpa_supplicant_deinit
wpa_supplicant.c, 903
wpa_supplicant_i.h, 926
wpa_supplicant_disassociate
wpa_supplicant.c, 904
wpa_supplicant_i.h, 926
wpa_supplicant_driver_init
wpa_supplicant.c, 904
wpa_supplicant_i.h, 927
wpa_supplicant_event
events.c, 530
wpa_supplicant.h, 918
wpa_supplicant_full_license1
wpa_supplicant.c, 912
wpa_supplicant_full_license2
wpa_supplicant.c, 913
wpa_supplicant_full_license3
wpa_supplicant.c, 913
wpa_supplicant_full_license4
wpa_supplicant.c, 913
wpa_supplicant_full_license5
wpa_supplicant.c, 913
wpa_supplicant_get_dbus_path
ctrl_iface_dbus.c, 235
wpa_supplicant_get_iface
wpa_supplicant.c, 905
wpa_supplicant_i.h, 928
wpa_supplicant_get_iface_by_dbus_path
ctrl_iface_dbus.c, 235
wpa_supplicant_get_scan_results
wpa_supplicant.c, 905
wpa_supplicant_i.h, 928
wpa_supplicant_get_ssid
wpa_supplicant.c, 906
wpa_supplicant_i.h, 928
wpa_supplicant_get_state
wpa_supplicant.c, 906
wpa_supplicant_global_ctrl_iface_deinit
ctrl_iface.h, 228
ctrl_iface_named_pipe.c, 255
ctrl_iface_udp.c, 259
ctrl_iface_unix.c, 263
wpa_supplicant_global_ctrl_iface_init
ctrl_iface.h, 229
ctrl_iface_named_pipe.c, 255
ctrl_iface_udp.c, 260
ctrl_iface_unix.c, 264
wpa_supplicant_global_ctrl_iface_process
ctrl_iface.c, 223
ctrl_iface.h, 229
wpa_supplicant_i.h, 919
wpa_blacklist_add, 922
wpa_blacklist_clear, 922
wpa_blacklist_get, 922
wpa_clear_keys, 923
wpa_supplicant_add_iface, 923
wpa_supplicant_associate, 923
wpa_supplicant_cancel_auth_timeout, 924
wpa_supplicant_cancel_scan, 925
wpa_supplicant_deauthenticate, 925
wpa_supplicant_deinit, 926
wpa_supplicant_disassociate, 926
wpa_supplicant_driver_init, 927
wpa_supplicant_get_iface, 928
wpa_supplicant_get_scan_results, 928
wpa_supplicant_get_ssid, 928
wpa_supplicant_init, 929
wpa_supplicant_initiate_eapol, 929
wpa_supplicant_reload_configuration, 930
wpa_supplicant_remove_iface, 931
wpa_supplicant_req_auth_timeout, 931
wpa_supplicant_req_scan, 932
wpa_supplicant_run, 932
wpa_supplicant_scard_init, 933

wpa_supplicant_set_non_wpa_policy, 933
wpa_supplicant_set_state, 934
wpa_supplicant_set_suites, 934
wpa_supplicant_state_txt, 935
wpa_supplicant_init
wpa_supplicant.c, 906
wpa_supplicant_i.h, 929
wpa_supplicant_initiate_eapol
wpa_supplicant.c, 907
wpa_supplicant_i.h, 929
wpa_supplicant_license
wpa_supplicant.c, 914
wpa_supplicant_reload_configuration
wpa_supplicant.c, 907
wpa_supplicant_i.h, 930
wpa_supplicant_remove_iface
wpa_supplicant.c, 908
wpa_supplicant_i.h, 931
wpa_supplicant_req_auth_timeout
wpa_supplicant.c, 908
wpa_supplicant_i.h, 931
wpa_supplicant_req_scan
wpa_supplicant.c, 909
wpa_supplicant_i.h, 932
wpa_supplicant_run
wpa_supplicant.c, 909
wpa_supplicant_i.h, 932
wpa_supplicant_rx_eapol
wpa_supplicant.c, 910
wpa_supplicant.h, 918
wpa_supplicant_scard_init
events.c, 530
wpa_supplicant_i.h, 933
wpa_supplicant_set_dbus_path
ctrl_iface_dbus.c, 236
wpa_supplicant_set_non_wpa_policy
wpa_supplicant.c, 910
wpa_supplicant_i.h, 933
wpa_supplicant_set_state
wpa_supplicant.c, 911
wpa_supplicant_i.h, 934
wpa_supplicant_set_suites
wpa_supplicant.c, 911
wpa_supplicant_i.h, 934
wpa_supplicant_state_txt
wpa_supplicant.c, 912
wpa_supplicant_i.h, 935
wpa_supplicant_version
wpa_supplicant.c, 914
wpas_dbus_bssid_properties
ctrl_iface_dbus_handlers.c, 242
wpas_dbus_decompose_object_path
ctrl_iface_dbus.c, 236
wpas_dbus_global_add_interface
ctrl_iface_dbus_handlers.c, 242
wpas_dbus_global_get_interface
ctrl_iface_dbus_handlers.c, 243
wpas_dbus_global_remove_interface
ctrl_iface_dbus_handlers.c, 244
wpas_dbus_iface_add_network
ctrl_iface_dbus_handlers.c, 244
wpas_dbus_iface_capabilities
ctrl_iface_dbus_handlers.c, 245
wpas_dbus_iface_disable_network
ctrl_iface_dbus_handlers.c, 245
wpas_dbus_iface_disconnect
ctrl_iface_dbus_handlers.c, 246
wpas_dbus_iface_enable_network
ctrl_iface_dbus_handlers.c, 246
wpas_dbus_iface_get_state
ctrl_iface_dbus_handlers.c, 247
wpas_dbus_iface_remove_network
ctrl_iface_dbus_handlers.c, 247
wpas_dbus_iface_scan
ctrl_iface_dbus_handlers.c, 248
wpas_dbus_iface_scan_results
ctrl_iface_dbus_handlers.c, 248
wpas_dbus_iface_select_network
ctrl_iface_dbus_handlers.c, 249
wpas_dbus_iface_set_ap_scan
ctrl_iface_dbus_handlers.c, 250
wpas_dbus_iface_set_network
ctrl_iface_dbus_handlers.c, 250
wpas_dbus_new_invalid_iface_error
ctrl_iface_dbus.c, 236
wpas_dbus_new_invalid_network_error
ctrl_iface_dbus.c, 236
wpas_dbus_register_iface
ctrl_iface_dbus.c, 237
wpas_dbus_unregister_iface
ctrl_iface_dbus.c, 237

x509v3.c, 936
x509v3.h, 938