

wpa_supplicant Reference Manual
0.4.x

Generated by Doxygen 1.4.2

Sat May 6 21:13:28 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	9
4.1	wpa_supplicant Related Pages	9
5	wpa_supplicant Data Structure Documentation	11
5.1	eap_config Struct Reference	11
5.2	eap_method Struct Reference	13
5.3	eap_method_ret Struct Reference	18
5.4	eap_sm Struct Reference	19
5.5	eapol_callbacks Struct Reference	21
5.6	eapol_config Struct Reference	24
5.7	eapol_ctx Struct Reference	26
5.8	eapol_sm Struct Reference	30
5.9	rsn_pmksa_cache Struct Reference	33
5.10	tls_connection_params Struct Reference	34
5.11	wpa_config Struct Reference	36
5.12	wpa_config_blob Struct Reference	41
5.13	wpa_ctrl Struct Reference	42
5.14	wpa_ctrl_dst Struct Reference	43
5.15	wpa_driver_associate_params Struct Reference	44
5.16	wpa_driver_capa Struct Reference	46
5.17	wpa_driver_ops Struct Reference	47
5.18	wpa_event_data Union Reference	57

5.19	wpa_event_data::assoc_info Struct Reference	59
5.20	wpa_event_data::interface_status Struct Reference	61
5.21	wpa_event_data::michael_mic_failure Struct Reference	62
5.22	wpa_event_data::pmkid_candidate Struct Reference	63
5.23	wpa_global Struct Reference	64
5.24	wpa_interface Struct Reference	65
5.25	wpa_params Struct Reference	67
5.26	wpa_ptk Struct Reference	69
5.27	wpa_scan_result Struct Reference	70
5.28	wpa_sm Struct Reference	71
5.29	wpa_ssid Struct Reference	73
5.30	wpa_supPLICANT Struct Reference	88
6	wpa_supPLICANT File Documentation	91
6.1	aes.c File Reference	91
6.2	aes_wrap.c File Reference	96
6.3	aes_wrap.h File Reference	103
6.4	base64.c File Reference	110
6.5	base64.h File Reference	112
6.6	common.c File Reference	114
6.7	common.h File Reference	119
6.8	config.c File Reference	126
6.9	config.h File Reference	134
6.10	config_file.c File Reference	142
6.11	config_ssid.h File Reference	145
6.12	crypto.c File Reference	148
6.13	crypto.h File Reference	150
6.14	crypto_gnutls.c File Reference	155
6.15	ctrl_iface.c File Reference	157
6.16	ctrl_iface.h File Reference	162
6.17	defs.h File Reference	163
6.18	driver.h File Reference	165
6.19	driver_atmel.c File Reference	167
6.20	driver_broadcom.c File Reference	169
6.21	driver_bsd.c File Reference	172
6.22	driver_hostap.c File Reference	175
6.23	driver_hostap.h File Reference	177

6.24 driver_ipw.c File Reference	179
6.25 driver_madwifi.c File Reference	182
6.26 driver_ndis.c File Reference	185
6.27 driver_ndis.h File Reference	189
6.28 driver_ndis_.c File Reference	190
6.29 driver_ndiswrapper.c File Reference	192
6.30 driver_prism54.c File Reference	195
6.31 driver_test.c File Reference	197
6.32 driver_wext.c File Reference	199
6.33 driver_wext.h File Reference	208
6.34 driver_wired.c File Reference	215
6.35 drivers.c File Reference	217
6.36 eap.c File Reference	218
6.37 eap.h File Reference	233
6.38 eap_aka.c File Reference	247
6.39 eap_defs.h File Reference	249
6.40 eap_fast.c File Reference	251
6.41 eap_gtc.c File Reference	253
6.42 eap_i.h File Reference	255
6.43 eap_leap.c File Reference	258
6.44 eap_md5.c File Reference	260
6.45 eap_mschapv2.c File Reference	262
6.46 eap_otp.c File Reference	264
6.47 eap_pax.c File Reference	266
6.48 eap_pax_common.c File Reference	268
6.49 eap_pax_common.h File Reference	271
6.50 eap_peap.c File Reference	275
6.51 eap_psk.c File Reference	277
6.52 eap_psk_common.c File Reference	279
6.53 eap_psk_common.h File Reference	280
6.54 eap_sim.c File Reference	281
6.55 eap_sim_common.c File Reference	283
6.56 eap_sim_common.h File Reference	285
6.57 eap_tls.c File Reference	287
6.58 eap_tls_common.c File Reference	289
6.59 eap_tls_common.h File Reference	292

6.60	eap_tlv.c File Reference	294
6.61	eap_tlv.h File Reference	297
6.62	eap_ttls.c File Reference	300
6.63	eap_ttls.h File Reference	302
6.64	eapol_sm.c File Reference	304
6.65	eapol_sm.h File Reference	318
6.66	eapol_test.c File Reference	330
6.67	eloop.c File Reference	333
6.68	eloop.h File Reference	338
6.69	events.c File Reference	343
6.70	l2_packet.h File Reference	346
6.71	l2_packet_freebsd.c File Reference	351
6.72	l2_packet_linux.c File Reference	356
6.73	l2_packet_pcap.c File Reference	361
6.74	main.c File Reference	365
6.75	md5.c File Reference	367
6.76	md5.h File Reference	370
6.77	ms_funcs.c File Reference	372
6.78	ms_funcs.h File Reference	378
6.79	pcsc_funcs.c File Reference	384
6.80	pcsc_funcs.h File Reference	389
6.81	preauth.c File Reference	391
6.82	preauth.h File Reference	401
6.83	preauth_test.c File Reference	409
6.84	priv_netlink.h File Reference	412
6.85	radius.c File Reference	414
6.86	radius_client.c File Reference	417
6.87	rc4.c File Reference	420
6.88	rc4.h File Reference	422
6.89	sha1.c File Reference	424
6.90	sha1.h File Reference	431
6.91	tls.h File Reference	436
6.92	tls_gnutls.c File Reference	448
6.93	tls_none.c File Reference	460
6.94	tls_openssl.c File Reference	462
6.95	tls_schannel.c File Reference	474

6.96	win_if_list.c File Reference	486
6.97	wpa.c File Reference	487
6.98	wpa.h File Reference	500
6.99	wpa_cli.c File Reference	512
6.100	wpa_ctrl.c File Reference	514
6.101	wpa_ctrl.h File Reference	519
6.102	wpa_i.h File Reference	525
6.103	wpa_passphrase.c File Reference	526
6.104	wpa_supplicant.c File Reference	527
6.105	wpa_supplicant.h File Reference	546
6.106	wpa_supplicant_i.h File Reference	552
7	wpa_supplicant Page Documentation	569
7.1	Structure of the source code	569
7.2	Control interface	573
7.3	Driver wrapper implementation (driver.h, drivers.c)	580
7.4	EAP peer implementation	583
7.5	Porting to different target boards and operating systems	584
7.6	Testing and development tools	586

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-04.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++.

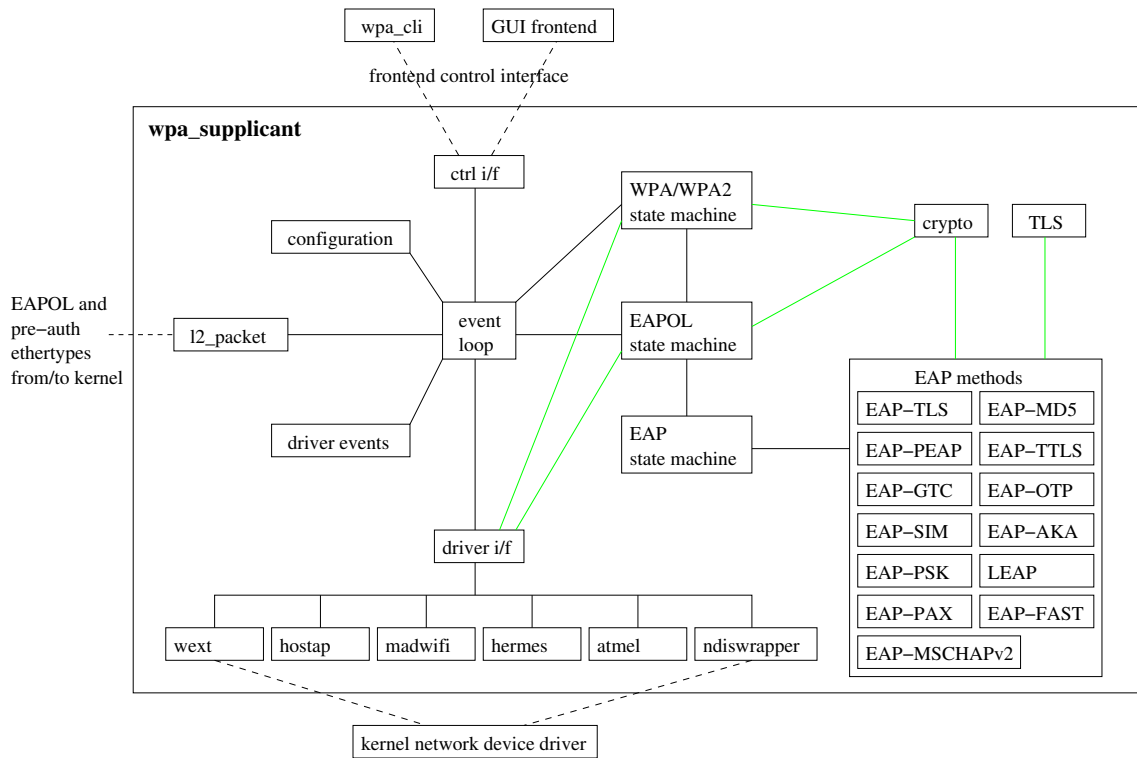


Figure 1.1: wpa_supplicant modules

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)	11
eap_method (EAP method interface)	13
eap_method_ret (EAP return values from struct eap_method::process())	18
eap_sm (EAP state machine data)	19
eapol_callbacks (Callback functions from EAP to lower layer)	21
eapol_config (Per network configuration for EAPOL state machines)	24
eapol_ctx (Global (for all networks) EAPOL state machine context)	26
eapol_sm (Internal data for EAPOL state machines)	30
rsn_pmksa_cache (PMKSA cache entry)	33
tls_connection_params (Parameters for TLS connection)	34
wpa_config (Wpa_supplicant configuration data)	36
wpa_config_blob (Named configuration blob)	41
wpa_ctrl (Internal structure for control interface library)	42
wpa_ctrl_dst (Internal data structure of control interface monitors)	43
wpa_driver_associate_params (Association parameters)	44
wpa_driver_capa (Driver capability information)	46
wpa_driver_ops (Driver interface API definition)	47
wpa_event_data	57
wpa_event_data::assoc_info (Data for EVENT_ASSOC and EVENT_ASSOCINFO events)	59
wpa_event_data::interface_status (Data for EVENT_INTERFACE_STATUS)	61
wpa_event_data::michael_mic_failure (Data for EVENT_MICHAEL_MIC_FAILURE)	62
wpa_event_data::pmkid_candidate (Data for EVENT_PMKID_CANDIDATE)	63
wpa_global (Internal, global data for all wpa_supplicant interfaces)	64
wpa_interface (Parameters for wpa_supplicant_add_iface())	65
wpa_params (Parameters for wpa_supplicant_init())	67
wpa_ptk (WPA Pairwise Transient Key)	69
wpa_scan_result (Scan results)	70
wpa_sm (Internal WPA state machine data)	71
wpa_ssid (Network configuration data)	73
wpa_supplicant (Internal data for wpa_supplicant interface)	88

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)	91
aes_wrap.c (AES-based functions)	96
aes_wrap.h (AES-based functions)	103
base64.c (Base64 encoding/decoding (RFC1341))	110
base64.h (Base64 encoding/decoding (RFC1341))	112
common.c (Wpa_supplicant/hostapd / common helper functions, etc)	114
common.h (Wpa_supplicant/hostapd / common helper functions, etc)	119
config.c (WPA Supplicant / Configuration parser and common functions)	126
config.h (WPA Supplicant / Configuration file structures)	134
config_file.c (WPA Supplicant / Configuration backend: text file)	142
config_ssid.h (WPA Supplicant / Network configuration structures)	145
config_types.h	??
crypto.c (WPA Supplicant / wrapper functions for libcrypto)	148
crypto.h (WPA Supplicant / wrapper functions for crypto libraries)	150
crypto_gnutls.c (WPA Supplicant / wrapper functions for libgrypt)	155
ctrl_iface.c (WPA Supplicant / UNIX domain and UDP socket -based control interface)	157
ctrl_iface.h (WPA Supplicant / UNIX domain socket -based control interface)	162
defs.h (WPA Supplicant - Common definitions)	163
driver.h (WPA Supplicant - driver interface definition)	165
driver_atmel.c (WPA Supplicant - Driver interaction with Atmel Wireless LAN drivers)	167
driver_broadcom.c (WPA Supplicant - driver interaction with Broadcom wlo driver)	169
driver_bsd.c (WPA Supplicant - driver interaction with BSD net80211 layer)	172
driver_hostap.c (WPA Supplicant - driver interaction with Linux Host AP driver)	175
driver_hostap.h (WPA Supplicant - driver interaction with Linux Host AP driver)	177
driver_ipw.c (WPA Supplicant - driver interaction with Linux ipw2100/2200 drivers)	179
driver_madwifi.c (WPA Supplicant - driver interaction with MADWIFI 802.11 driver)	182
driver_ndis.c (WPA Supplicant - Windows/NDIS driver interface)	185
driver_ndis.h (WPA Supplicant - Windows/NDIS driver interface)	189
driver_ndis_c (WPA Supplicant - Windows/NDIS driver interface - event processing)	190
driver_ndiswrapper.c (WPA Supplicant - driver interaction with Linux ndiswrapper)	192
driver_prism54.c (WPA Supplicant - driver interaction with Linux Prism54.org driver)	195
driver_test.c (WPA Supplicant - testing driver interface)	197
driver_wext.c (WPA Supplicant - driver interaction with generic Linux Wireless Extensions)	199

driver_wext.h (WPA Supplicant - driver_wext exported functions)	208
driver_wired.c (WPA Supplicant - wired Ethernet driver interface)	215
drivers.c (WPA Supplicant / driver interface list)	217
eap.c (WPA Supplicant / EAP state machines (RFC 4137))	218
eap.h (WPA Supplicant / EAP state machine functions (RFC 4137))	233
eap_aka.c (WPA Supplicant / EAP-AKA (draft-arkko-pppext-eap-aka-12.txt))	247
eap_defs.h (WPA Supplicant/hostapd / Shared EAP definitions)	249
eap_fast.c (WPA Supplicant / EAP-FAST (draft-cam-winget-eap-fast-00.txt))	251
eap_gtc.c (WPA Supplicant / EAP-GTC (RFC 2284))	253
eap_i.h (WPA Supplicant / EAP state machines internal structures (RFC 4137))	255
eap_leap.c (WPA Supplicant / EAP-LEAP)	258
eap_md5.c (WPA Supplicant / EAP-MD5)	260
eap_mschapv2.c (WPA Supplicant / EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.txt))	262
eap_otp.c (WPA Supplicant / EAP-OTP (RFC 3748))	264
eap_pax.c (WPA Supplicant / EAP-PAX (draft-clancy-eap-pax-04.txt))	266
eap_pax_common.c (WPA Supplicant / EAP-PAX shared routines)	268
eap_pax_common.h (WPA Supplicant / EAP-PAX shared routines)	271
eap_peap.c (WPA Supplicant / EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-07.txt))	275
eap_psk.c (WPA Supplicant / EAP-PSK (draft-bersani-eap-psk-09.txt))	277
eap_psk_common.c (WPA Supplicant / EAP-PSK shared routines)	279
eap_psk_common.h (WPA Supplicant / EAP-PSK shared routines)	280
eap_sim.c (WPA Supplicant / EAP-SIM (draft-haverinen-pppext-eap-sim-13.txt))	281
eap_sim_common.c (WPA Supplicant / EAP-SIM/AKA shared routines)	283
eap_sim_common.h (WPA Supplicant / EAP-SIM/AKA shared routines)	285
eap_tls.c (WPA Supplicant / EAP-TLS (RFC 2716))	287
eap_tls_common.c (WPA Supplicant / EAP-TLS/PEAP/TTLS/FAST common functions)	289
eap_tls_common.h (WPA Supplicant / EAP-TLS/PEAP/TTLS/FAST common functions)	292
eap_tlv.c (WPA Supplicant / EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt))	294
eap_tlv.h (WPA Supplicant / EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt))	297
eap_ttls.c (WPA Supplicant / EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt))	300
eap_ttls.h (WPA Supplicant / EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt))	302
eapol_sm.c (WPA Supplicant / EAPOL state machines)	304
eapol_sm.h (WPA Supplicant / EAPOL state machines)	318
eapol_test.c (WPA Supplicant - test code)	330
elooop.c (Event loop based on select() loop)	333
elooop.h (Event loop)	338
events.c (WPA Supplicant - Driver event processing)	343
hostapd.h	??
l2_packet.h (WPA Supplicant - Layer2 packet interface definition)	346
l2_packet_freebsd.c (WPA Supplicant - Layer2 packet handling with FreeBSD)	351
l2_packet_linux.c (WPA Supplicant - Layer2 packet handling with Linux packet sockets)	356
l2_packet_pcap.c (WPA Supplicant - Layer2 packet handling with libpcap/libdnet and WinPcap)	361
main.c (WPA Supplicant / main() function for UNIX like OSes and MinGW)	365
md5.c (MD5 hash implementation and interface functions)	367
md5.h (MD5 hash implementation and interface functions)	370
ms_funcs.c (WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759)	372
ms_funcs.h (WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759)	378
pcsc_funcs.c (WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM)	384
pcsc_funcs.h (WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM)	389
preauth.c (WPA Supplicant - RSN pre-authentication and PMKSA caching)	391
preauth.h (Wpa_supplicant - WPA2/RSN pre-authentication functions)	401
preauth_test.c (WPA Supplicant - test code for pre-authentication)	409
priv_netlink.h (Wpa_supplicant - Private copy of Linux netlink/rtnetlink definitions)	412

radius.c (Host AP (software wireless LAN access point) user space daemon for Host AP kernel driver / RADIUS client)	414
radius.h	??
radius_client.c (Host AP (software wireless LAN access point) user space daemon for Host AP kernel driver / RADIUS client)	417
radius_client.h	??
rc4.c (RC4 stream cipher)	420
rc4.h (RC4 stream cipher)	422
sha1.c (SHA1 hash implementation and interface functions)	424
sha1.h (SHA1 hash implementation and interface functions)	431
tls.h (WPA Supplicant / SSL/TLS interface definition)	436
tls_gnutls.c (WPA Supplicant / SSL/TLS interface functions for openssl)	448
tls_none.c (WPA Supplicant / SSL/TLS interface functions for no TLS case)	460
tls_openssl.c (WPA Supplicant / SSL/TLS interface functions for openssl)	462
tls_schannel.c (WPA Supplicant / SSL/TLS interface functions for Microsoft Schannel)	474
version.h	??
win_if_list.c (Win_if_list - Display network interfaces with description (for Windows))	486
wireless_copy.h	??
wpa.c (WPA Supplicant - WPA state machine and EAPOL-Key processing)	487
wpa.h (Wpa_supplicant - WPA definitions)	500
wpa_cli.c (WPA Supplicant - command line interface for wpa_supplicant daemon)	512
wpa_ctrl.c (Wpa_supplicant/hostapd control interface library)	514
wpa_ctrl.h (Wpa_supplicant/hostapd control interface library)	519
wpa_i.h (Wpa_supplicant - Internal WPA state machine definitions)	525
wpa_passphrase.c (WPA Supplicant - ASCII passphrase to WPA PSK tool)	526
wpa_supplicant.c (WPA Supplicant)	527
wpa_supplicant.h (Wpa_supplicant - Exported functions for wpa_supplicant modules)	546
wpa_supplicant_i.h (Wpa_supplicant - Internal definitions)	552
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	??
wpa_gui-qt4/ eventhistory.ui.h	??
wpa_gui-qt4/ networkconfig.ui.h	??
wpa_gui-qt4/ scanresults.ui.h	??
wpa_gui-qt4/ userdatarequest.ui.h	??
wpa_gui-qt4/ wpaui.ui.h	??
wpa_gui-qt4/ wpaui.ui.h	??
wpa_gui/ eventhistory.ui.h	??
wpa_gui/ networkconfig.ui.h	??
wpa_gui/ scanresults.ui.h	??
wpa_gui/ userdatarequest.ui.h	??
wpa_gui/ wpaui.ui.h	??
wpa_gui/ wpaui.ui.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	569
Control interface	573
Driver wrapper implementation (driver.h, drivers.c)	580
EAP peer implementation	583
Porting to different target boards and operating systems	584
Testing and development tools	586

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 * [openc_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 223 of file eap.h.

5.1.2 Field Documentation

5.1.2.1 const char* [eap_config::openc_engine_path](#)

OpenSC engine for OpenSSL engine support.

Usually, path to engine_openc.so.

Definition at line 230 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 237 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 244 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>
```

Data Fields

- 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.

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 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.

5.2.2.4 `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.

5.2.2.5 `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.

5.2.2.6 `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.

5.2.2.7 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 its method-specific state and return a pointer that will be used as the `priv` argument to other calls.

5.2.2.8 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.

5.2.2.9 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

5.2.2.10 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.

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\(\)](#).

This 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:

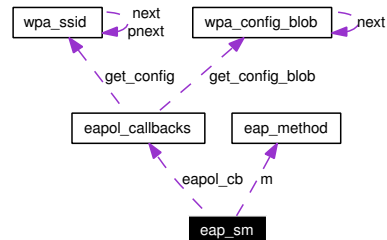
- [eap_i.h](#)

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**
- 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 **num_rounds**
- int **force_disabled**

5.4.1 Detailed Description

EAP state machine data.

Definition at line 234 of file [eap_i.h](#).

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

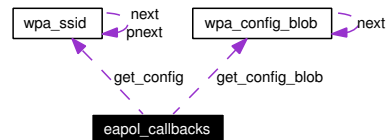
- [eap_i.h](#)

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.

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 142 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::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

5.5.2.7 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.

5.5.2.8 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:

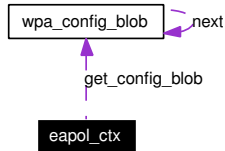
- [eapol_sm.h](#)

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 * [openc_engine_path](#)
Path to the OpenSSL engine for openc.
- 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

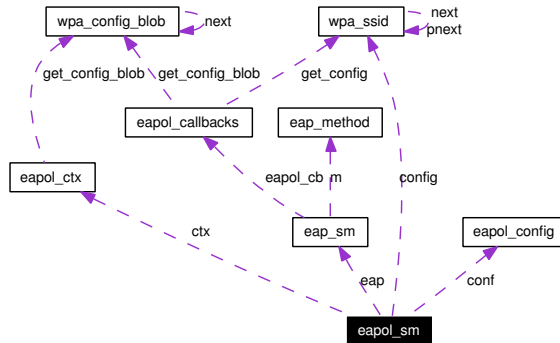
The documentation for this struct was generated from the following file:

- [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 **eapolSuccess**
- 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**
- [eapol_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 36 of file eapol_sm.c.

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

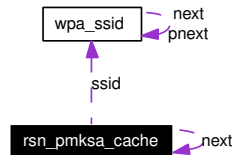
- [eapol_sm.c](#)

5.9 rsn_pmksa_cache Struct Reference

PMKSA cache entry.

```
#include <wpa_i.h>
```

Collaboration diagram for rsn_pmksa_cache:



Data Fields

- [rsn_pmksa_cache](#) * **next**
- u8 **pmkid** [PMKID_LEN]
- u8 **pmk** [PMK_LEN]
- size_t **pmk_len**
- time_t **expiration**
- time_t **reauth_time**
- int **akmp**
- u8 **aa** [ETH_ALEN]
- [wpa_ssid](#) * **ssid**
- int **opportunistic**

5.9.1 Detailed Description

PMKSA cache entry.

Definition at line 48 of file wpa_i.h.

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

- [wpa_i.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 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 connection parameters to be configured with `tls_connection_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 84 of file `tls.h`.

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

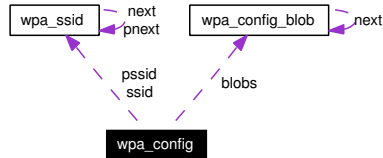
- [tls.h](#)

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](#)
Directory for UNIX domain sockets.
- [int fast_reauth](#)
EAP fast re-authentication (session resumption).
- [char * openssl_engine_path](#)
Path to the OpenSSL engine for openssl.
- [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 dot11RSNACfgPMKLifetime](#)

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 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 75 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 140 of file `config.h`.

5.11.2.2 char* `wpa_config::ctrl_interface`

Directory for UNIX domain sockets.

This variable is used to configure where the UNIX domain sockets for the control interface are created. If UDP-based `ctrl_iface` is used, this variable can be set to any string (i.e., NULL is not allowed).

Definition at line 151 of file `config.h`.

5.11.2.3 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 243 of file `config.h`.

5.11.2.4 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 252 of file `config.h`.

5.11.2.5 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 261 of file `config.h`.

5.11.2.6 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 234 of file `config.h`.

5.11.2.7 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 110 of file `config.h`.

5.11.2.8 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 195 of file `config.h`.

5.11.2.9 `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 97 of file `config.h`.

5.11.2.10 `char* wpa_config::opencsc_engine_path`

Path to the OpenSSL engine for opencsc.

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

Definition at line 204 of file `config.h`.

5.11.2.11 `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 213 of file `config.h`.

5.11.2.12 `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 (`opencsc-pkcs11.so`); if NULL, this module is not loaded.

Definition at line 223 of file `config.h`.

5.11.2.13 `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 82 of file `config.h`.

5.11.2.14 `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 overridden. 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 275 of file config.h.

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

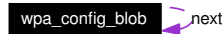
- [config.h](#)

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 39 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.

Data Fields

- int **s**
- sockaddr_un **local**
- sockaddr_un **dest**

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 44 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 monitors.

Collaboration diagram for wpa_ctrl_dst:



Data Fields

- `wpa_ctrl_dst * next`
- `CTRL_IFACE_SOCKET addr`
- `socklen_t addrlen`
- `int debug_level`
- `int errors`

5.14.1 Detailed Description

Internal data structure of control interface monitors.

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

Definition at line 60 of file `ctrl_iface.c`.

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

- `ctrl_iface.c`

5.15 wpa_driver_associate_params Struct Reference

Association parameters.

```
#include <driver.h>
```

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_**.

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 147 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.

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 181 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 185 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 `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.

5.17.2.11 `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.

5.17.2.12 `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.

5.17.2.13 `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.

5.17.2.14 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.

5.17.2.15 const char* [wpa_driver_ops::name](#)

Name of the driver interface

Definition at line 183 of file driver.h.

5.17.2.16 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.

5.17.2.17 `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.

5.17.2.18 `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()`.

5.17.2.19 `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.

5.17.2.20 `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

5.17.2.21 `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.

5.17.2.22 `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.

5.17.2.23 `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_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

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 trigger Michael MIC errors. This can be fixed by changing the order of MIC keys by swapping the 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.

5.17.2.24 `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).

5.17.2.25 `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).

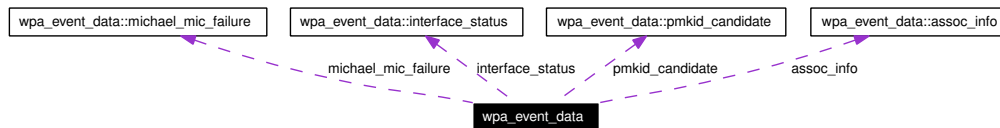
The documentation for this struct was generated from the following file:

- [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.

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.

5.18.1 Detailed Description

union `wpa_event_data` - Additional data for [wpa_supplicant_event\(\)](#) calls

Definition at line 139 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 149 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 207 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 162 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 182 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** [20]
- enum wpa_event_data::interface_status:: { ... } **ievent**

5.20.1 Detailed Description

Data for EVENT_INTERFACE_STATUS.

Definition at line 227 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 219 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 238 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 240 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 242 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 244 of file wpa_supplicant.h.

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

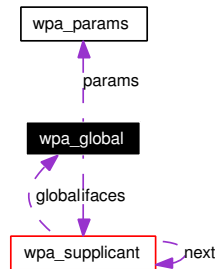
- [wpa_supplicant.h](#)

5.23 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
- int ctrl_sock

5.23.1 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 160 of file wpa_supplicant_i.h.

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

- [wpa_supplicant_i.h](#)

5.24 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 * [iface](#)
Interface name.

5.24.1 Detailed Description

Parameters for [wpa_supplicant_add_iface\(\)](#).

Definition at line 36 of file `wpa_supplicant_i.h`.

5.24.2 Field Documentation

5.24.2.1 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 45 of file `wpa_supplicant_i.h`.

5.24.2.2 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 57 of file `wpa_supplicant_i.h`.

5.24.2.3 `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 75 of file `wpa_supplicant_i.h`.

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

- [wpa_supplicant_i.h](#)

5.25 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.

5.25.1 Detailed Description

Parameters for [wpa_supplicant_init\(\)](#).

Definition at line 88 of file [wpa_supplicant_i.h](#).

5.25.2 Field Documentation

5.25.2.1 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 120 of file [wpa_supplicant_i.h](#).

5.25.2.2 `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 105 of file `wpa_supplicant_i.h`.

5.25.2.3 `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 138 of file `wpa_supplicant_i.h`.

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

- [wpa_supplicant_i.h](#)

5.26 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**

5.26.1 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](#)

5.27 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**

5.27.1 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](#)

- u8 **ssid** [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**
- 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**

5.28.1 Detailed Description

Internal WPA state machine data.

Definition at line 66 of file wpa_i.h.

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

- [wpa_i.h](#)

5.29 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 pre-shared key.
- size_t [eappsk_len](#)
EAP-PSK pre-shared key length.
- u8 * [nai](#)
User NAI (for EAP-PSK/PAX).
- 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](#)
Constraint for server certificate alt. subject.
- 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.
- u8 * [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 `epol_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 (Infrastructure/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.

5.29.1 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 61 of file [config_ssid.h](#).

5.29.2 Field Documentation

5.29.2.1 u8* [wpa_ssid::altsubject_match](#)

Constraint for server certificate alt. subject.

This substring is matched against the alternative subject name of the authentication server certificate. If this string is set, the server certificate is only accepted if it contains this string in an alternative subject name extension.

altSubjectName string is in following format: TYPE:VALUE

Example: DNS:server.example.com

Following types are supported: EMAIL, DNS, URI

Definition at line 405 of file config_ssids.h.

5.29.2.2 u8* wpa_ssids::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 508 of file config_ssids.h.

5.29.2.3 u8* wpa_ssids::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 233 of file config_ssids.h.

5.29.2.4 int wpa_ssids::auth_alg

Bitfield of allowed authentication algorithms.

WPA_AUTH_ALG_*

Definition at line 201 of file config_ssids.h.

5.29.2.5 u8 wpa_ssids::bssid[ETH_ALEN]

BSSID.

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

Definition at line 139 of file config_ssids.h.

5.29.2.6 u8* wpa_ssids::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"`.

Definition at line 297 of file `config_ssid.h`.

5.29.2.7 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 424 of file `config_ssid.h`.

5.29.2.8 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 309 of file `config_ssid.h`.

5.29.2.9 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 439 of file `config_ssid.h`.

5.29.2.10 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 324 of file `config_ssid.h`.

5.29.2.11 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 453 of file `config_ssid.h`.

5.29.2.12 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 375 of file `config_ssid.h`.

5.29.2.13 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 490 of file `config_ssid.h`.

5.29.2.14 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 839 of file `config_ssid.h`.

5.29.2.15 u8* wpa_ssid::eap_methods

Allowed EAP methods.

Zero (`EAP_TYPE_NONE`) terminated list of allowed EAP methods or NULL if all methods are accepted.

Definition at line 517 of file `config_ssid.h`.

5.29.2.16 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 774 of file `config_ssid.h`.

5.29.2.17 size_t wpa_ssid::eappsk_len

EAP-PSK pre-shared key length.

This field is always 16 for the current version of EAP-PSK.

Definition at line 253 of file `config_ssid.h`.

5.29.2.18 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 595 of file `config_ssid.h`.

5.29.2.19 char* wpa_ssid::engine_id

Engine ID for OpenSSL engine.

"opensec" 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 607 of file `config_ssid.h`.

5.29.2.20 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 90 of file `config_ssid.h`.

5.29.2.21 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 616 of file `config_ssid.h`.

5.29.2.22 int wpa_ssid::key_mgmt

Bitfield of allowed key management protocols.

WPA_KEY_MGMT_*

Definition at line 187 of file config_ssid.h.

5.29.2.23 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 750 of file config_ssid.h.

5.29.2.24 int wpa_ssid::mode

IEEE 802.11 operation mode (Infrastructure/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 804 of file config_ssid.h.

5.29.2.25 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 813 of file config_ssid.h.

5.29.2.26 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 823 of file config_ssid.h.

5.29.2.27 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 69 of file config_ssid.h.

5.29.2.28 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 760 of file config_ssid.h.

5.29.2.29 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 669 of file config_ssid.h.

5.29.2.30 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 787 of file config_ssid.h.

5.29.2.31 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 167 of file config_ssid.h.

5.29.2.32 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 574 of file config_ssid.h.

5.29.2.33 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 685 of file config_ssid.h.

5.29.2.34 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 715 of file config_ssid.h.

5.29.2.35 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 735 of file config_ssid.h.

5.29.2.36 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 725 of file config_ssid.h.

5.29.2.37 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 695 of file config_ssid.h.

5.29.2.38 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 705 of file config_ssid.h.

5.29.2.39 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 553 of file `config_ssid.h`.

5.29.2.40 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 562 of file `config_ssid.h`.

5.29.2.41 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 586 of file `config_ssid.h`.

5.29.2.42 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 79 of file `config_ssid.h`.

5.29.2.43 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 110 of file config_ssid.h.

5.29.2.44 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"

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

Definition at line 348 of file config_ssid.h.

5.29.2.45 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 467 of file config_ssid.h.

5.29.2.46 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 476 of file config_ssid.h.

5.29.2.47 u8* [wpa_ssid::private_key_passwd](#)

Password for private key file.

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

Definition at line 356 of file config_ssid.h.

5.29.2.48 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 660 of file config_ssid.h.

5.29.2.49 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 211 of file config_ssid.h.

5.29.2.50 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 124 of file config_ssid.h.

5.29.2.51 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 certificate 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 388 of file config_ssid.h.

5.29.2.52 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 499 of file config_ssid.h.

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

- [config_ssid.h](#)

- [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**

5.30.1 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 175 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.

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



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 856 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 858 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 1097 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 1103 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 1084 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 1072 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 1078 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 1059 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 894 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.

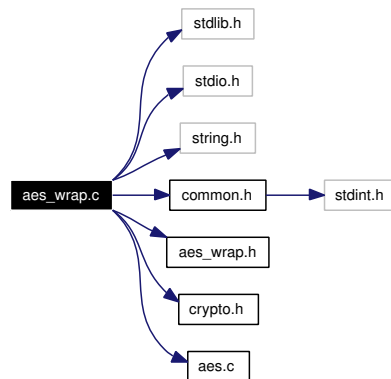
Definition at line 868 of file aes.c.

6.2 aes_wrap.c File Reference

AES-based functions.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "aes_wrap.h"
#include "crypto.h"
#include "aes.c"
```

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.
- 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.

6.2.1 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](#).

6.2.2 Function Documentation

6.2.2.1 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

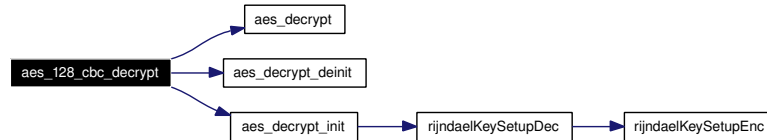
data_len Length of data in bytes (must be divisible by 16)

Returns:

0 on success, -1 on failure

Definition at line 441 of file aes_wrap.c.

Here is the call graph for this function:

**6.2.2.2 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

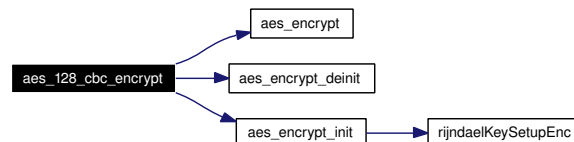
data_len Length of data in bytes (must be divisible by 16)

Returns:

0 on success, -1 on failure

Definition at line 407 of file aes_wrap.c.

Here is the call graph for this function:

**6.2.2.3 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

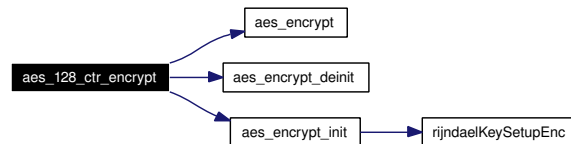
data_len Length of data in bytes

Returns:

0 on success, -1 on failure

Definition at line 245 of file aes_wrap.c.

Here is the call graph for this function:



6.2.2.4 `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:

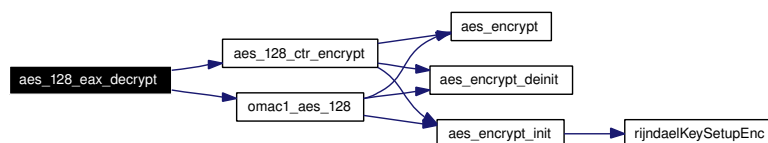
- key* Key for decryption (16 bytes)
- nonce* Nonce for counter mode
- nonce_len* Nonce length in bytes
- hdr* Header data to be authenticity protected
- hdr_len* Length of the header data bytes
- data* Data to encrypt in-place
- data_len* Length of data in bytes
- tag* 16-byte tag value

Returns:

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

Definition at line 350 of file aes_wrap.c.

Here is the call graph for this function:



6.2.2.5 `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

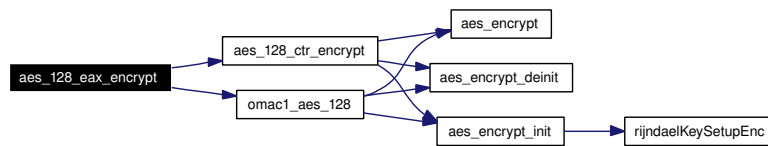
tag 16-byte tag value

Returns:

0 on success, -1 on failure

Definition at line 292 of file aes_wrap.c.

Here is the call graph for this function:



6.2.2.6 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)

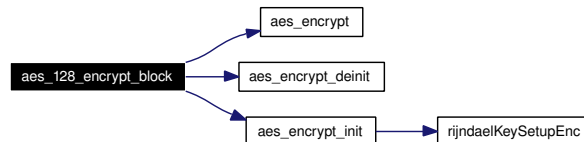
out Output of the AES block operation (16 bytes)

Returns:

0 on success, -1 on failure

Definition at line 224 of file aes_wrap.c.

Here is the call graph for this function:



6.2.2.7 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

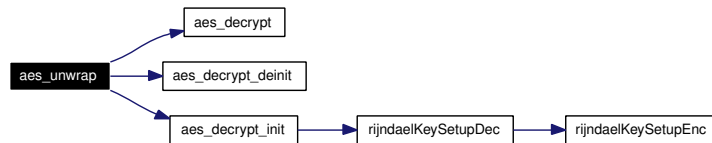
plain Plaintext key, $n * 64$ bit

Returns:

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

Definition at line 101 of file aes_wrap.c.

Here is the call graph for this function:



6.2.2.8 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

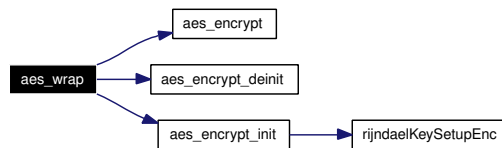
cipher Wrapped key, $(n + 1) * 64$ bit

Returns:

0 on success, -1 on failure

Definition at line 44 of file aes_wrap.c.

Here is the call graph for this function:



6.2.2.9 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.

Parameters:

key Key for the hash operation

data Data buffer for which a MAC is determined

data Length of data buffer in bytes

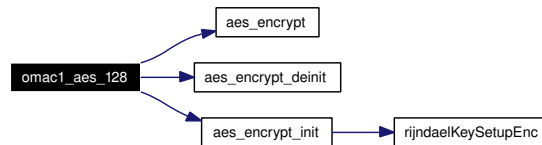
mac Buffer for MAC (128 bits, i.e., 16 bytes)

Returns:

0 on success, -1 on failure

Definition at line 176 of file aes_wrap.c.

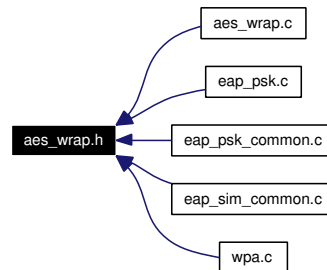
Here is the call graph for this function:



6.3 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.
- 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.h](#).

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:

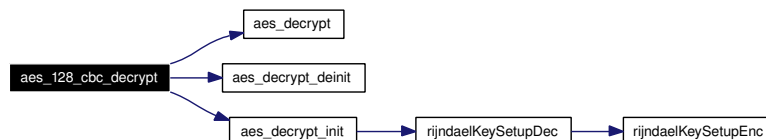
- key* Decryption key
- iv* Decryption IV for CBC mode (16 bytes)
- data* Data to decrypt in-place
- data_len* Length of data in bytes (must be divisible by 16)

Returns:

- 0 on success, -1 on failure

Definition at line 441 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

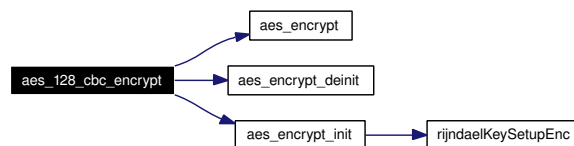
data_len Length of data in bytes (must be divisible by 16)

Returns:

0 on success, -1 on failure

Definition at line 407 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

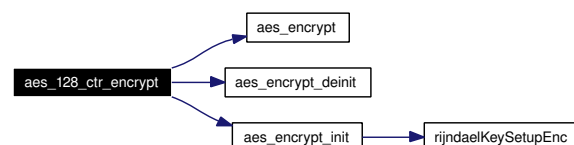
data_len Length of data in bytes

Returns:

0 on success, -1 on failure

Definition at line 245 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:

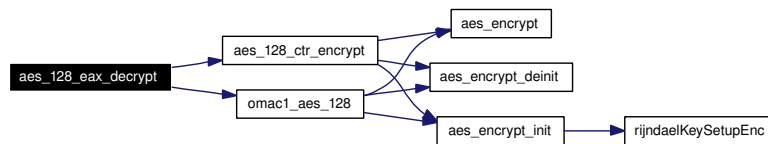
key Key for decryption (16 bytes)
nonce Nonce for counter mode
nonce_len Nonce length in bytes
hdr Header data to be authenticity protected
hdr_len Length of the header data bytes
data Data to encrypt in-place
data_len Length of data in bytes
tag 16-byte tag value

Returns:

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

Definition at line 350 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:

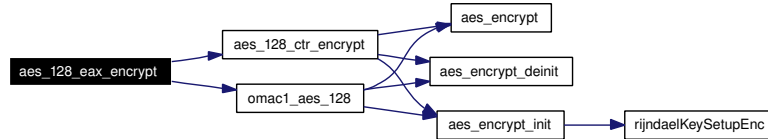
key Key for encryption (16 bytes)
nonce Nonce for counter mode
nonce_len Nonce length in bytes
hdr Header data to be authenticity protected
hdr_len Length of the header data bytes
data Data to encrypt in-place
data_len Length of data in bytes
tag 16-byte tag value

Returns:

0 on success, -1 on failure

Definition at line 292 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:

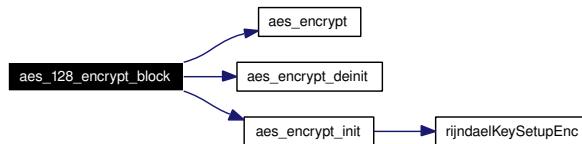
- key* Key for AES
- in* Input data (16 bytes)
- out* Output of the AES block operation (16 bytes)

Returns:

- 0 on success, -1 on failure

Definition at line 224 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:

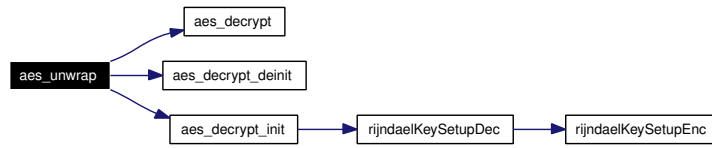
- kek* Key encryption key (KEK)
- n* Length of the wrapped key in 64-bit units; e.g., 2 = 128-bit = 16 bytes
- cipher* Wrapped key to be unwrapped, $(n + 1) * 64$ bit
- plain* Plaintext key, $n * 64$ bit

Returns:

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

Definition at line 101 of file aes_wrap.c.

Here is the call graph for this function:



6.3.2.8 int aes_unwrap (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

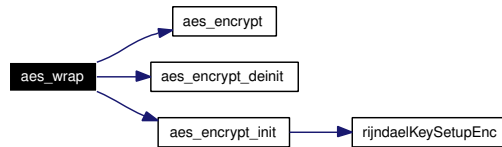
cipher Wrapped key, $(n + 1) * 64$ bit

Returns:

0 on success, -1 on failure

Definition at line 44 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.

Parameters:

key Key for the hash operation

data Data buffer for which a MAC is determined

data Length of data buffer in bytes

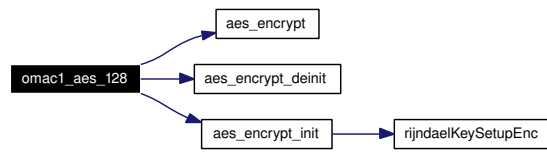
mac Buffer for MAC (128 bits, i.e., 16 bytes)

Returns:

0 on success, -1 on failure

Definition at line 176 of file aes_wrap.c.

Here is the call graph for this function:



6.4 base64.c File Reference

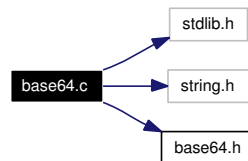
Base64 encoding/decoding (RFC1341).

```
#include <stdlib.h>
```

```
#include <string.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.

6.4.1 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](#).

6.4.2 Function Documentation

6.4.2.1 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.

6.4.2.2 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

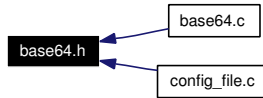
Caller is responsible for freeing the returned buffer. Returned buffer is nul terminated to make it easier to use as a C string. The nul terminator is not included in *out_len*.

Definition at line 37 of file base64.c.

6.5 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.

6.5.1 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](#).

6.5.2 Function Documentation

6.5.2.1 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.

6.5.2.2 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*.

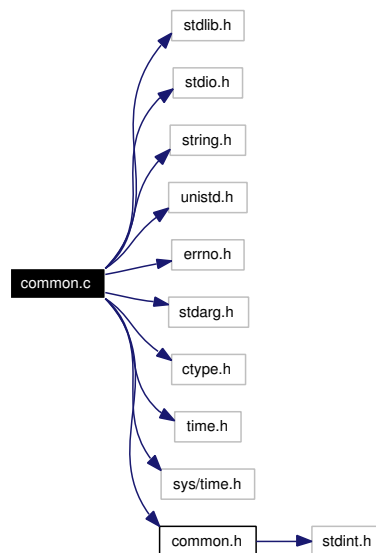
Definition at line 37 of file base64.c.

6.6 common.c File Reference

wpa_supplicant/hostapd / common helper functions, etc.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <stdarg.h>
#include <ctype.h>
#include <time.h>
#include <sys/time.h>
#include "common.h"
```

Include dependency graph for common.c:



Functions

- int **hostapd_get_rand** (u8 *buf, size_t len)
- void **hostapd_hexdump** (const char *title, const u8 *buf, size_t len)
- 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.
- char * **rel2abs_path** (const char *rel_path)
- void **inc_byte_array** (u8 *counter, size_t len)

Increment arbitrary length byte array by one.

- void **print_char** (char c)
- void **fprint_char** (FILE *f, char c)
- 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

Variables

- int **wpa_debug_level** = MSG_INFO
- int **wpa_debug_show_keys** = 0
- int **wpa_debug_timestamp** = 0

6.6.1 Detailed Description

wpa_supplicant/hostapd / common helper functions, etc.

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 [common.c](#).

6.6.2 Function Documentation

6.6.2.1 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 143 of file common.c.

6.6.2.2 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 112 of file common.c.

6.6.2.3 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 209 of file common.c.

6.6.2.4 void wpa_debug_print_timestamp (void)

Print timestamp for debug output.

This function prints a timestamp in <seconds from 1970>.<microseconds> format if debug output has been configured to include timestamps in debug messages.

Definition at line 241 of file common.c.

6.6.2.5 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 303 of file common.c.

6.6.2.6 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 359 of file common.c.

6.6.2.7 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 365 of file common.c.

6.6.2.8 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 309 of file common.c.

6.6.2.9 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

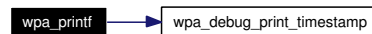
This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration.

Note: New line `

` is added to the end of the text when printing to stdout.

Definition at line 270 of file common.c.

Here is the call graph for this function:



6.7 common.h File Reference

wpa_supplicant/hostapd / common helper functions, etc.

```
#include <stdint.h>
```

Include dependency graph for common.h:



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

Defines

- #define **le_to_host16**(n) (n)
- #define **host_to_le16**(n) (n)
- #define **be_to_host16**(n) bswap_16(n)
- #define **host_to_be16**(n) bswap_16(n)
- #define **le_to_host32**(n) (n)
- #define **be_to_host32**(n) bswap_32(n)
- #define **host_to_be32**(n) bswap_32(n)
- #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_BE32**(a)
- #define **ETH_ALEN** 6
- #define **WPA_ASSERT**(a) do { } while (0)

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**

Enumerations

- enum {
MSG_MSGDUMP, **MSG_DEBUG**, **MSG_INFO**, **MSG_WARNING**,
MSG_ERROR }

Functions

- int **hostapd_get_rand** (u8 *buf, size_t len)
- void **hostapd_hexdump** (const char *title, const u8 *buf, size_t len)
- 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.
- char * **rel2abs_path** (const char *rel_path)
- void **inc_byte_array** (u8 *counter, size_t len)
Increment arbitrary length byte array by one.
- void **print_char** (char c)

- void **fprint_char** (FILE *f, char c)
- void **wpa_debug_print_timestamp** (void)
Print timestamp for debug output.
- void **wpa_printf** (int level, char *fmt,...) `__attribute__((format(printf, 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

6.7.1 Detailed Description

wpa_supplicant/hostapd / common helper functions, etc.

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 [common.h](#).

6.7.2 Define Documentation

6.7.2.1 #define WPA_GET_BE32(a)

Value:

```
((((u32) (a)[0]) << 24) | (((u32) (a)[1]) << 16) | \
  (((u32) (a)[2]) << 8) | ((u32) (a)[3]))
```

Definition at line 149 of file common.h.

6.7.2.2 #define WPA_PUT_BE16(a, val)

Value:

```
do {
    (a)[0] = ((u16) (val)) >> 8; \
    (a)[1] = ((u16) (val)) & 0xff; \
} while (0)
```

Definition at line 136 of file common.h.

6.7.2.3 #define WPA_PUT_LE16(a, val)

Value:

```
do {
    (a)[1] = ((u16) (val)) >> 8; \
    (a)[0] = ((u16) (val)) & 0xff; \
} while (0)
```

Definition at line 143 of file common.h.

6.7.3 Function Documentation

6.7.3.1 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 143 of file common.c.

6.7.3.2 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 112 of file common.c.

6.7.3.3 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 209 of file common.c.

6.7.3.4 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 241 of file common.c.

6.7.3.5 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 303 of file common.c.

6.7.3.6 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 359 of file common.c.

6.7.3.7 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 365 of file common.c.

6.7.3.8 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 309 of file common.c.

6.7.3.9 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 `

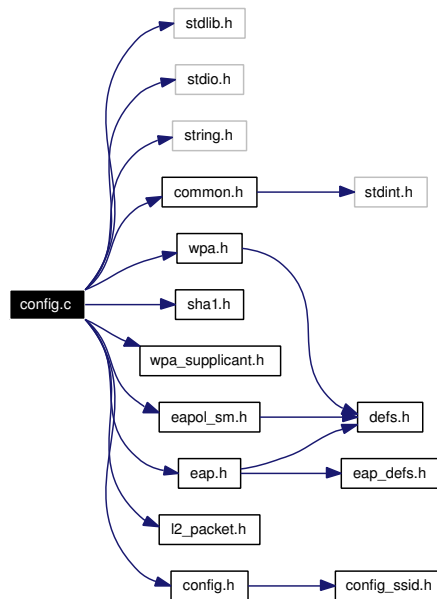
` is added to the end of the text when printing to stdout.

6.8 config.c File Reference

WPA Supplicant / Configuration parser and common functions.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "wpa.h"
#include "sha1.h"
#include "wpa_supplicant.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)
- #define **STR_LEN**(f) STR(f), .param2 = OFFSET(f ## _len)
- #define **STR_RANGE**(f, min, max)
- #define **INT**(f)
- #define **INT_RANGE**(f, min, max)
- #define **FUNC**(f)
- #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 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.
- 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.

6.8.1 Detailed Description

WPA Supplicant / Configuration parser and common 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 [config.c](#).

6.8.2 Define Documentation

6.8.2.1 #define FUNC(f)

Value:

```
.name = #f, .parser = wpa_config_parse_ ## f, \  
          .writer = wpa_config_write_ ## f
```

Definition at line 985 of file config.c.

6.8.2.2 #define INT(f)

Value:

```
.name = #f, .parser = wpa_config_parse_int, \  
          .writer = wpa_config_write_int, \  
          .param1 = OFFSET(f), .param2 = (void *) 0
```

Definition at line 975 of file config.c.

6.8.2.3 #define INT_RANGE(f, min, max)

Value:

```
INT(f), .param3 = (void *) (min), \  
        .param4 = (void *) (max)
```

Definition at line 980 of file config.c.

6.8.2.4 #define STR(f)

Value:

```
.name = #f, .parser = wpa_config_parse_str, \  
          .writer = wpa_config_write_str, .param1 = OFFSET(f)
```

Definition at line 960 of file config.c.

6.8.2.5 #define STR_RANGE(f, min, max)

Value:

```
STR_LEN(f), .param3 = (void *) (min), \  
    .param4 = (void *) (max)
```

Definition at line 970 of file config.c.

6.8.3 Function Documentation

6.8.3.1 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 1296 of file config.c.

6.8.3.2 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 1086 of file config.c.

6.8.3.3 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 1563 of file config.c.

6.8.3.4 int wpa_config_allowed_eap_method (struct wpa_ssid * ssid, int method)

Check whether EAP method is allowed.

Parameters:

ssid Pointer to configuration data

method EAP type

Returns:

1 = allowed EAP method, 0 = not allowed

Definition at line 1251 of file config.c.

6.8.3.5 void wpa_config_free (struct wpa_config * config)

Free configuration data.

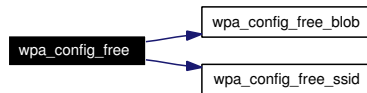
Parameters:

config Configuration data from [wpa_config_read\(\)](#)

This function frees all resources allocated for the configuration data by [wpa_config_read\(\)](#).

Definition at line 1215 of file config.c.

Here is the call graph for this function:



6.8.3.6 void wpa_config_free_blob (struct wpa_config_blob * blob)

Free blob data.

Parameters:

blob Pointer to blob to be freed

Definition at line 1517 of file config.c.

6.8.3.7 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 1167 of file config.c.

6.8.3.8 `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 1437 of file config.c.

6.8.3.9 `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 1480 of file config.c.

6.8.3.10 `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 1275 of file config.c.

6.8.3.11 `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 1534 of file config.c.

Here is the call graph for this function:



6.8.3.12 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 1334 of file config.c.

Here is the call graph for this function:



6.8.3.13 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

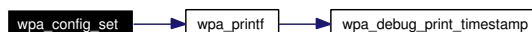
Returns:

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 1390 of file config.c.

Here is the call graph for this function:



6.8.3.14 void wpa_config_set_blob (struct wpa_config * config, struct wpa_config_blob * blob)

Set or add a named configuration blob.

Parameters:

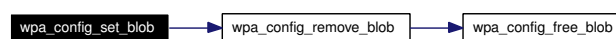
config Configuration data from wpa_config_read()

blob New value for the blob

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

Definition at line 1503 of file config.c.

Here is the call graph for this function:

**6.8.3.15 void wpa_config_set_network_defaults (struct wpa_ssid * ssid)**

Set network default values.

Parameters:

ssid Pointer to network configuration data

Definition at line 1365 of file config.c.

6.8.3.16 void wpa_config_update_psk (struct wpa_ssid * ssid)

Update WPA PSK based on passphrase and SSID.

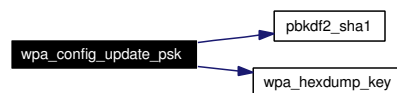
Parameters:

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 1462 of file config.c.

Here is the call graph for this function:



6.9 config.h File Reference

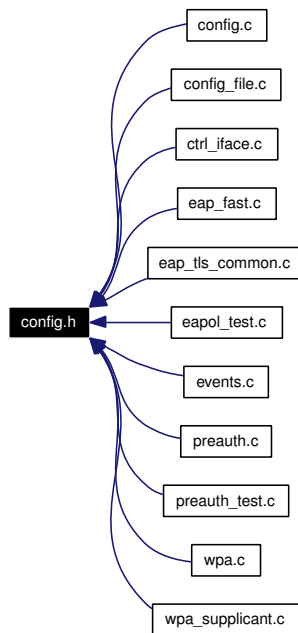
WPA Supplicant / Configuration file structures.

```
#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.
- `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.
- `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.

6.9.1 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](#).

6.9.2 Function Documentation

6.9.2.1 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 1296 of file [config.c](#).

6.9.2.2 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 1086 of file [config.c](#).

6.9.2.3 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 1563 of file [config.c](#).

6.9.2.4 void wpa_config_free (struct wpa_config * config)

Free configuration data.

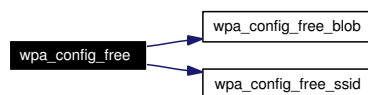
Parameters:

config Configuration data from [wpa_config_read\(\)](#)

This function frees all resources allocated for the configuration data by [wpa_config_read\(\)](#).

Definition at line 1215 of file config.c.

Here is the call graph for this function:



6.9.2.5 void wpa_config_free_blob (struct wpa_config_blob * blob)

Free blob data.

Parameters:

blob Pointer to blob to be freed

Definition at line 1517 of file config.c.

6.9.2.6 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 1167 of file config.c.

6.9.2.7 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 1437 of file config.c.

6.9.2.8 `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 1480 of file config.c.

6.9.2.9 `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 1275 of file config.c.

6.9.2.10 `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:

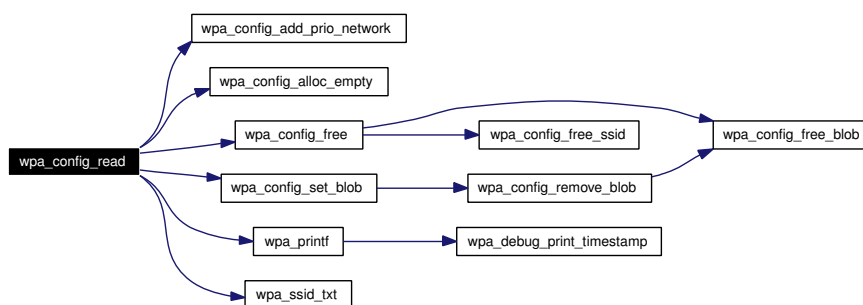
Pointer to allocated configuration data or NULL on failure

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 213 of file config_file.c.

Here is the call graph for this function:



6.9.2.11 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 1534 of file config.c.

Here is the call graph for this function:



6.9.2.12 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 1334 of file config.c.

Here is the call graph for this function:



6.9.2.13 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

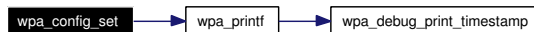
Returns:

- 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 1390 of file config.c.

Here is the call graph for this function:



6.9.2.14 void wpa_config_set_blob (struct wpa_config * config, struct wpa_config_blob * blob)

Set or add a named configuration blob.

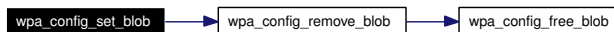
Parameters:

- 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 1503 of file config.c.

Here is the call graph for this function:



6.9.2.15 void wpa_config_set_network_defaults (struct wpa_ssid * ssid)

Set network default values.

Parameters:

- ssid* Pointer to network configuration data

Definition at line 1365 of file config.c.

6.9.2.16 void wpa_config_update_psk (struct wpa_ssid * ssid)

Update WPA PSK based on passphrase and SSID.

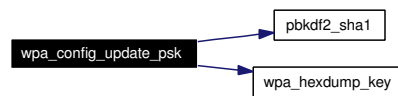
Parameters:

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 1462 of file config.c.

Here is the call graph for this function:

**6.9.2.17 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 624 of file config_file.c.

Here is the call graph for this function:

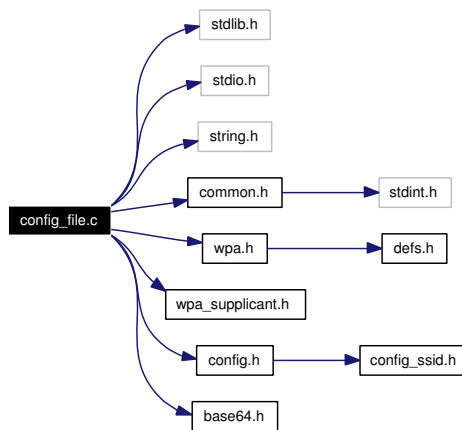


6.10 config_file.c File Reference

WPA Supplicant / Configuration backend: text file.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "wpa.h"
#include "wpa_supplicant.h"
#include "config.h"
#include "base64.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.

6.10.1 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](#).

6.10.2 Function Documentation**6.10.2.1 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:

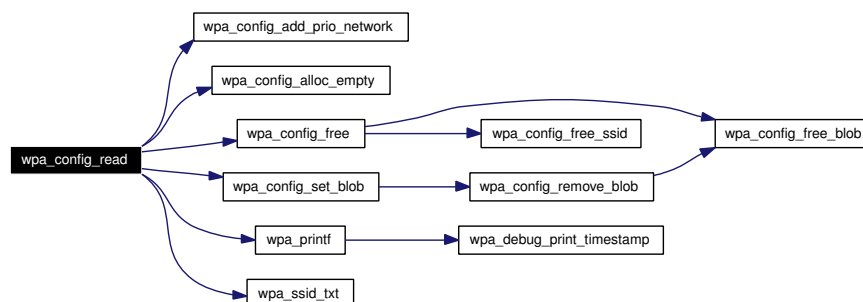
Pointer to allocated configuration data or NULL on failure

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 213 of file config_file.c.

Here is the call graph for this function:

**6.10.2.2 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:

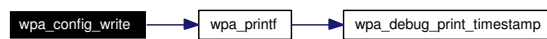
0 on success, -1 on failure

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 624 of file config_file.c.

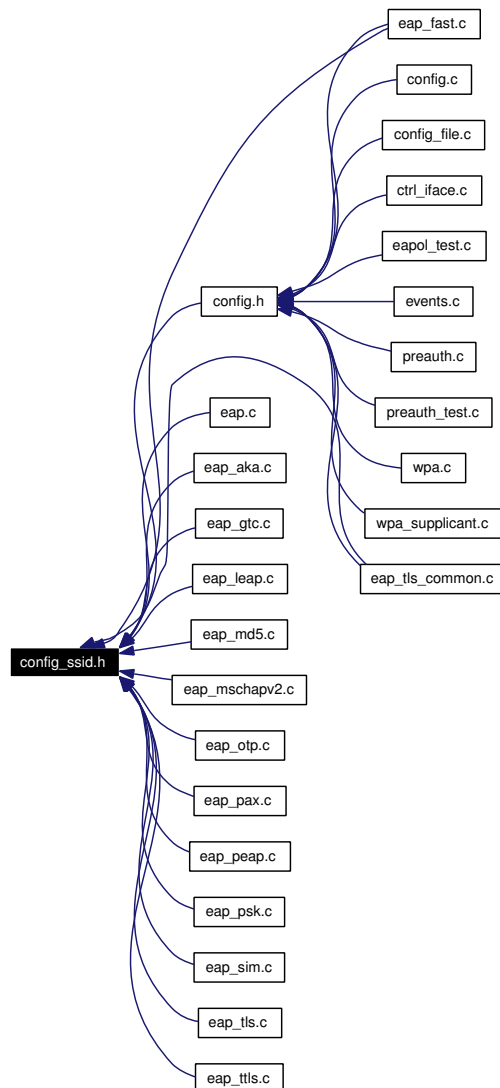
Here is the call graph for this function:



6.11 config_ssid.h File Reference

WPA Supplicant / Network configuration structures.

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



Defines

- #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** 16
- #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 **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 method)
Check whether EAP method is allowed.

6.11.1 Detailed Description

WPA Supplicant / Network configuration 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_ssid.h](#).

6.11.2 Define Documentation

6.11.2.1 #define DEFAULT_EAPOL_FLAGS

Value:

```
(EAPOL_FLAG_REQUIRE_KEY_UNICAST | \
    EAPOL_FLAG_REQUIRE_KEY_BROADCAST)
```

Definition at line 44 of file [config_ssid.h](#).

6.11.2.2 #define DEFAULT_GROUP

Value:

```
(WPA_CIPHER_CCMP | WPA_CIPHER_TKIP | \
    WPA_CIPHER_WEP104 | WPA_CIPHER_WEP40)
```

Definition at line 49 of file config_ssid.h.

6.11.3 Function Documentation

6.11.3.1 int wpa_config_allowed_eap_method (struct [wpa_ssid](#) * *ssid*, int *method*)

Check whether EAP method is allowed.

Parameters:

ssid Pointer to configuration data

method EAP type

Returns:

1 = allowed EAP method, 0 = not allowed

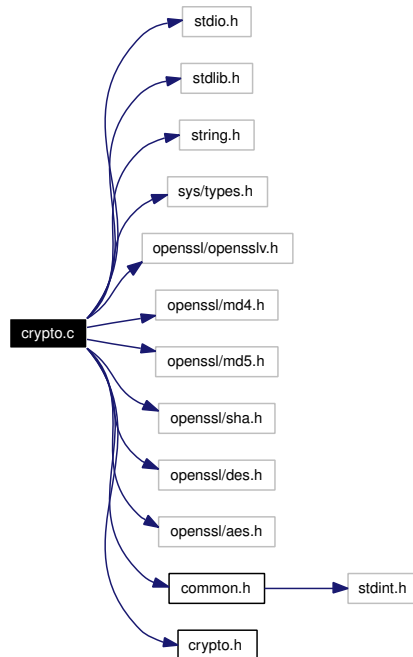
Definition at line 1251 of file config.c.

6.12 crypto.c File Reference

WPA Supplicant / wrapper functions for libcrypto.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.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.

6.12.1 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](#).

6.12.2 Function Documentation

6.12.2.1 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 52 of file `crypto.c`.

6.12.2.2 void md4_vector (size_t num_elem, const u8 * addr[], const size_t * len, u8 * mac)

MD4 hash for data vector.

Parameters:

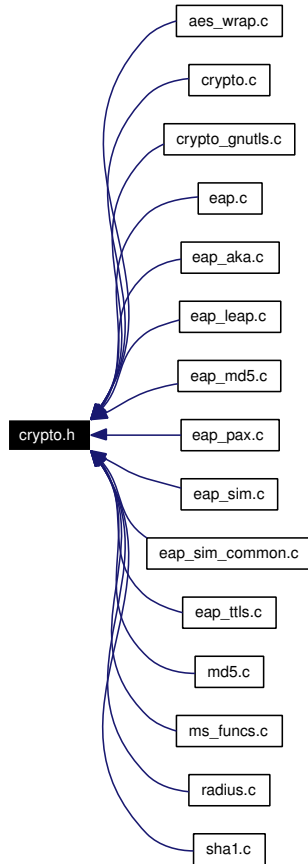
- num_elem* Number of elements in the data vector
- addr* Pointers to the data areas
- len* Lengths of the data blocks
- mac* Buffer for the hash

Definition at line 40 of file `crypto.c`.

6.13 crypto.h File Reference

WPA Supplicant / wrapper functions for crypto libraries.

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



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.
- void [sha1_transform](#) (u8 *state, const u8 data[64])
Perform one SHA-1 transform step.
- 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.

6.13.1 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](#).

6.13.2 Function Documentation

6.13.2.1 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 1097 of file aes.c.

6.13.2.2 void aes_decrypt_deinit (void * ctx)

Deinitialize AES decryption.

Parameters:

ctx Context pointer from [aes_encrypt_init\(\)](#)

Definition at line 1103 of file aes.c.

6.13.2.3 void* aes_decrypt_init (const u8 * key, size_t len)

Initialize AES for decryption.

Parameters:

key Decryption key

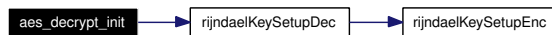
len Key length in bytes (usually 16, i.e., 128 bits)

Returns:

Pointer to context data or NULL on failure

Definition at line 1084 of file aes.c.

Here is the call graph for this function:



6.13.2.4 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 1072 of file aes.c.

6.13.2.5 void aes_encrypt_deinit (void * ctx)

Deinitialize AES encryption.

Parameters:

ctx Context pointer from [aes_encrypt_init\(\)](#)

Definition at line 1078 of file aes.c.

6.13.2.6 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 1059 of file aes.c.

Here is the call graph for this function:

**6.13.2.7 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 52 of file crypto.c.

6.13.2.8 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 40 of file crypto.c.

6.13.2.9 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

Definition at line 141 of file md5.c.

6.13.2.10 void sha1_transform (u8 * state, const u8 data[64])

Perform one SHA-1 transform step.

Parameters:

state SHA-1 state

data Input data for the SHA-1 transform

This function is used to implement 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 and as such, access to just part of the SHA-1 is needed.

Definition at line 424 of file sha1.c.

6.13.2.11 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

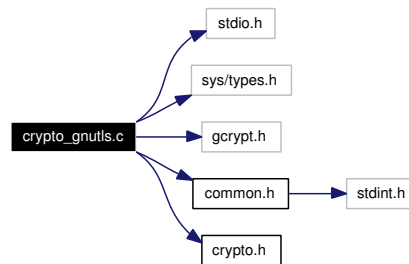
Definition at line 400 of file sha1.c.

6.14 crypto_gnutls.c File Reference

WPA Supplicant / wrapper functions for libgcrypt.

```
#include <stdio.h>
#include <sys/types.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.

6.14.1 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](#).

6.14.2 Function Documentation

6.14.2.1 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 40 of file crypto_gnutls.c.

6.14.2.2 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

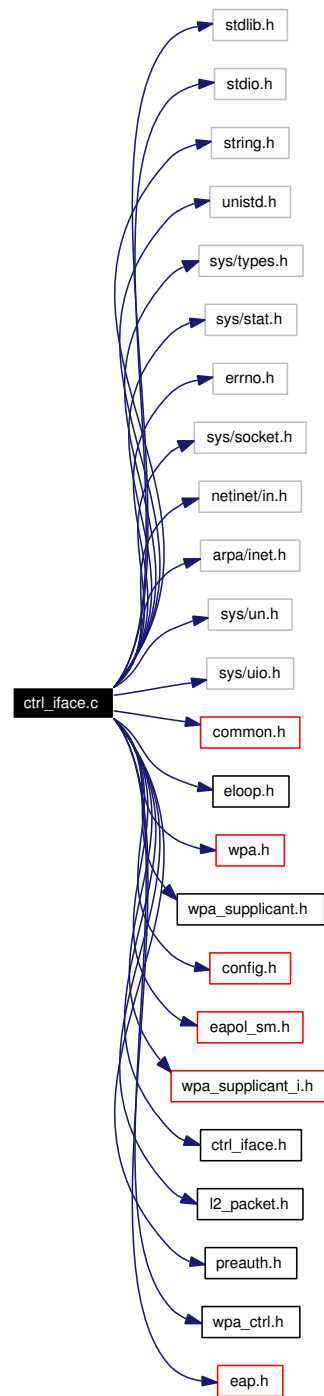
Definition at line 23 of file crypto_gnutls.c.

6.15 ctrl_iface.c File Reference

WPA Supplicant / UNIX domain and UDP socket -based control interface.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/un.h>
#include <sys/uio.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 "wpa_ctrl.h"
#include "eap.h"
```

Include dependency graph for ctrl_iface.c:



Defines

- `#define CTRL_IFACE_SOCKET` struct sockaddr_un

Functions

- `int wpa_supplicant_ctrl_iface_init` (struct `wpa_supplicant` *wpa_s)

Initialize control interface.

- void `wpa_supplicant_ctrl_iface_deinit` (struct `wpa_supplicant` *`wpa_s`)
Deinitialize control interface.
- void `wpa_supplicant_ctrl_iface_send` (struct `wpa_supplicant` *`wpa_s`, int level, char *buf, size_t len)
Send a control interface packet to monitors.
- void `wpa_supplicant_ctrl_iface_wait` (struct `wpa_supplicant` *`wpa_s`)
Wait for ctrl_iface monitor.
- int `wpa_supplicant_global_ctrl_iface_init` (struct `wpa_global` *`global`)
Initialize global control interface.
- void `wpa_supplicant_global_ctrl_iface_deinit` (struct `wpa_global` *`global`)
Deinitialize global ctrl interface.

6.15.1 Detailed Description

WPA Supplicant / UNIX domain and 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.c](#).

6.15.2 Function Documentation

6.15.2.1 void wpa_supplicant_ctrl_iface_deinit (struct wpa_supplicant * wpa_s)

Deinitialize control interface.

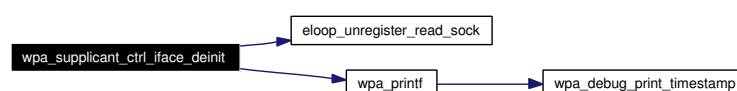
Parameters:

`wpa_s` Pointer to `wpa_supplicant` data

Deinitialize the control interface that was initialized with `wpa_supplicant_ctrl_iface_init()`.

Definition at line 1270 of file `ctrl_iface.c`.

Here is the call graph for this function:



6.15.2.2 int wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * wpa_s)

Initialize control interface.

Parameters:

wpa_s Pointer to [wpa_supplicant](#) data

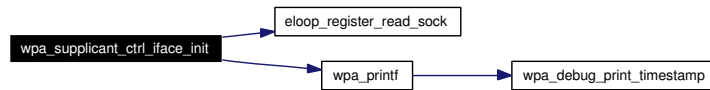
Returns:

0 on success, -1 on failure

Initialize the control interface and start receiving commands from external programs.

Definition at line 1137 of file ctrl_iface.c.

Here is the call graph for this function:



6.15.2.3 void wpa_supplicant_ctrl_iface_send (struct wpa_supplicant * wpa_s, int level, char * buf, size_t len)

Send a control interface packet to monitors.

Parameters:

wpa_s Pointer to [wpa_supplicant](#) data

level Priority level of the message

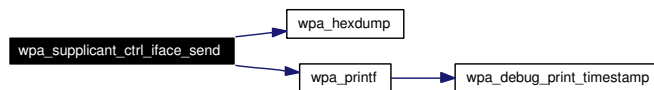
buf Message data

len Message length

Send a packet to all monitor programs attached to the control interface.

Definition at line 1328 of file ctrl_iface.c.

Here is the call graph for this function:



6.15.2.4 void wpa_supplicant_ctrl_iface_wait (struct wpa_supplicant * wpa_s)

Wait for ctrl_iface monitor.

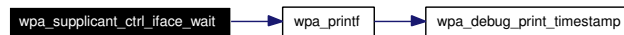
Parameters:

wpa_s Pointer to [wpa_supplicant](#) data

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.

Definition at line 1430 of file ctrl_iface.c.

Here is the call graph for this function:



6.15.2.5 void wpa_supplicant_global_ctrl_iface_deinit (struct wpa_global * global)

Deinitialize global ctrl interface.

Parameters:

global Pointer to global data from [wpa_supplicant_init\(\)](#)

Deinitialize the global control interface that was initialized with [wpa_supplicant_global_ctrl_iface_init\(\)](#).

Definition at line 1682 of file ctrl_iface.c.

Here is the call graph for this function:



6.15.2.6 int wpa_supplicant_global_ctrl_iface_init (struct wpa_global * global)

Initialize global control interface.

Parameters:

global Pointer to global data from [wpa_supplicant_init\(\)](#)

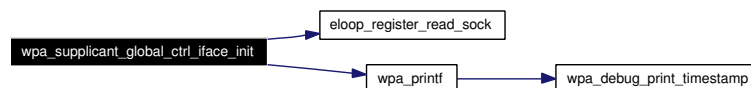
Returns:

0 on success, -1 on failure

Initialize the global control interface and start receiving commands from external programs.

Definition at line 1590 of file ctrl_iface.c.

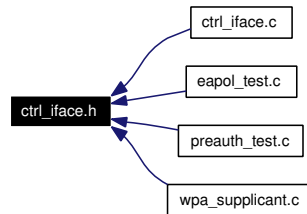
Here is the call graph for this function:



6.16 ctrl_iface.h File Reference

WPA Supplicant / UNIX domain socket -based control interface.

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



6.16.1 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.

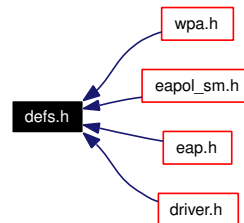
See README and COPYING for more details.

Definition in file [ctrl_iface.h](#).

6.17 defs.h File Reference

WPA Supplicant - Common definitions.

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



Enumerations

- enum **Boolean** { **FALSE** = 0, **TRUE** = 1 }
- enum **wpa_alg** { **WPA_ALG_NONE**, **WPA_ALG_WEP**, **WPA_ALG_TKIP**, **WPA_ALG_CCMP** }
- 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** }

6.17.1 Detailed Description

WPA Supplicant - Common definitions.

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 [defs.h](#).

6.17.2 Enumeration Type Documentation

6.17.2.1 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 44 of file defs.h.

6.18 driver.h File Reference

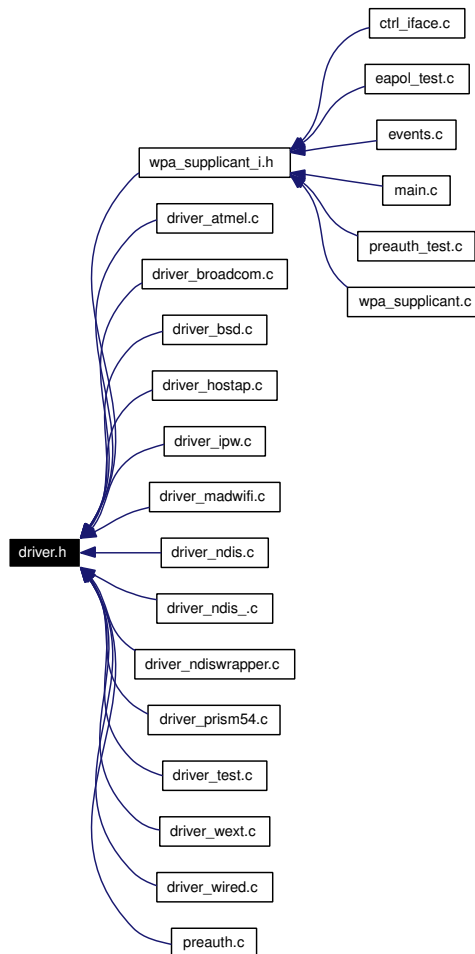
WPA Supplicant - driver interface definition.

```
#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

6.18.1 Detailed Description

WPA Supplicant - driver 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.

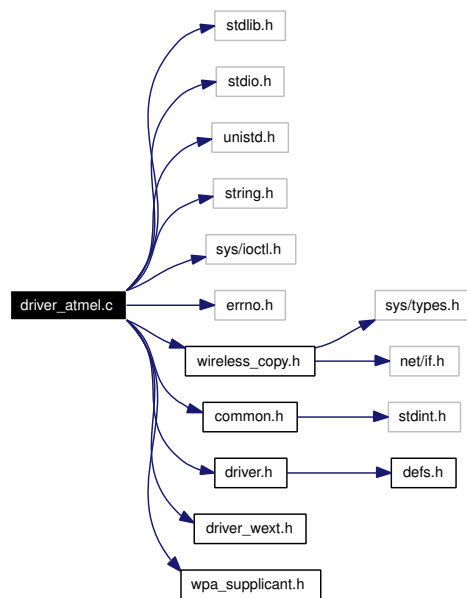
Definition in file [driver.h](#).

6.19 driver_atmel.c File Reference

WPA Supplicant - Driver interaction with Atmel Wireless LAN drivers.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.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](#)

6.19.1 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](#).

6.19.2 Variable Documentation

6.19.2.1 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,
}
```

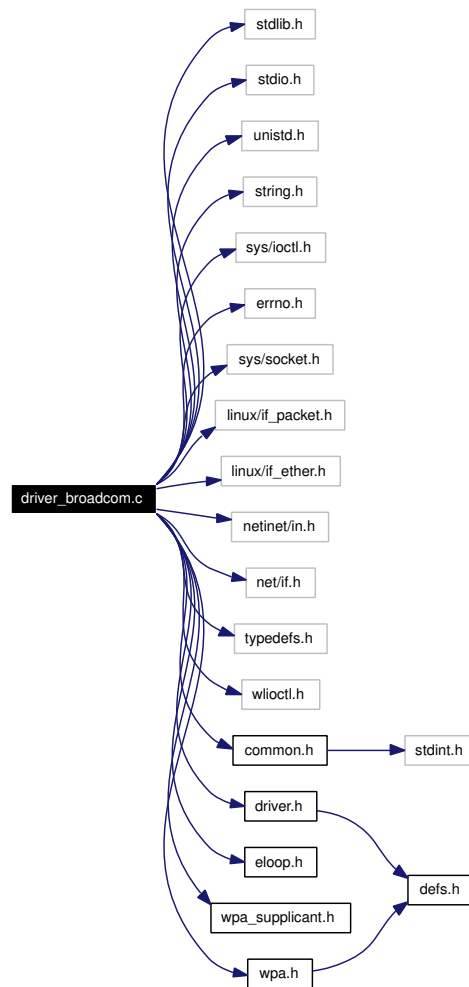
Definition at line 492 of file [driver_atmel.c](#).

6.20 driver_broadcom.c File Reference

WPA Supplicant - driver interaction with Broadcom wl.o driver.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.h>
#include <sys/socket.h>
#include <linux/if_packet.h>
#include <linux/if_ether.h>
#include <netinet/in.h>
#include <net/if.h>
#include <typedefs.h>
#include <wliioctl.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`

6.20.1 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](#).

6.20.2 Variable Documentation

6.20.2.1 `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,
}
```

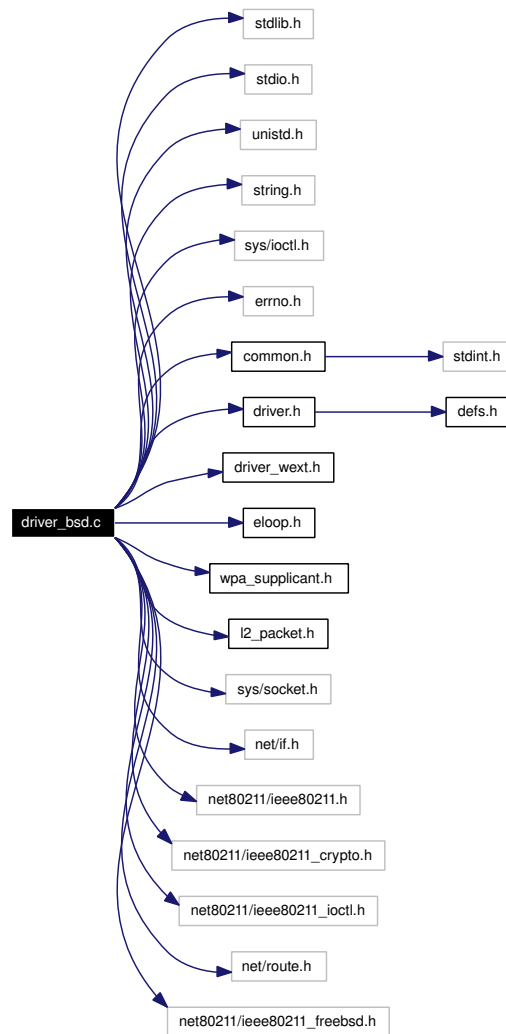
Definition at line 585 of file [driver_broadcom.c](#).

6.21 driver_bsd.c File Reference

WPA Supplicant - driver interaction with BSD net80211 layer.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.h>
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "l2_packet.h"
#include <sys/socket.h>
#include <net/if.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))

Variables

- const struct [wpa_driver_ops](#) **wpa_driver_bsd_ops**

6.21.1 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](#).

6.21.2 Define Documentation

6.21.2.1 #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 494 of file driver_bsd.c.

6.21.3 Variable Documentation

6.21.3.1 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,
}
```

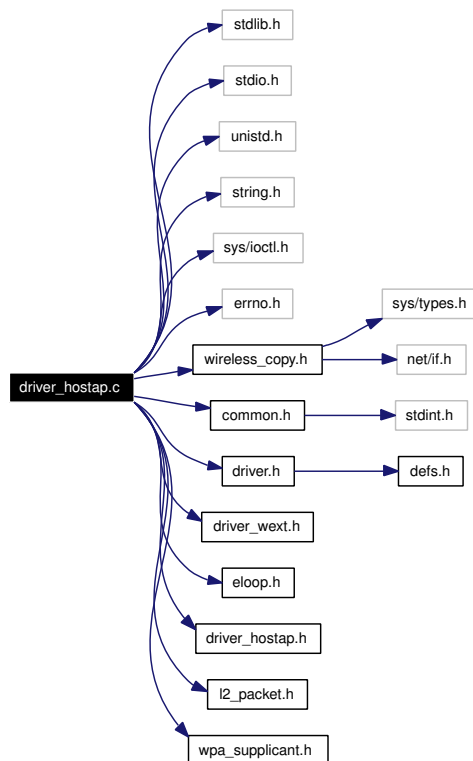
Definition at line 620 of file driver_bsd.c.

6.22 driver_hostap.c File Reference

WPA Supplicant - driver interaction with Linux Host AP driver.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.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](#)

6.22.1 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](#).

6.22.2 Variable Documentation

6.22.2.1 const struct [wpa_driver_ops](#) [wpa_driver_hostap_ops](#)

Initial value:

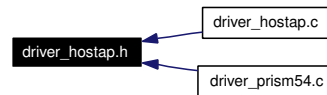
```
{
    .name = "hostap",
    .desc = "Host AP driver (Intersil Prism2/2.5/3)",
    .get_bssid = wpa_driver_hostap_get_bssid,
    .get_ssid = wpa_driver_hostap_get_ssid,
    .set_wpa = wpa_driver_hostap_set_wpa,
    .set_key = wpa_driver_hostap_set_key,
    .set_countermeasures = wpa_driver_hostap_set_countermeasures,
    .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,
}
```

Definition at line 498 of file [driver_hostap.c](#).

6.23 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 }

```

6.23.1 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.

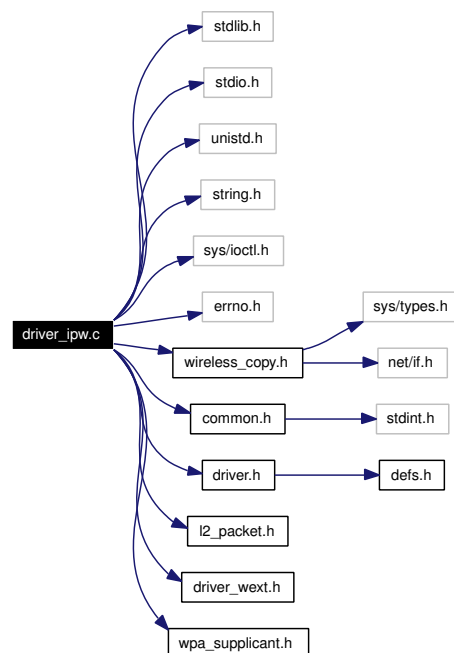
Definition in file [driver_hostap.h](#).

6.24 driver_ipw.c File Reference

WPA Supplicant - driver interaction with Linux ipw2100/2200 drivers.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.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**

6.24.1 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.

Definition in file [driver_ipw.c](#).

6.24.2 Variable Documentation

6.24.2.1 const struct [wpa_driver_ops](#) **wpa_driver_ipw_ops**

Initial value:

```
{
    .name = "ipw",
    .desc = "Intel ipw2100/2200 driver",
    .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,  
}
```

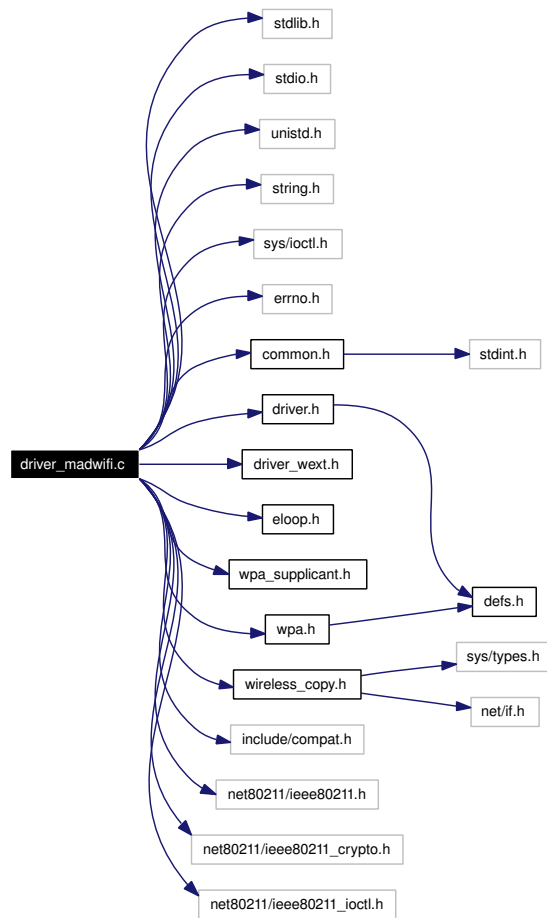
Definition at line 435 of file driver_ipw.c.

6.25 driver_madwifi.c File Reference

WPA Supplicant - driver interaction with MADWIFI 802.11 driver.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.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**

6.25.1 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](#).

6.25.2 Variable Documentation

6.25.2.1 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,
}
```

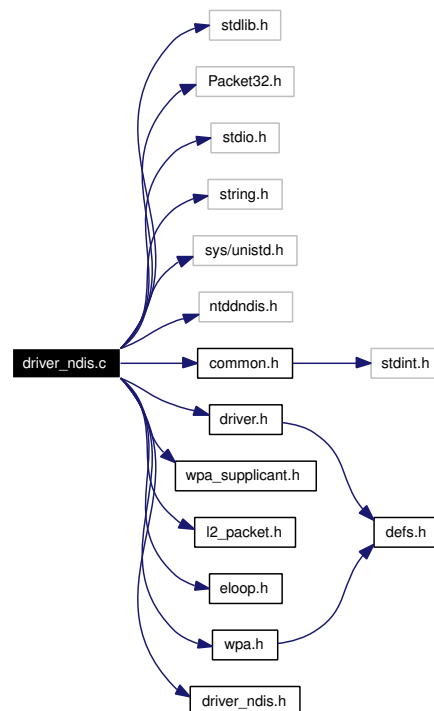
Definition at line 514 of file `driver_madwifi.c`.

6.26 driver_ndis.c File Reference

WPA Supplicant - Windows/NDIS driver interface.

```
#include <stdlib.h>
#include <Packet32.h>
#include <stdio.h>
#include <string.h>
#include <sys/unistd.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_PREAMTH_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

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

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_11AuthModeAutoSwitch,
Ndis802_11AuthModeWPA,
Ndis802_11AuthModeWPAPSK, Ndis802_11AuthModeWPANone, Ndis802_11AuthModeWPA2,
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_11WEPKeyAbsent,
Ndis802_11WEPNotSupported, Ndis802_11EncryptionNotSupported = Ndis802_11WEPNotSupported,
Ndis802_11Encryption2Enabled, Ndis802_11Encryption2KeyAbsent, Ndis802_11Encryption3Enabled,
Ndis802_11Encryption3KeyAbsent }
- enum NDIS_802_11_PRIVACY_FILTER { Ndis802_11PrivFilterAcceptAll, Ndis802_11PrivFilter8021xWEP }
- 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_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)

Variables

- const struct [wpa_driver_ops](#) **wpa_driver_ndis_ops**

6.26.1 Detailed Description

WPA Supplicant - Windows/NDIS driver 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 [driver_ndis.c](#).

6.26.2 Variable Documentation

6.26.2.1 const struct [wpa_driver_ops](#) wpa_driver_ndis_ops

Initial value:

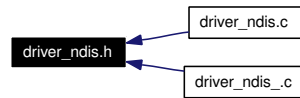
```
{
    .name = "ndis",
    .desc = "Windows NDIS driver",
    .init = wpa_driver_ndis_init,
    .deinit = wpa_driver_ndis_deinit,
    .set_wpa = wpa_driver_ndis_set_wpa,
    .scan = wpa_driver_ndis_scan,
    .get_scan_results = wpa_driver_ndis_get_scan_results,
    .get_bssid = wpa_driver_ndis_get_bssid,
    .get_ssid = wpa_driver_ndis_get_ssid,
    .set_key = wpa_driver_ndis_set_key,
    .associate = wpa_driver_ndis_associate,
    .deauthenticate = wpa_driver_ndis_deauthenticate,
    .disassociate = wpa_driver_ndis_disassociate,
    .poll = wpa_driver_ndis_poll,
    .add_pmkid = wpa_driver_ndis_add_pmkid,
    .remove_pmkid = wpa_driver_ndis_remove_pmkid,
    .flush_pmkid = wpa_driver_ndis_flush_pmkid,
    .get_capa = wpa_driver_ndis_get_capa,
    .get_ifname = wpa_driver_ndis_get_ifname,
    .get_mac_addr = wpa_driver_ndis_get_mac_addr,
}
```

Definition at line 1680 of file driver_ndis.c.

6.27 driver_ndis.h File Reference

WPA Supplicant - Windows/NDIS driver interface.

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



6.27.1 Detailed Description

WPA Supplicant - Windows/NDIS driver 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.

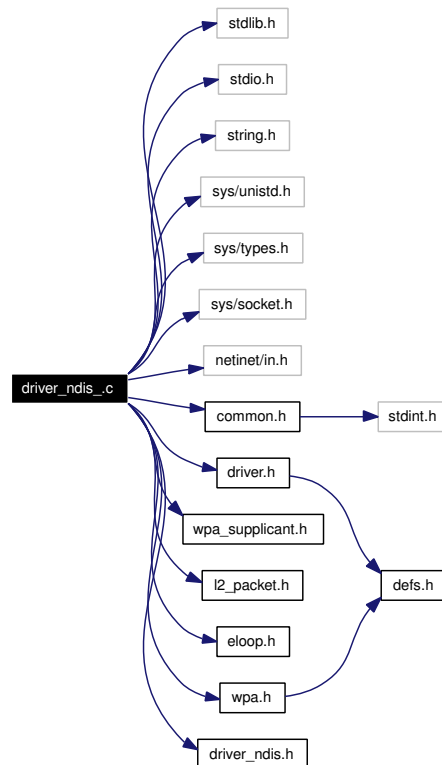
Definition in file [driver_ndis.h](#).

6.28 driver_ndis_.c File Reference

WPA Supplicant - Windows/NDIS driver interface - event processing.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.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` }

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)
- int `wpa_driver_register_event_cb` (struct `wpa_driver_ndis_data` *drv)

6.28.1 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.

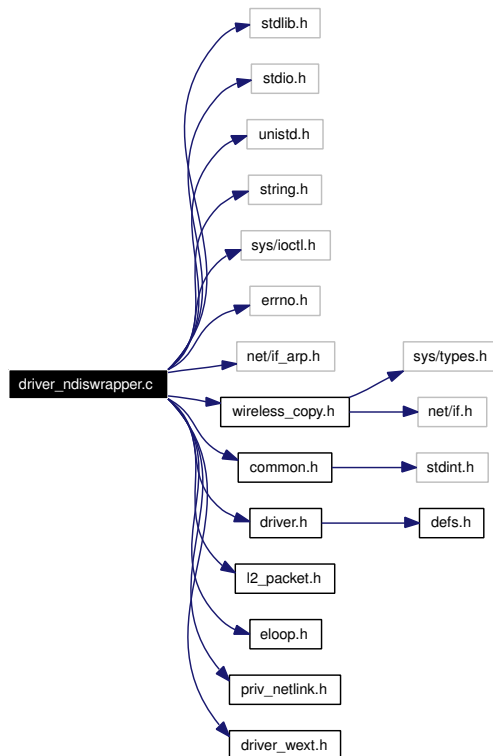
Definition in file [driver_ndis_.c](#).

6.29 driver_ndiswrapper.c File Reference

WPA Supplicant - driver interaction with Linux ndiswrapper.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.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**

6.29.1 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](#).

6.29.2 Variable Documentation

6.29.2.1 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,  
}
```

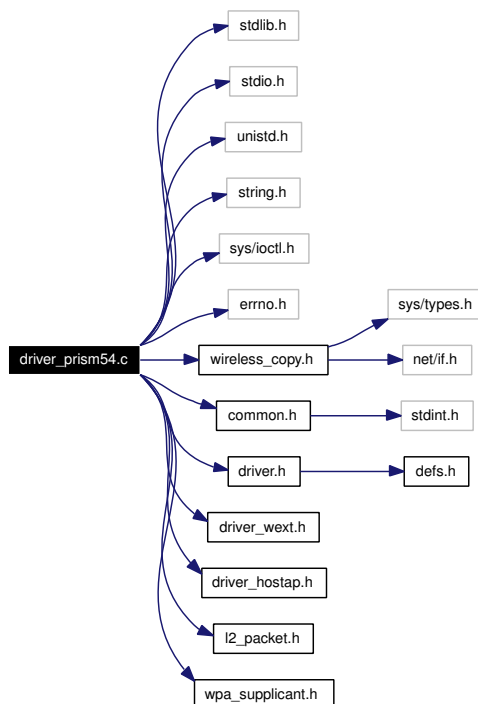
Definition at line 351 of file driver_ndiswrapper.c.

6.30 driver_prism54.c File Reference

WPA Supplicant - driver interaction with Linux Prism54.org driver.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.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**

6.30.1 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](#).

6.30.2 Variable Documentation

6.30.2.1 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,
}
```

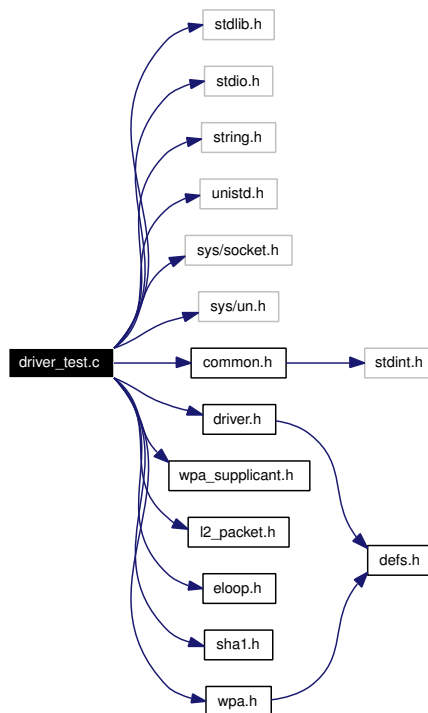
Definition at line 370 of file [driver_prism54.c](#).

6.31 driver_test.c File Reference

WPA Supplicant - testing driver interface.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/un.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 dependency graph for driver_test.c:



Variables

- const struct [wpa_driver_ops](#) `wpa_driver_test_ops`

6.31.1 Detailed Description

WPA Supplicant - testing driver 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 [driver_test.c](#).

6.31.2 Variable Documentation

6.31.2.1 const struct [wpa_driver_ops](#) [wpa_driver_test_ops](#)

Initial value:

```
{
    .name = "test",
    .desc = "wpa_supplicant test driver",
    .init = wpa_driver_test_init,
    .deinit = wpa_driver_test_deinit,
    .set_param = wpa_driver_test_set_param,
    .set_wpa = wpa_driver_test_set_wpa,
    .scan = wpa_driver_test_scan,
    .get_scan_results = wpa_driver_test_get_scan_results,
    .set_key = wpa_driver_test_set_key,
    .associate = wpa_driver_test_associate,
    .get_bssid = wpa_driver_test_get_bssid,
    .get_ssid = wpa_driver_test_get_ssid,
    .get_mac_addr = wpa_driver_test_get_mac_addr,
    .send_eapol = wpa_driver_test_send_eapol,
    .deauthenticate = wpa_driver_test_deauthenticate,
    .disassociate = wpa_driver_test_disassociate,
    .get_capa = wpa_driver_test_get_capa,
}
```

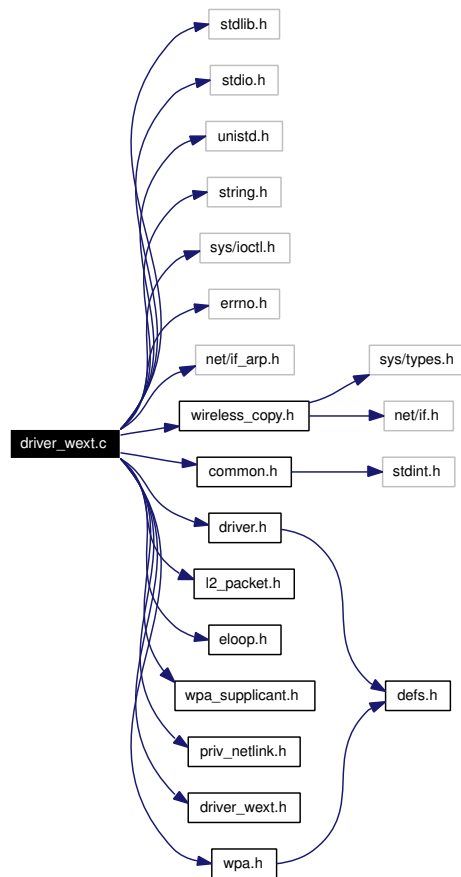
Definition at line 582 of file [driver_test.c](#).

6.32 driver_wext.c File Reference

WPA Supplicant - driver interaction with generic Linux Wireless Extensions.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.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 *loop_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)

Variables

- const struct [wpa_driver_ops](#) [wpa_driver_wext_ops](#)

6.32.1 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](#).

6.32.2 Function Documentation

6.32.2.1 void wpa_driver_wext_deinit (void * *priv*)

Deinitialize WE driver interface.

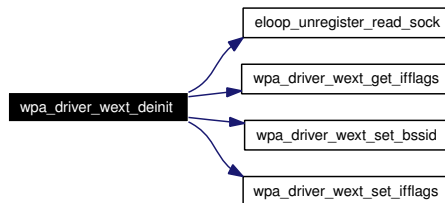
Parameters:

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 827 of file driver_wext.c.

Here is the call graph for this function:



6.32.2.2 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 95 of file driver_wext.c.

6.32.2.3 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 695 of file driver_wext.c.

6.32.2.4 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)

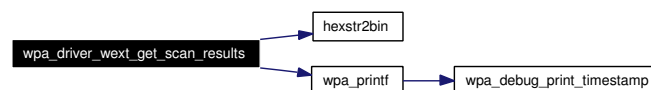
Returns:

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 969 of file driver_wext.c.

Here is the call graph for this function:



6.32.2.5 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 151 of file driver_wext.c.

6.32.2.6 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\(\)](#)

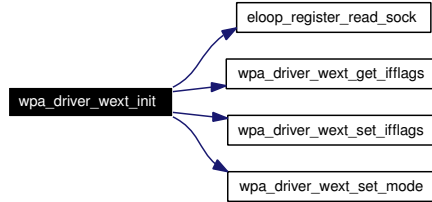
ifname interface name, e.g., wlan0

Returns:

Pointer to private data, NULL on failure

Definition at line 740 of file driver_wext.c.

Here is the call graph for this function:



6.32.2.7 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)

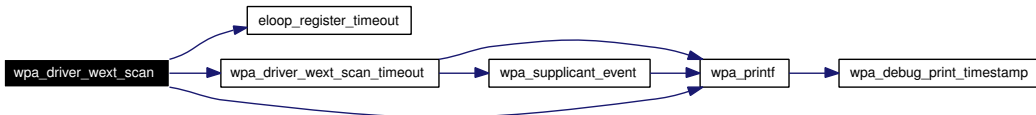
ssid_len Length of the SSID

Returns:

0 on success, -1 on failure

Definition at line 877 of file driver_wext.c.

Here is the call graph for this function:



6.32.2.8 void wpa_driver_wext_scan_timeout (void * eloop_ctx, void * timeout_ctx)

Scan timeout to report scan completion.

Parameters:

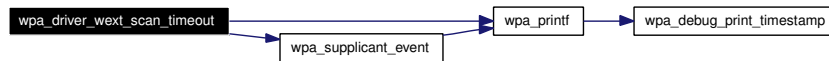
eloop_ctx Unused

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 860 of file driver_wext.c.

Here is the call graph for this function:



6.32.2.9 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 121 of file driver_wext.c.

6.32.2.10 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 237 of file driver_wext.c.

6.32.2.11 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 717 of file driver_wext.c.

6.32.2.12 `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)

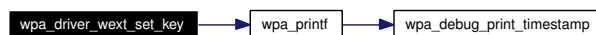
Returns:

- 0 on success, -1 on failure

This function uses SIOCSIWENCODDEEXT by default, but tries to use SIOCSIWENCODDE if the extended ioctl fails when configuring a WEP key.

Definition at line 1410 of file driver_wext.c.

Here is the call graph for this function:



6.32.2.13 `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 1748 of file driver_wext.c.

6.32.2.14 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 190 of file driver_wext.c.

6.32.3 Variable Documentation

6.32.3.1 const struct [wpa_driver_ops](#) wpa_driver_wext_ops

Initial value:

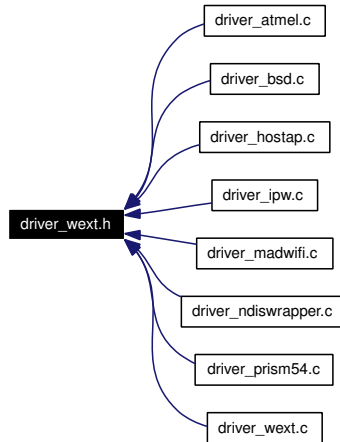
```
{
    .name = "wext",
    .desc = "Linux wireless extensions (generic)",
    .get_bssid = wpa_driver_wext_get_bssid,
    .get_ssid = wpa_driver_wext_get_ssid,
    .set_wpa = wpa_driver_wext_set_wpa,
    .set_key = wpa_driver_wext_set_key,
    .set_countermeasures = wpa_driver_wext_set_countermeasures,
    .set_drop_unencrypted = wpa_driver_wext_set_drop_unencrypted,
    .scan = wpa_driver_wext_scan,
    .get_scan_results = wpa_driver_wext_get_scan_results,
    .deauthenticate = wpa_driver_wext_deauthenticate,
    .disassociate = wpa_driver_wext_disassociate,
    .associate = wpa_driver_wext_associate,
    .set_auth_alg = wpa_driver_wext_set_auth_alg,
    .init = wpa_driver_wext_init,
    .deinit = wpa_driver_wext_deinit,
    .add_pmkid = wpa_driver_wext_add_pmkid,
    .remove_pmkid = wpa_driver_wext_remove_pmkid,
    .flush_pmkid = wpa_driver_wext_flush_pmkid,
    .get_capa = wpa_driver_wext_get_capa,
}
```

Definition at line 1848 of file driver_wext.c.

6.33 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.

6.33.1 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](#).

6.33.2 Function Documentation

6.33.2.1 void [wpa_driver_wext_deinit](#) (void * *priv*)

Deinitialize WE driver interface.

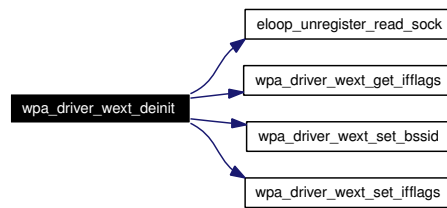
Parameters:

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 827 of file driver_wext.c.

Here is the call graph for this function:



6.33.2.2 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 95 of file driver_wext.c.

6.33.2.3 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 695 of file driver_wext.c.

6.33.2.4 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 969 of file `driver_wext.c`.

Here is the call graph for this function:



6.33.2.5 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 151 of file `driver_wext.c`.

6.33.2.6 void* wpa_driver_wext_init (void * ctx, const char * ifname)

Initialize WE driver interface.

Parameters:

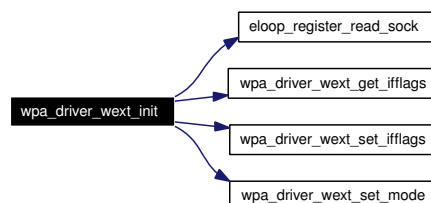
- ctx* context to be used when calling [wpa_supplicant](#) functions, e.g., [wpa_supplicant_event\(\)](#)
- ifname* interface name, e.g., wlan0

Returns:

- Pointer to private data, NULL on failure

Definition at line 740 of file `driver_wext.c`.

Here is the call graph for this function:



6.33.2.7 int wpa_driver_wext_scan (void * priv, const u8 * ssid, size_t ssid_len)

Request the driver to initiate scan.

Parameters:

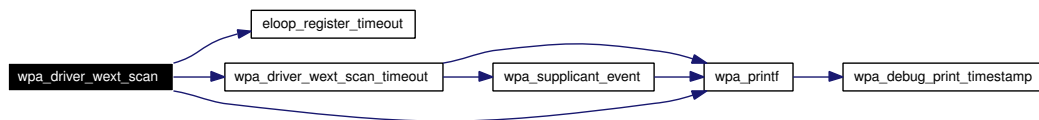
- priv* Pointer to private wext data from [wpa_driver_wext_init\(\)](#)
- ssid* Specific SSID to scan for (ProbeReq) or NULL to scan for all SSIDs (either active scan with broadcast SSID or passive scan)
- ssid_len* Length of the SSID

Returns:

0 on success, -1 on failure

Definition at line 877 of file driver_wext.c.

Here is the call graph for this function:



6.33.2.8 void wpa_driver_wext_scan_timeout (void * eloop_ctx, void * timeout_ctx)

Scan timeout to report scan completion.

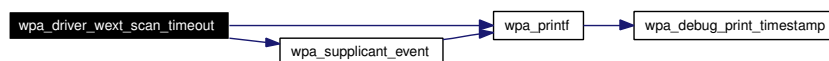
Parameters:

- 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 860 of file driver_wext.c.

Here is the call graph for this function:



6.33.2.9 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 121 of file driver_wext.c.

6.33.2.10 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 237 of file driver_wext.c.

6.33.2.11 int wpa_driver_wext_set_iflags (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 717 of file driver_wext.c.

6.33.2.12 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)

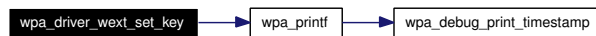
Returns:

0 on success, -1 on failure

This function uses SIOCSIWENCODDEEXT by default, but tries to use SIOCSIWENCODDE if the extended ioctl fails when configuring a WEP key.

Definition at line 1410 of file driver_wext.c.

Here is the call graph for this function:



6.33.2.13 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 1748 of file driver_wext.c.

6.33.2.14 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

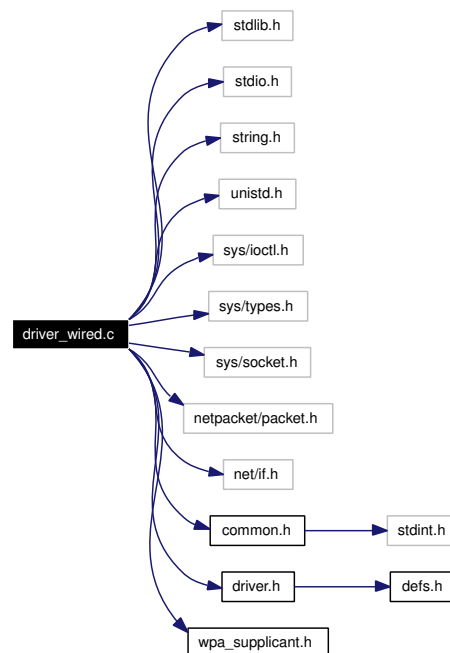
Definition at line 190 of file driver_wext.c.

6.34 driver_wired.c File Reference

WPA Supplicant - wired Ethernet driver interface.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netpacket/packet.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`

6.34.1 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](#).

6.34.2 Variable Documentation

6.34.2.1 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 264 of file driver_wired.c.

6.35 drivers.c File Reference

WPA Supplicant / driver interface list.

```
#include <stdlib.h>
```

Include dependency graph for drivers.c:



Variables

- `wpa_driver_ops * wpa_supplicant_drivers []`

6.35.1 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.

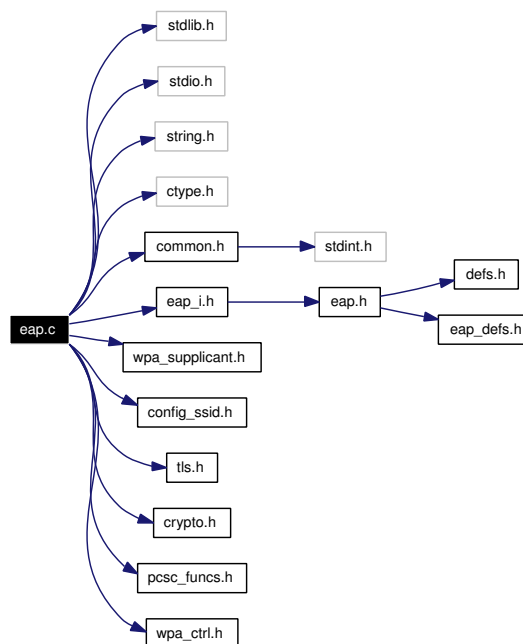
Definition in file [drivers.c](#).

6.36 eap.c File Reference

WPA Supplicant / EAP state machines (RFC 4137).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
#include "tls.h"
#include "crypto.h"
#include "pcsc_funcs.h"
#include "wpa_ctrl.h"
```

Include dependency graph for eap.c:



Defines

- #define **EAP_MAX_AUTH_ROUNDS** 50
- #define **NUM_EAP_METHODS** (sizeof(eap_methods) / sizeof(eap_methods[0]))
- #define **SM_STATE**(machine, state)
- #define **SM_ENTRY**(machine, state)
- #define **SM_ENTER**(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 0)

- #define **SM_ENTER_GLOBAL**(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 1)
- #define **SM_STEP**(machine) static void sm_ ## machine ## _Step(struct [eap_sm](#) *sm)
- #define **SM_STEP_RUN**(machine) sm_ ## machine ## _Step(sm)

Enumerations

- enum **eap_ctrl_req_type** {
TYPE_IDENTITY, **TYPE_PASSWORD**, **TYPE_OTP**, **TYPE_PIN**,
TYPE_NEW_PASSWORD, **TYPE_PASSPHRASE** }

Functions

- const struct [eap_method](#) * [eap_sm_get_eap_methods](#) (int method)
Get EAP method based on type number.
- **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, struct [wpa_ssid](#) *config)

Request identity from user (ctrl_iface).

- void `eap_sm_request_password` (struct `eap_sm` *sm, struct `wpa_ssid` *config)
Request password from user (ctrl_iface).
- void `eap_sm_request_new_password` (struct `eap_sm` *sm, struct `wpa_ssid` *config)
Request new password from user (ctrl_iface).
- void `eap_sm_request_pin` (struct `eap_sm` *sm, struct `wpa_ssid` *config)
Request SIM or smart card PIN from user (ctrl_iface).
- void `eap_sm_request_otp` (struct `eap_sm` *sm, struct `wpa_ssid` *config, const char *msg, size_t msg_len)
Request one time password from user (ctrl_iface).
- void `eap_sm_request_passphrase` (struct `eap_sm` *sm, struct `wpa_ssid` *config)
Request passphrase from user (ctrl_iface).
- void `eap_sm_notify_ctrl_attached` (struct `eap_sm` *sm)
Notification of attached monitor.
- u8 `eap_get_type` (const char *name)
Get EAP type for the given EAP method name.
- const char * `eap_get_name` (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.
- u8 `eap_get_phase2_type` (const char *name)
Get EAP type for the given EAP phase 2 method name.
- u8 * `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.
- 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_sccard_ctx` (struct `eap_sm` *sm, void *ctx)
Notification of smart card context.
- const u8 * `eap_hdr_validate` (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.

6.36.1 Detailed Description

WPA Supplicant / EAP state machines (RFC 4137).

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 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`.

6.36.2 Define Documentation

6.36.2.1 #define SM_ENTRY(machine, state)

Value:

```
if (!global || sm->machine ## _state != machine ## _ ## state) { \
    sm->changed = TRUE; \
    wpa_printf(MSG_DEBUG, "EAP: " #machine " entering state " #state); \
} \
sm->machine ## _state = machine ## _ ## state;
```

Definition at line 161 of file eap.c.

6.36.2.2 #define SM_STATE(machine, state)

Value:

```
static void sm_ ## machine ## _ ## state ## _Enter(struct eap_sm *sm, \
    int global)
```

Definition at line 157 of file eap.c.

6.36.3 Function Documentation

6.36.3.1 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

Definition at line 1741 of file eap.c.

6.36.3.2 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 1934 of file eap.c.

6.36.3.3 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 1818 of file `eap.c`.

6.36.3.4 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 1842 of file `eap.c`.

6.36.3.5 const char* eap_get_name (EapType type)

Get EAP method name for the given EAP type.

Parameters:

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 1614 of file `eap.c`.

6.36.3.6 `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 1633 of file eap.c.

6.36.3.7 `u8 eap_get_phase2_type (const char * name)`

Get EAP type for the given EAP phase 2 method name.

Parameters:

name EAP method name, e.g., MD5

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 1667 of file eap.c.

Here is the call graph for this function:



6.36.3.8 `u8* 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 1686 of file eap.c.

6.36.3.9 u8 eap_get_type (const char * name)

Get EAP type for the given EAP method name.

Parameters:

name EAP method name, e.g., TLS

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 1594 of file eap.c.

6.36.3.10 const u8* eap_hdr_validate (EapType eap_type, const u8 * msg, size_t msglen, size_t * plen)

Validate EAP header.

Parameters:

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

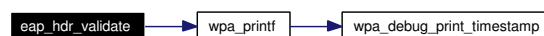
Returns:

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.

Definition at line 1889 of file eap.c.

Here is the call graph for this function:



6.36.3.11 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 1753 of file eap.c.

6.36.3.12 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 1786 of file eap.c.

6.36.3.13 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 1769 of file eap.c.

6.36.3.14 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 1869 of file eap.c.

6.36.3.15 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 1921 of file eap.c.

6.36.3.16 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 1717 of file eap.c.

6.36.3.17 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 1950 of file eap.c.

6.36.3.18 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 1729 of file eap.c.

6.36.3.19 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 1219 of file eap.c.

6.36.3.20 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)

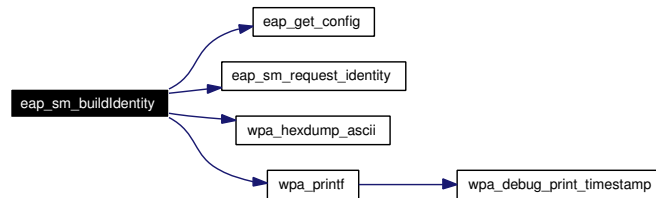
Returns:

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 940 of file eap.c.

Here is the call graph for this function:

**6.36.3.21 void eap_sm_deinit (struct eap_sm * sm)**

Deinitialize and free an EAP state machine.

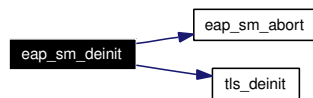
Parameters:

- 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 1177 of file eap.c.

Here is the call graph for this function:

**6.36.3.22 const struct eap_method* eap_sm_get_eap_methods (int method)**

Get EAP method based on type number.

Parameters:

- method* EAP type number

Returns:

Pointer to EAP method of NULL if not found

Definition at line 134 of file eap.c.

6.36.3.23 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 1313 of file eap.c.

Here is the call graph for this function:



6.36.3.24 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

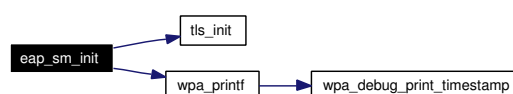
Returns:

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 1138 of file eap.c.

Here is the call graph for this function:



6.36.3.25 void eap_sm_notify_ctrl_attached (struct eap_sm * sm)

Notification of attached monitor.

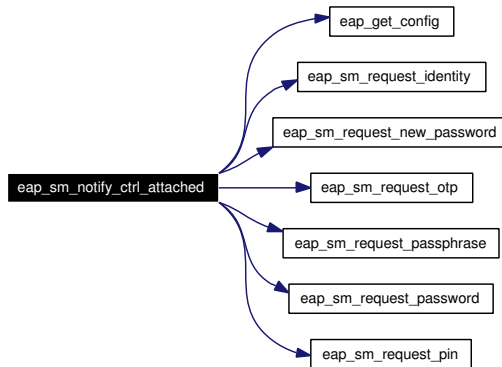
Parameters:

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 1559 of file eap.c.

Here is the call graph for this function:



6.36.3.26 void eap_sm_request_identity (struct eap_sm * sm, struct wpa_ssid * config)

Request identity from user (ctrl_iface).

Parameters:

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

config Pointer to the current network configuration

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 1458 of file eap.c.

6.36.3.27 void eap_sm_request_new_password (struct eap_sm * sm, struct wpa_ssid * config)

Request new password from user (ctrl_iface).

Parameters:

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

config Pointer to the current network configuration

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 1492 of file eap.c.

6.36.3.28 void eap_sm_request_otp (struct eap_sm * sm, struct wpa_ssid * config, 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()`
- config* Pointer to the current network configuration
- 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 1527 of file eap.c.

6.36.3.29 void eap_sm_request_passphrase (struct eap_sm * sm, struct wpa_ssid * config)

Request passphrase from user (ctrl_iface).

Parameters:

- sm* Pointer to EAP state machine allocated with `eap_sm_init()`
- config* Pointer to the current network configuration

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 1545 of file eap.c.

6.36.3.30 void eap_sm_request_password (struct eap_sm * sm, struct wpa_ssid * config)

Request password from user (ctrl_iface).

Parameters:

- sm* Pointer to EAP state machine allocated with `eap_sm_init()`
- config* Pointer to the current network configuration

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 1475 of file eap.c.

6.36.3.31 void eap_sm_request_pin (struct eap_sm * sm, struct wpa_ssid * config)

Request SIM or smart card PIN from user (ctrl_iface).

Parameters:

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

config Pointer to the current network configuration

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 1509 of file eap.c.

6.36.3.32 `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 1198 of file eap.c.

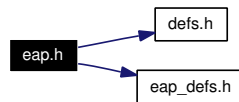
6.37 eap.h File Reference

WPA Supplicant / EAP state machine functions (RFC 4137).

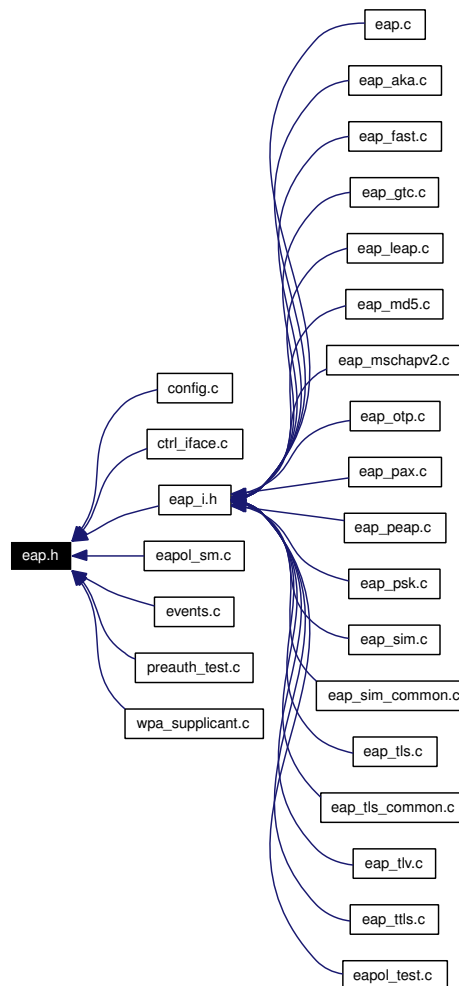
```
#include "defs.h"
```

```
#include "eap_defs.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.
- const struct eap_method * `eap_sm_get_eap_methods` (int method)
Get EAP method based on type number.
- void `eap_sm_request_identity` (struct eap_sm *sm, struct wpa_ssid *config)
Request identity from user (ctrl_iface).
- void `eap_sm_request_password` (struct eap_sm *sm, struct wpa_ssid *config)
Request password from user (ctrl_iface).
- void `eap_sm_request_new_password` (struct eap_sm *sm, struct wpa_ssid *config)
Request new password from user (ctrl_iface).
- void `eap_sm_request_pin` (struct eap_sm *sm, struct wpa_ssid *config)
Request SIM or smart card PIN from user (ctrl_iface).
- void `eap_sm_request_otp` (struct eap_sm *sm, struct wpa_ssid *config, const char *msg, size_t msg_len)
Request one time password from user (ctrl_iface).
- void `eap_sm_request_passphrase` (struct eap_sm *sm, struct wpa_ssid *config)
Request passphrase from user (ctrl_iface).
- void `eap_sm_notify_ctrl_attached` (struct eap_sm *sm)
Notification of attached monitor.

- u8 [eap_get_type](#) (const char *name)
Get EAP type for the given EAP method name.
- const char * [eap_get_name](#) (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.
- u8 [eap_get_phase2_type](#) (const char *name)
Get EAP type for the given EAP phase 2 method name.
- u8 * [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.
- [wpa_ssid](#) * [eap_get_config](#) (struct [eap_sm](#) *sm)
Get current network configuration.
- 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.

6.37.1 Detailed Description

WPA Supplicant / EAP state machine functions (RFC 4137).

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.h](#).

6.37.2 Enumeration Type Documentation

6.37.2.1 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 37 of file eap.h.

6.37.2.2 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 119 of file `eap.h`.

6.37.3 Function Documentation

6.37.3.1 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

Definition at line 1741 of file `eap.c`.

6.37.3.2 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 1818 of file `eap.c`.

6.37.3.3 `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 1842 of file `eap.c`.

6.37.3.4 const char* eap_get_name (EapType type)

Get EAP method name for the given EAP type.

Parameters:

- 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 1614 of file `eap.c`.

6.37.3.5 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 1633 of file `eap.c`.

6.37.3.6 u8 eap_get_phase2_type (const char * name)

Get EAP type for the given EAP phase 2 method name.

Parameters:

- name* EAP method name, e.g., MD5

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 1667 of file eap.c.

Here is the call graph for this function:



6.37.3.7 `u8* 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 1686 of file eap.c.

6.37.3.8 `u8 eap_get_type (const char * name)`

Get EAP type for the given EAP method name.

Parameters:

name EAP method name, e.g., TLS

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 1594 of file eap.c.

6.37.3.9 `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 1753 of file eap.c.

6.37.3.10 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 1786 of file eap.c.

6.37.3.11 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 1769 of file eap.c.

6.37.3.12 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 1869 of file eap.c.

6.37.3.13 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 1717 of file eap.c.

6.37.3.14 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 1950 of file eap.c.

6.37.3.15 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 1729 of file eap.c.

6.37.3.16 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 1219 of file eap.c.

6.37.3.17 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)

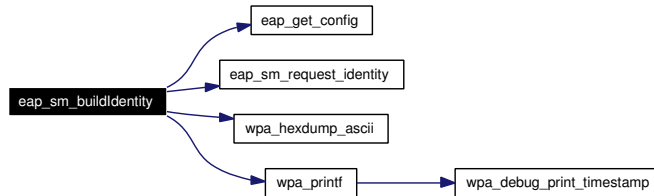
Returns:

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 940 of file eap.c.

Here is the call graph for this function:



6.37.3.18 void eap_sm_deinit (struct eap_sm * sm)

Deinitialize and free an EAP state machine.

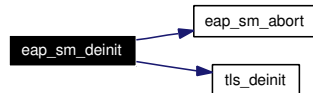
Parameters:

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 1177 of file eap.c.

Here is the call graph for this function:



6.37.3.19 const struct eap_method* eap_sm_get_eap_methods (int method)

Get EAP method based on type number.

Parameters:

method EAP type number

Returns:

Pointer to EAP method or NULL if not found

Definition at line 134 of file eap.c.

6.37.3.20 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 1313 of file eap.c.

Here is the call graph for this function:



6.37.3.21 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

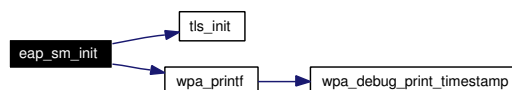
Returns:

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 1138 of file eap.c.

Here is the call graph for this function:



6.37.3.22 void eap_sm_notify_ctrl_attached (struct eap_sm * sm)

Notification of attached monitor.

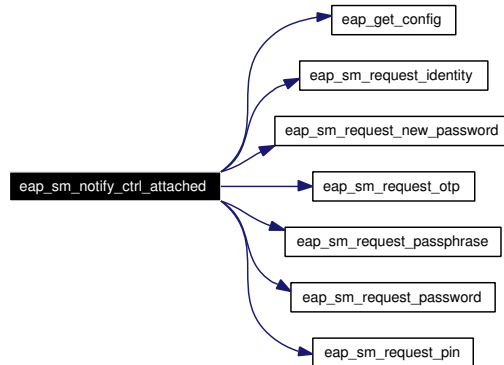
Parameters:

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 1559 of file `eap.c`.

Here is the call graph for this function:



6.37.3.23 void eap_sm_request_identity (struct eap_sm * sm, struct wpa_ssid * config)

Request identity from user (`ctrl_iface`).

Parameters:

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

config Pointer to the current network configuration

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 1458 of file `eap.c`.

6.37.3.24 void eap_sm_request_new_password (struct eap_sm * sm, struct wpa_ssid * config)

Request new password from user (`ctrl_iface`).

Parameters:

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

config Pointer to the current network configuration

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 1492 of file `eap.c`.

6.37.3.25 void `eap_sm_request_otp` (struct `eap_sm` * *sm*, struct `wpa_ssid` * *config*, 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()`

config Pointer to the current network configuration

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 1527 of file eap.c.

6.37.3.26 void `eap_sm_request_passphrase` (struct `eap_sm` * *sm*, struct `wpa_ssid` * *config*)

Request passphrase from user (ctrl_iface).

Parameters:

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

config Pointer to the current network configuration

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 1545 of file eap.c.

6.37.3.27 void `eap_sm_request_password` (struct `eap_sm` * *sm*, struct `wpa_ssid` * *config*)

Request password from user (ctrl_iface).

Parameters:

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

config Pointer to the current network configuration

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 1475 of file eap.c.

6.37.3.28 void `eap_sm_request_pin` (struct `eap_sm` * *sm*, struct `wpa_ssid` * *config*)

Request SIM or smart card PIN from user (ctrl_iface).

Parameters:

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

config Pointer to the current network configuration

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 1509 of file eap.c.

6.37.3.29 `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.

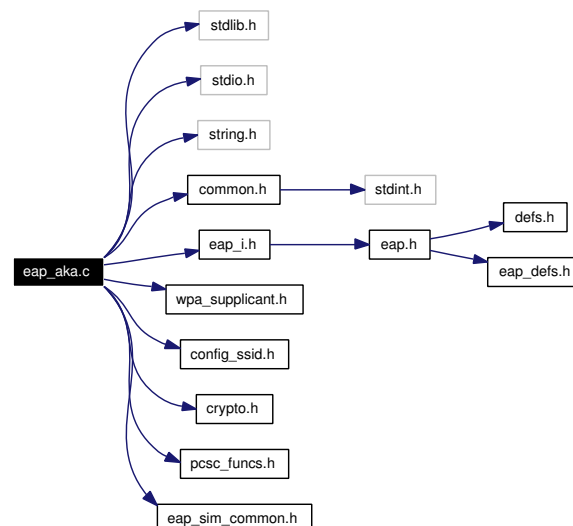
Definition at line 1198 of file eap.c.

6.38 eap_aka.c File Reference

WPA Supplicant / EAP-AKA (draft-arkko-pppext-eap-aka-12.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
#include "crypto.h"
#include "pcsc_funcs.h"
#include "eap_sim_common.h"
```

Include dependency graph for eap_aka.c:



Defines

- #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 **AKA_AUTS_LEN** 14
- #define **RES_MAX_LEN** 16
- #define **IK_LEN** 16

- #define **CK_LEN** 16
- #define **EAP_AKA_MAX_FAST_REAUTHS** 1000
- #define **CLEAR_PSEUDONYM** 0x01
- #define **CLEAR_REAUTH_ID** 0x02
- #define **CLEAR_EAP_ID** 0x04

Variables

- const struct [eap_method](#) **eap_method_aka**

6.38.1 Detailed Description

WPA Supplicant / EAP-AKA (draft-arkko-pppext-eap-aka-12.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_aka.c](#).

6.38.2 Variable Documentation

6.38.2.1 const struct [eap_method](#) eap_method_aka

Initial value:

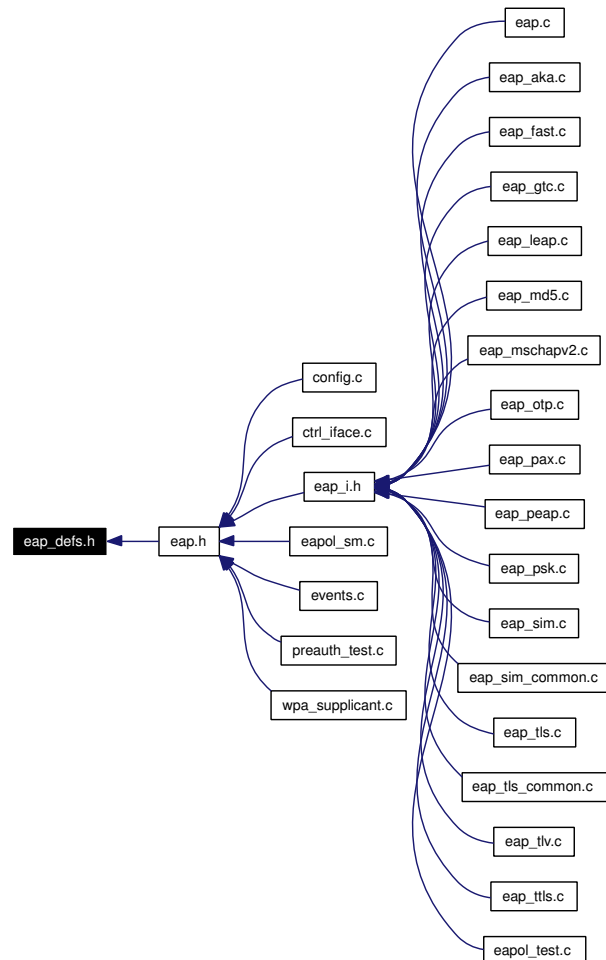
```
{
    .method = EAP_TYPE_AKA,
    .name = "AKA",
    .init = eap_aka_init,
    .deinit = eap_aka_deinit,
    .process = eap_aka_process,
    .isKeyAvailable = eap_aka_isKeyAvailable,
    .getKey = eap_aka_getKey,
    .has_reauth_data = eap_aka_has_reauth_data,
    .deinit_for_reauth = eap_aka_deinit_for_reauth,
    .init_for_reauth = eap_aka_init_for_reauth,
    .get_identity = eap_aka_get_identity,
}
```

Definition at line 918 of file [eap_aka.c](#).

6.39 eap_defs.h File Reference

WPA Supplicant/hostapd / Shared EAP definitions.

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



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_EXPANDED_NAK** = 254, **EAP_TYPE_PSK** = 255 }

Variables

- `eap_hdr` **packed**

6.39.1 Detailed Description

WPA Supplicant/hostapd / Shared EAP definitions.

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.

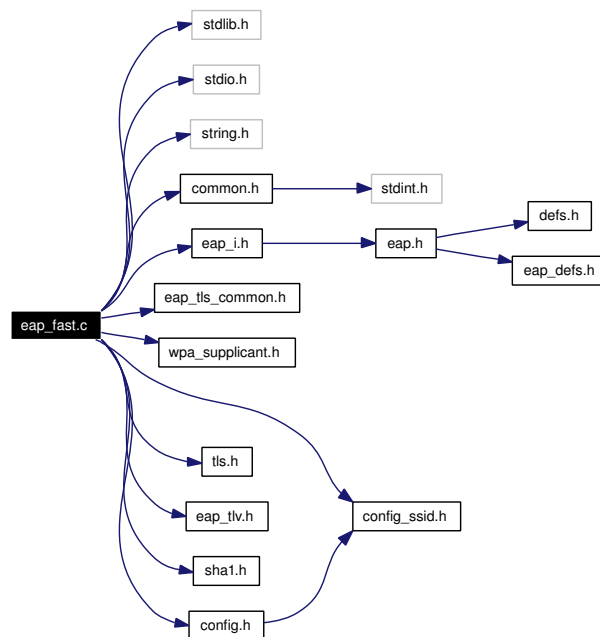
Definition in file [eap_defs.h](#).

6.40 eap_fast.c File Reference

WPA Supplicant / EAP-FAST (draft-cam-winget-eap-fast-00.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "wpa_supplicant.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 **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

Variables

- const struct [eap_method](#) **eap_method_fast**

6.40.1 Detailed Description

WPA Supplicant / EAP-FAST (draft-cam-winget-eap-fast-00.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_fast.c](#).

6.40.2 Variable Documentation

6.40.2.1 const struct [eap_method](#) eap_method_fast

Initial value:

```
{
    .method = EAP_TYPE_FAST,
    .name = "FAST",
    .init = eap_fast_init,
    .deinit = eap_fast_deinit,
    .process = eap_fast_process,
    .isKeyAvailable = eap_fast_isKeyAvailable,
    .getKey = eap_fast_getKey,
    .get_status = eap_fast_get_status,
}
```

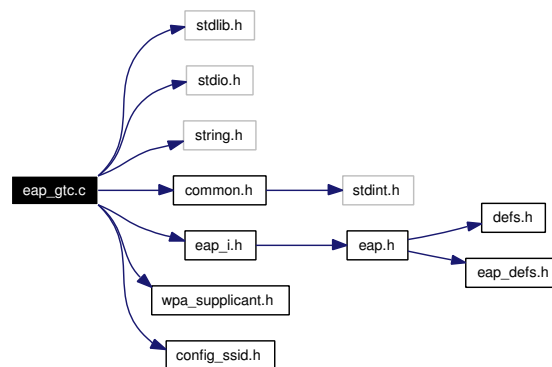
Definition at line 1968 of file [eap_fast.c](#).

6.41 eap_gtc.c File Reference

WPA Supplicant / EAP-GTC (RFC 2284).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
```

Include dependency graph for eap_gtc.c:



Variables

- const struct [eap_method](#) `eap_method_gtc`

6.41.1 Detailed Description

WPA Supplicant / EAP-GTC (RFC 2284).

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_gtc.c](#).

6.41.2 Variable Documentation

6.41.2.1 const struct [eap_method](#) eap_method_gtc

Initial value:

```
{
    .method = EAP_TYPE_GTC,
    .name = "GTC",
    .init = eap_gtc_init,
    .deinit = eap_gtc_deinit,
    .process = eap_gtc_process,
}
```

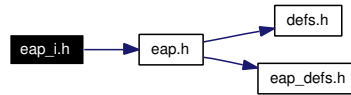
Definition at line 155 of file eap_gtc.c.

6.42 eap_i.h File Reference

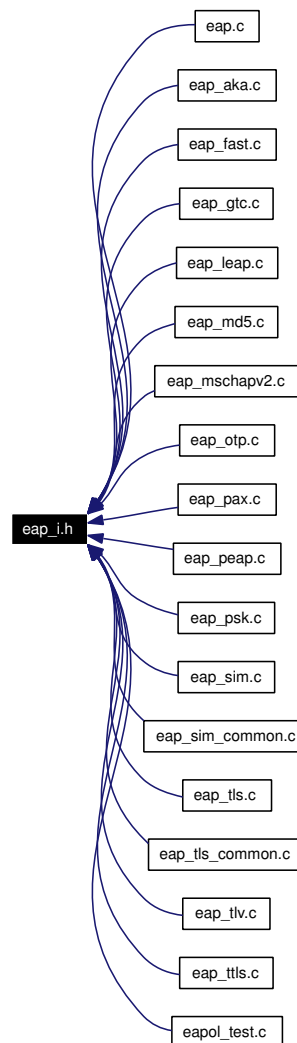
WPA Supplicant / EAP state machines internal structures (RFC 4137).

```
#include "eap.h"
```

Include dependency graph for eap_i.h:



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



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** (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.

6.42.1 Detailed Description

WPA Supplicant / EAP state machines internal structures (RFC 4137).

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_i.h](#).

6.42.2 Function Documentation

6.42.2.1 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 1934 of file [eap.c](#).

6.42.2.2 `const u8* eap_hdr_validate (EapType eap_type, const u8 * msg, size_t msglen, size_t * plen)`

Validate EAP header.

Parameters:

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.

Definition at line 1889 of file `eap.c`.

Here is the call graph for this function:



6.42.2.3 `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.

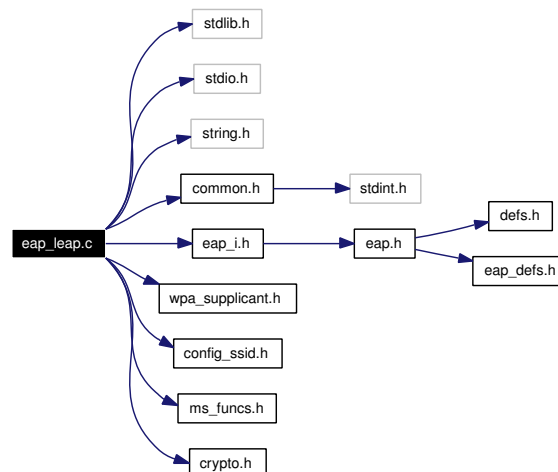
Definition at line 1921 of file `eap.c`.

6.43 eap_leap.c File Reference

WPA Supplicant / EAP-LEAP.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.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

Variables

- const struct [eap_method](#) **eap_method_leap**

6.43.1 Detailed Description

WPA Supplicant / EAP-LEAP.

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_leap.c](#).

6.43.2 Variable Documentation**6.43.2.1 const struct [eap_method](#) eap_method_leap****Initial value:**

```
{
    .method = EAP_TYPE_LEAP,
    .name = "LEAP",
    .init = eap_leap_init,
    .deinit = eap_leap_deinit,
    .process = eap_leap_process,
    .isKeyAvailable = eap_leap_isKeyAvailable,
    .getKey = eap_leap_getKey,
}
```

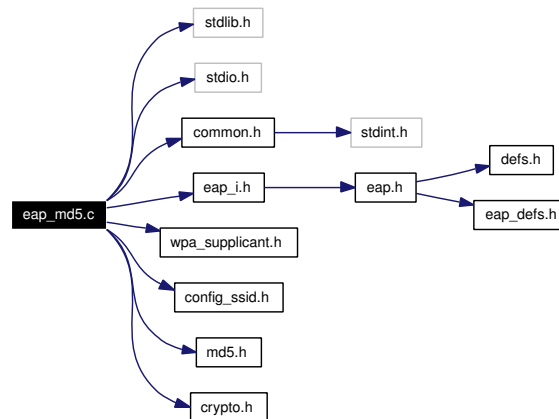
Definition at line 381 of file [eap_leap.c](#).

6.44 eap_md5.c File Reference

WPA Supplicant / EAP-MD5.

```
#include <stdlib.h>
#include <stdio.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
#include "md5.h"
#include "crypto.h"
```

Include dependency graph for eap_md5.c:



Variables

- const struct [eap_method](#) `eap_method_md5`

6.44.1 Detailed Description

WPA Supplicant / EAP-MD5.

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_md5.c](#).

6.44.2 Variable Documentation

6.44.2.1 const struct [eap_method](#) eap_method_md5

Initial value:

```
{
    .method = EAP_TYPE_MD5,
    .name = "MD5",
    .init = eap_md5_init,
    .deinit = eap_md5_deinit,
    .process = eap_md5_process,
}
```

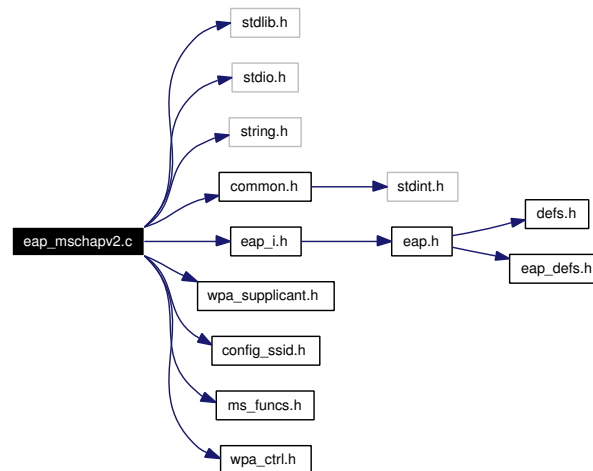
Definition at line 111 of file eap_md5.c.

6.45 eap_mschapv2.c File Reference

WPA Supplicant / EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
#include "ms_funcs.h"
#include "wpa_ctrl.h"
```

Include dependency graph for eap_mschapv2.c:



Defines

- #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 **MSCHAPV2_RESP_LEN** 49
- #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

Variables

- eap_mschapv2_hdr **packed**
- const struct [eap_method](#) **eap_method_mschapv2**

6.45.1 Detailed Description

WPA Supplicant / EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.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_mschapv2.c](#).

6.45.2 Variable Documentation

6.45.2.1 const struct [eap_method](#) eap_method_mschapv2

Initial value:

```
{
    .method = EAP_TYPE_MSCHAPV2,
    .name = "MSCHAPV2",
    .init = eap_mschapv2_init,
    .deinit = eap_mschapv2_deinit,
    .process = eap_mschapv2_process,
    .isKeyAvailable = eap_mschapv2_isKeyAvailable,
    .getKey = eap_mschapv2_getKey,
}
```

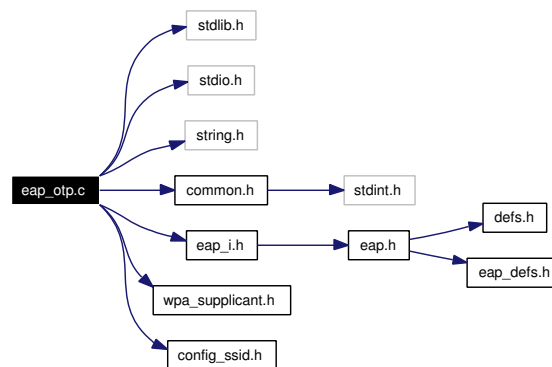
Definition at line 754 of file [eap_mschapv2.c](#).

6.46 eap_otp.c File Reference

WPA Supplicant / EAP-OTP (RFC 3748).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
```

Include dependency graph for eap_otp.c:



Variables

- const struct [eap_method](#) `eap_method_otp`

6.46.1 Detailed Description

WPA Supplicant / EAP-OTP (RFC 3748).

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_otp.c](#).

6.46.2 Variable Documentation

6.46.2.1 const struct [eap_method](#) eap_method_otp

Initial value:

```
{
    .method = EAP_TYPE_OTP,
    .name = "OTP",
    .init = eap_otp_init,
    .deinit = eap_otp_deinit,
    .process = eap_otp_process,
}
```

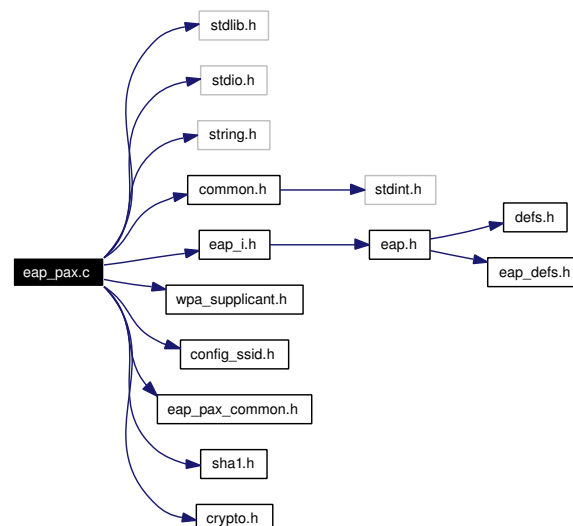
Definition at line 105 of file eap_otp.c.

6.47 eap_pax.c File Reference

WPA Supplicant / EAP-PAX (draft-clancy-eap-pax-04.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
#include "eap_pax_common.h"
#include "sha1.h"
#include "crypto.h"
```

Include dependency graph for eap_pax.c:



Variables

- const struct [eap_method](#) `eap_method_pax`

6.47.1 Detailed Description

WPA Supplicant / EAP-PAX (draft-clancy-eap-pax-04.txt).

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.c](#).

6.47.2 Variable Documentation

6.47.2.1 const struct [eap_method](#) eap_method_pax

Initial value:

```
{
    .method = EAP_TYPE_PAX,
    .name = "PAX",
    .init = eap_pax_init,
    .deinit = eap_pax_deinit,
    .process = eap_pax_process,
    .isKeyAvailable = eap_pax_isKeyAvailable,
    .getKey = eap_pax_getKey,
}
```

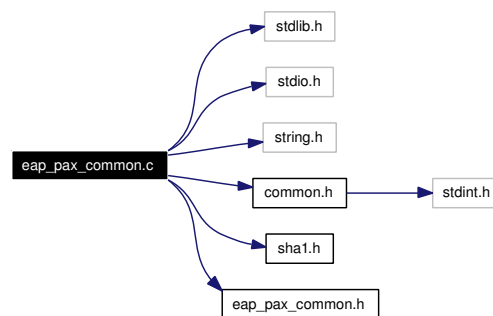
Definition at line 502 of file eap_pax.c.

6.48 eap_pax_common.c File Reference

WPA Supplicant / EAP-PAX shared routines.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.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.

6.48.1 Detailed Description

WPA Supplicant / 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](#).

6.48.2 Function Documentation

6.48.2.1 `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

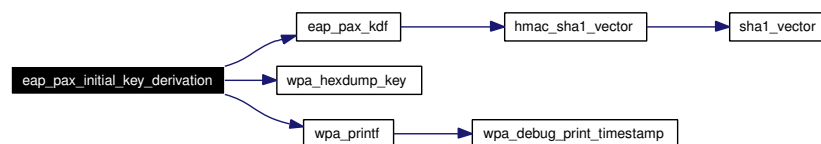
ick Buffer for the derived Integrity Check Key

Returns:

0 on success, -1 on failure

Definition at line 138 of file [eap_pax_common.c](#).

Here is the call graph for this function:



6.48.2.2 `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

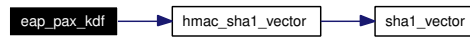
Returns:

0 on success, -1 failed

draft-clancy-eap-pax-04.txt, chap. 2.5: PAX-KDF-W(X, Y, Z)

Definition at line 40 of file eap_pax_common.c.

Here is the call graph for this function:



6.48.2.3 `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)

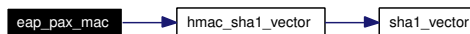
Returns:

0 on success, -1 on failure

Wrapper function to calculate EAP-PAX MAC.

Definition at line 97 of file eap_pax_common.c.

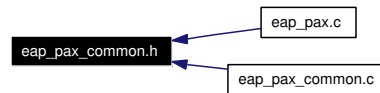
Here is the call graph for this function:



6.49 eap_pax_common.h File Reference

WPA Supplicant / 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_MAC_HMAC_SHA1_128** 0x01
- #define **EAP_PAX_MAC_AES_CBC_MAC_128** 0x02
- #define **EAP_PAX_DH_GROUP_NONE** 0x00
- #define **EAP_PAX_DH_GROUP_3072_MODP** 0x01
- #define **EAP_PAX_PUBLIC_KEY_NONE** 0x00
- #define **EAP_PAX_PUBLIC_KEY_RSA_OAEP_2048** 0x01
- #define **EAP_PAX_RAND_LEN** 32
- #define **EAP_PAX_MSK_LEN** 64
- #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` **packed**

6.49.1 Detailed Description

WPA Supplicant / 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.h](#).

6.49.2 Function Documentation

6.49.2.1 `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

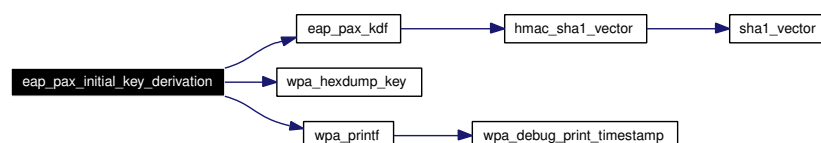
ick Buffer for the derived Integrity Check Key

Returns:

0 on success, -1 on failure

Definition at line 138 of file `eap_pax_common.c`.

Here is the call graph for this function:



6.49.2.2 `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

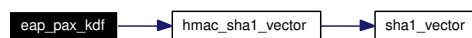
Returns:

0 on success, -1 failed

draft-clancy-eap-pax-04.txt, chap. 2.5: PAX-KDF-W(X, Y, Z)

Definition at line 40 of file eap_pax_common.c.

Here is the call graph for this function:



6.49.2.3 `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)

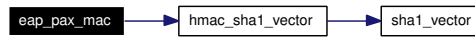
Returns:

0 on success, -1 on failure

Wrapper function to calculate EAP-PAX MAC.

Definition at line 97 of file eap_pax_common.c.

Here is the call graph for this function:

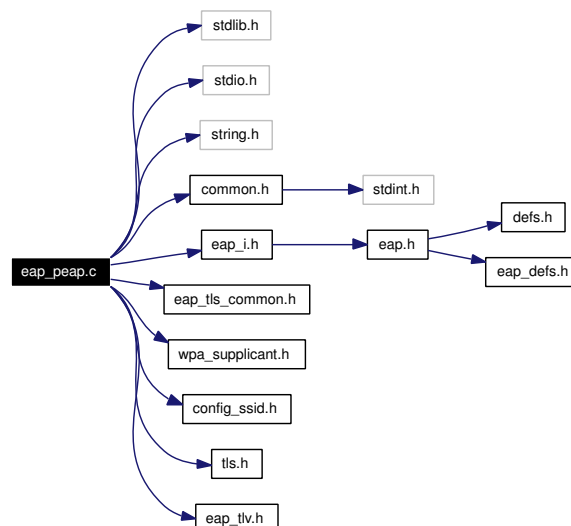


6.50 eap_peap.c File Reference

WPA Supplicant / EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-07.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "wpa_supplicant.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

Variables

- const struct [eap_method](#) **eap_method_peap**

6.50.1 Detailed Description

WPA Supplicant / EAP-PEAP (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_peap.c](#).

6.50.2 Variable Documentation

6.50.2.1 const struct [eap_method](#) eap_method_peap

Initial value:

```
{
    .method = EAP_TYPE_PEAP,
    .name = "PEAP",
    .init = eap_peap_init,
    .deinit = eap_peap_deinit,
    .process = eap_peap_process,
    .isKeyAvailable = eap_peap_isKeyAvailable,
    .getKey = eap_peap_getKey,
    .get_status = eap_peap_get_status,
    .has_reauth_data = eap_peap_has_reauth_data,
    .deinit_for_reauth = eap_peap_deinit_for_reauth,
    .init_for_reauth = eap_peap_init_for_reauth,
}
```

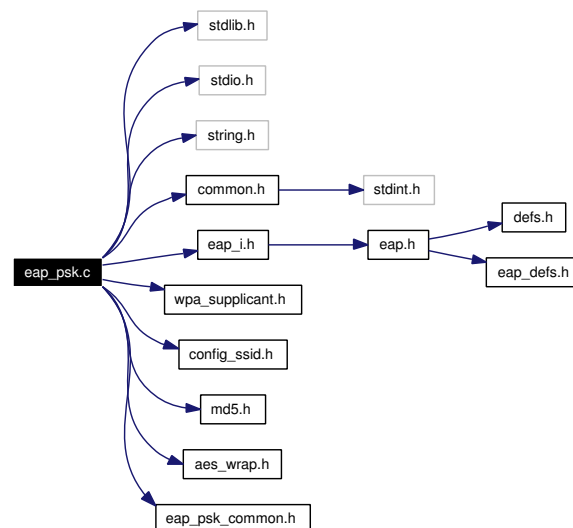
Definition at line 810 of file [eap_peap.c](#).

6.51 eap_psk.c File Reference

WPA Supplicant / EAP-PSK (draft-bersani-eap-psk-09.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
#include "md5.h"
#include "aes_wrap.h"
#include "eap_psk_common.h"
```

Include dependency graph for eap_psk.c:



Variables

- const struct [eap_method](#) `eap_method_psk`

6.51.1 Detailed Description

WPA Supplicant / EAP-PSK (draft-bersani-eap-psk-09.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.

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](#).

6.51.2 Variable Documentation

6.51.2.1 const struct [eap_method](#) eap_method_psk

Initial value:

```
{
    .method = EAP_TYPE_PSK,
    .name = "PSK",
    .init = eap_psk_init,
    .deinit = eap_psk_deinit,
    .process = eap_psk_process,
    .isKeyAvailable = eap_psk_isKeyAvailable,
    .getKey = eap_psk_getKey,
}
```

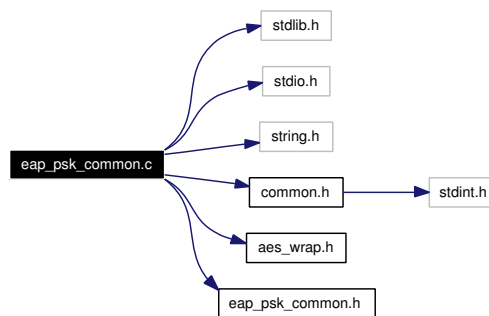
Definition at line 431 of file eap_psk.c.

6.52 eap_psk_common.c File Reference

WPA Supplicant / EAP-PSK shared routines.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "aes_wrap.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)

6.52.1 Detailed Description

WPA Supplicant / EAP-PSK shared routines.

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.

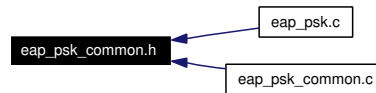
See README and COPYING for more details.

Definition in file [eap_psk_common.c](#).

6.53 eap_psk_common.h File Reference

WPA Supplicant / 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_MSK_LEN** 64
- #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

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)

Variables

- eap_psk_hdr **packed**

6.53.1 Detailed Description

WPA Supplicant / EAP-PSK shared routines.

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.

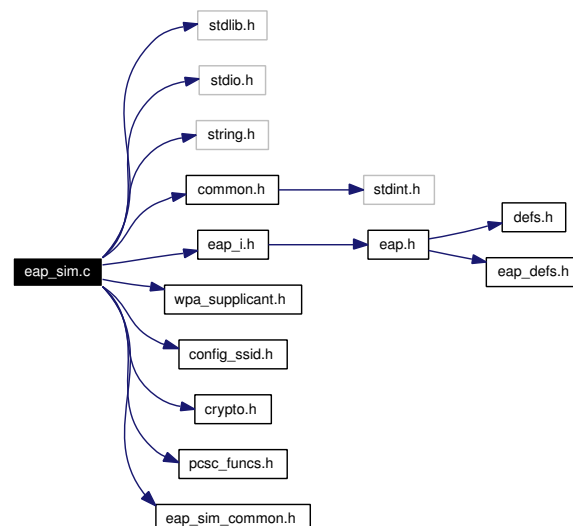
Definition in file [eap_psk_common.h](#).

6.54 eap_sim.c File Reference

WPA Supplicant / EAP-SIM (draft-haverinen-pppext-eap-sim-13.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "wpa_supplicant.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 **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 **KC_LEN** 8

- #define **SRES_LEN** 4
- #define **EAP_SIM_MAX_FAST_REAUTHS** 1000
- #define **CLEAR_PSEUDONYM** 0x01
- #define **CLEAR_REAUTH_ID** 0x02
- #define **CLEAR_EAP_ID** 0x04

Variables

- const struct [eap_method](#) **eap_method_sim**

6.54.1 Detailed Description

WPA Supplicant / EAP-SIM (draft-haverinen-pppext-eap-sim-13.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_sim.c](#).

6.54.2 Variable Documentation

6.54.2.1 const struct [eap_method](#) eap_method_sim

Initial value:

```
{
    .method = EAP_TYPE_SIM,
    .name = "SIM",
    .init = eap_sim_init,
    .deinit = eap_sim_deinit,
    .process = eap_sim_process,
    .isKeyAvailable = eap_sim_isKeyAvailable,
    .getKey = eap_sim_getKey,
    .has_reauth_data = eap_sim_has_reauth_data,
    .deinit_for_reauth = eap_sim_deinit_for_reauth,
    .init_for_reauth = eap_sim_init_for_reauth,
    .get_identity = eap_sim_get_identity,
}
```

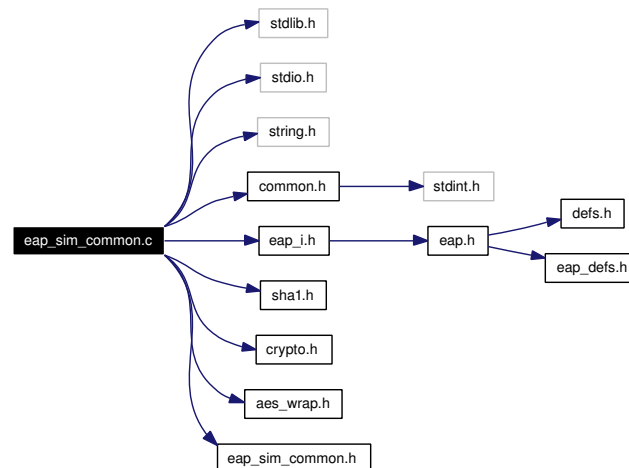
Definition at line 1006 of file [eap_sim.c](#).

6.55 eap_sim_common.c File Reference

WPA Supplicant / EAP-SIM/AKA shared routines.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.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:



Defines

- #define **EAP_SIM_INIT_LEN** 128

Functions

- void **eap_sim_derive_keys** (const u8 *mk, u8 *k_encr, u8 *k_aut, u8 *msk)
- void **eap_sim_derive_keys_reauth** (u16 _counter, const u8 *identity, size_t identity_len, const u8 *nonce_s, const u8 *mk, u8 *msk)
- 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)

6.55.1 Detailed Description

WPA Supplicant / EAP-SIM/AKA shared routines.

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.

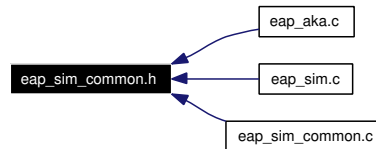
See README and COPYING for more details.

Definition in file [eap_sim_common.c](#).

6.56 eap_sim_common.h File Reference

WPA Supplicant / 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 **GSM_RAND_LEN** 16
- #define **AKA_RAND_LEN** 16
- #define **AKA_AUTN_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_keys** (const u8 *mk, u8 *k_encr, u8 *k_aut, u8 *msk)
- void **eap_sim_derive_keys_reauth** (u16 _counter, const u8 *identity, size_t identity_len, const u8 *nonce_s, const u8 *mk, u8 *msk)
- 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)

6.56.1 Detailed Description

WPA Supplicant / EAP-SIM/AKA shared routines.

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.

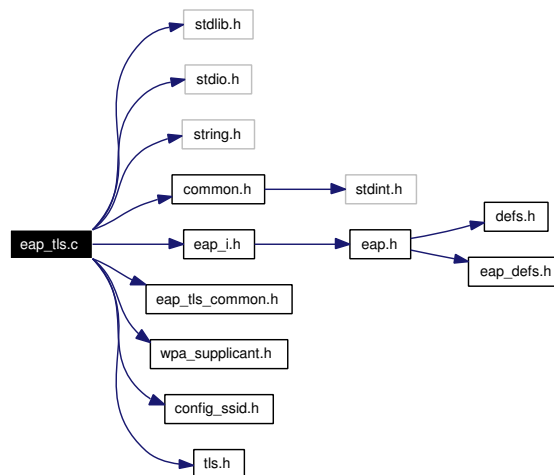
Definition in file [eap_sim_common.h](#).

6.57 eap_tls.c File Reference

WPA Supplicant / EAP-TLS (RFC 2716).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
#include "tls.h"
```

Include dependency graph for eap_tls.c:



Variables

- const struct [eap_method](#) `eap_method_tls`

6.57.1 Detailed Description

WPA Supplicant / EAP-TLS (RFC 2716).

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.c](#).

6.57.2 Variable Documentation

6.57.2.1 const struct [eap_method](#) eap_method_tls

Initial value:

```
{
    .method = EAP_TYPE_TLS,
    .name = "TLS",
    .init = eap_tls_init,
    .deinit = eap_tls_deinit,
    .process = eap_tls_process,
    .isKeyAvailable = eap_tls_isKeyAvailable,
    .getKey = eap_tls_getKey,
    .get_status = eap_tls_get_status,
    .has_reauth_data = eap_tls_has_reauth_data,
    .deinit_for_reauth = eap_tls_deinit_for_reauth,
    .init_for_reauth = eap_tls_init_for_reauth,
}
```

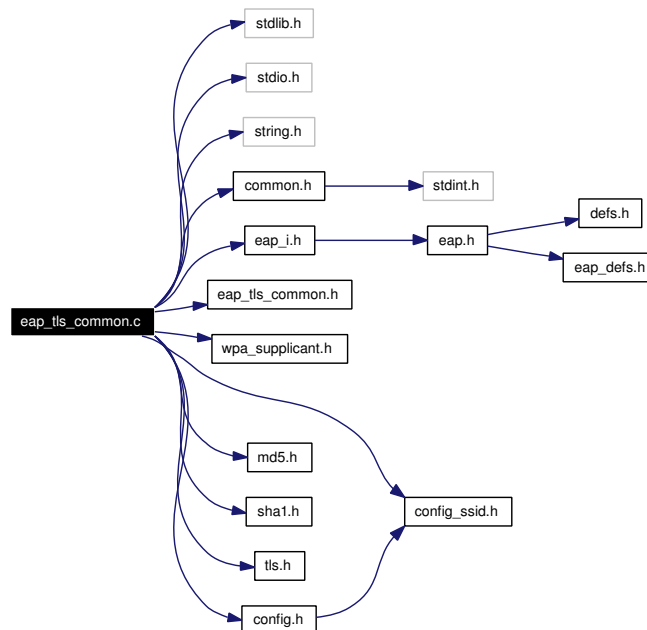
Definition at line 225 of file [eap_tls.c](#).

6.58 eap_tls_common.c File Reference

WPA Supplicant / EAP-TLS/PEAP/TTLS/FAST common functions.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "wpa_supplicant.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)
- void **eap_tls_ssl_deinit** (struct [eap_sm](#) *sm, struct [eap_ssl_data](#) *data)
- u8 * **eap_tls_derive_key** (struct [eap_sm](#) *sm, struct [eap_ssl_data](#) *data, char *label, size_t len)
- 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, int eap_type, int peap_version, u8 id, const u8 *in_data, size_t in_len, u8 **out_data, size_t *out_len)
- u8 * **eap_tls_build_ack** (struct eap_ssl_data *data, size_t *respDataLen, u8 id, int eap_type, int peap_version)
- int **eap_tls_reauth_init** (struct [eap_sm](#) *sm, struct eap_ssl_data *data)
- int **eap_tls_status** (struct [eap_sm](#) *sm, struct eap_ssl_data *data, char *buf, size_t buflen, int verbose)
- 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)

6.58.1 Detailed Description

WPA Supplicant / 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.c](#).

6.58.2 Function Documentation

6.58.2.1 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

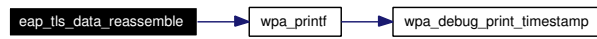
Returns:

Pointer to output data or NULL on error

This function reassembles TLS fragments.

Definition at line 228 of file [eap_tls_common.c](#).

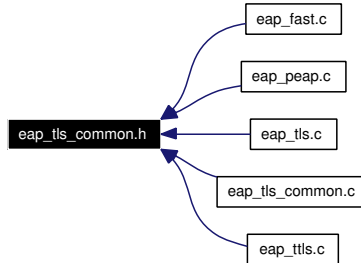
Here is the call graph for this function:



6.59 eap_tls_common.h File Reference

WPA Supplicant / 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)
- void **eap_tls_ssl_deinit** (struct [eap_sm](#) *sm, struct [eap_ssl_data](#) *data)
- u8 * **eap_tls_derive_key** (struct [eap_sm](#) *sm, struct [eap_ssl_data](#) *data, char *label, size_t len)
- 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, int eap_type, int peap_version, u8 id, const u8 *in_data, size_t in_len, u8 **out_data, size_t *out_len)
- u8 * **eap_tls_build_ack** (struct [eap_ssl_data](#) *data, size_t *respDataLen, u8 id, int eap_type, int peap_version)
- int **eap_tls_reauth_init** (struct [eap_sm](#) *sm, struct [eap_ssl_data](#) *data)
- int **eap_tls_status** (struct [eap_sm](#) *sm, struct [eap_ssl_data](#) *data, char *buf, size_t buflen, int verbose)
- 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)

6.59.1 Detailed Description

WPA Supplicant / 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](#).

6.59.2 Function Documentation

6.59.2.1 `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

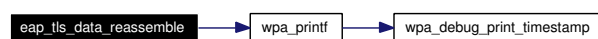
Returns:

Pointer to output data or NULL on error

This function reassembles TLS fragments.

Definition at line 228 of file [eap_tls_common.c](#).

Here is the call graph for this function:

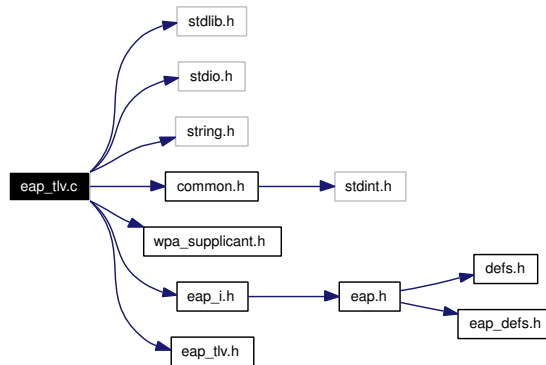


6.60 eap_tlv.c File Reference

WPA Supplicant / EAP-TLV (draft-josefsson-pppext-eap-tls-eap-07.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "wpa_supplicant.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.

6.60.1 Detailed Description

WPA Supplicant / 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.c](#).

6.60.2 Function Documentation

6.60.2.1 `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 function builds an EAP-TLV NAK message. The caller is responsible for freeing the returned buffer.

Definition at line 37 of file [eap_tlv.c](#).

6.60.2.2 `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 function builds an EAP-TLV Result message. The caller is responsible for freeing the returned buffer.

Definition at line 80 of file [eap_tlv.c](#).

6.60.2.3 `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.

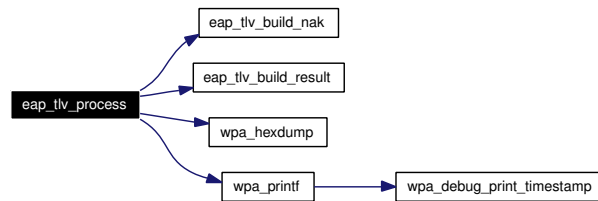
resp_len Buffer for returning the response length

Returns:

0 on success, -1 on failure

Definition at line 122 of file eap_tlv.c.

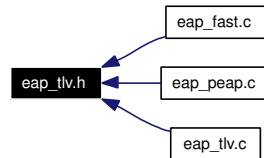
Here is the call graph for this function:



6.61 eap_tlv.h File Reference

WPA Supplicant / 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_nak_tlv](#) **packed**

6.61.1 Detailed Description

WPA Supplicant / 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](#).

6.61.2 Function Documentation

6.61.2.1 `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 function builds an EAP-TLV NAK message. The caller is responsible for freeing the returned buffer.

Definition at line 37 of file [eap_tlv.c](#).

6.61.2.2 `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 function builds an EAP-TLV Result message. The caller is responsible for freeing the returned buffer.

Definition at line 80 of file [eap_tlv.c](#).

6.61.2.3 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.

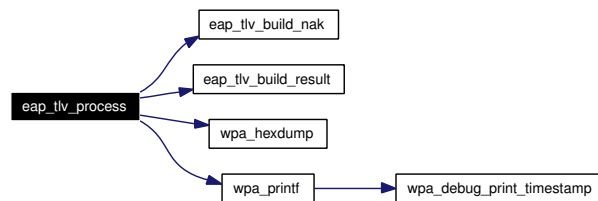
resp_len Buffer for returning the response length

Returns:

0 on success, -1 on failure

Definition at line 122 of file `eap_tlv.c`.

Here is the call graph for this function:

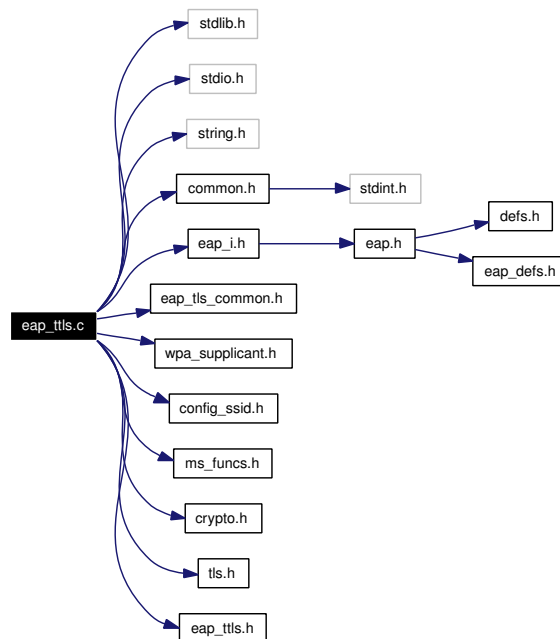


6.62 eap_ttls.c File Reference

WPA Supplicant / EAP-TTLS (draft-ietf-pppext-eap-ttls-03.txt).

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "wpa_supplicant.h"
#include "config_ssid.h"
#include "ms_funcs.h"
#include "crypto.h"
#include "tls.h"
#include "eap_ttls.h"
```

Include dependency graph for eap_ttls.c:



Variables

- const struct [eap_method](#) `eap_method_ttls`

6.62.1 Detailed Description

WPA Supplicant / EAP-TTLS (draft-ietf-pppext-eap-ttls-03.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_ttls.c](#).

6.62.2 Variable Documentation**6.62.2.1 const struct [eap_method](#) eap_method_ttls****Initial value:**

```
{
    .method = EAP_TYPE_TTLS,
    .name = "TTLS",
    .init = eap_ttls_init,
    .deinit = eap_ttls_deinit,
    .process = eap_ttls_process,
    .isKeyAvailable = eap_ttls_isKeyAvailable,
    .getKey = eap_ttls_getKey,
    .get_status = eap_ttls_get_status,
    .has_reauth_data = eap_ttls_has_reauth_data,
    .deinit_for_reauth = eap_ttls_deinit_for_reauth,
    .init_for_reauth = eap_ttls_init_for_reauth,
}
```

Definition at line 1348 of file [eap_ttls.c](#).

6.63 eap_ttls.h File Reference

WPA Supplicant / 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

6.63.1 Detailed Description

WPA Supplicant / EAP-TTLS (draft-ietf-pppext-eap-ttls-03.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_ttls.h](#).

6.63.2 Define Documentation

6.63.2.1 #define AVP_PAD(start, pos)

Value:

```
do { \
    int pad; \
    pad = (4 - (((pos) - (start)) & 3)) & 3; \
    memset((pos), 0, pad); \
    pos += pad; \
} while(0)
```

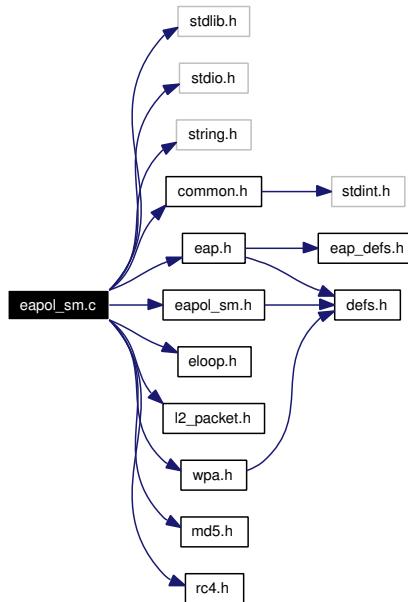
Definition at line 38 of file eap_tls.h.

6.64 eapol_sm.c File Reference

WPA Supplicant / EAPOL state machines.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.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 dependency graph for eapol_sm.c:



Defines

- #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 **SM_STATE**(machine, state)
- #define **SM_ENTRY**(machine, state)
- #define **SM_ENTER**(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 0)

- #define **SM_ENTER_GLOBAL**(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 1)
- #define **SM_STEP**(machine) static void sm_ ## machine ## _Step(struct eapol_sm *sm)
- #define **SM_STEP_RUN**(machine) sm_ ## machine ## _Step(sm)
- #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.
- `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` **packed**

6.64.1 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](#).

6.64.2 Define Documentation

6.64.2.1 `#define SM_ENTRY(machine, state)`

Value:

```
if (!global || sm->machine ## _state != machine ## _ ## state) { \
    sm->changed = TRUE; \
    wpa_printf(MSG_DEBUG, "EAPOL: " #machine " entering state " #state); \
} \
sm->machine ## _state = machine ## _ ## state;
```

Definition at line 195 of file `eapol_sm.c`.

6.64.2.2 `#define SM_STATE(machine, state)`

Value:

```
static void sm_ ## machine ## _ ## state ## _Enter(struct eapol_sm *sm, \
    int global)
```

Definition at line 191 of file `eapol_sm.c`.

6.64.3 Function Documentation

6.64.3.1 `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 962 of file eapol_sm.c.

6.64.3.2 void eapol_sm_deinit (struct eapol_sm * sm)

Deinitialize EAPOL state machine.

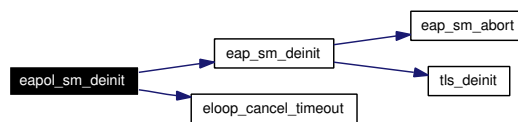
Parameters:

sm Pointer to EAPOL state machine allocated with [eapol_sm_init\(\)](#)

Deinitialize and free EAPOL state machine.

Definition at line 1742 of file eapol_sm.c.

Here is the call graph for this function:



6.64.3.3 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

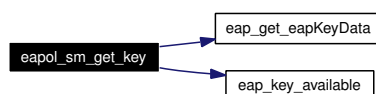
Returns:

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 1335 of file eapol_sm.c.

Here is the call graph for this function:



6.64.3.4 `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 1040 of file eapol_sm.c.

6.64.3.5 `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

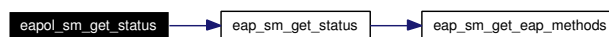
Returns:

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 992 of file eapol_sm.c.

Here is the call graph for this function:

**6.64.3.6** `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\(\)](#)

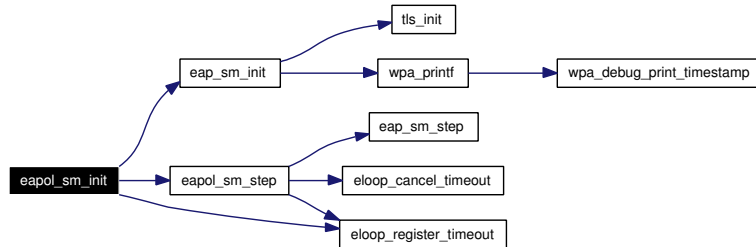
Returns:

Pointer to the allocated EAPOL state machine or NULL on failure

Allocate and initialize an EAPOL state machine.

Definition at line 1692 of file eapol_sm.c.

Here is the call graph for this function:



6.64.3.7 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 1377 of file eapol_sm.c.

Here is the call graph for this function:



6.64.3.8 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\(\)](#)

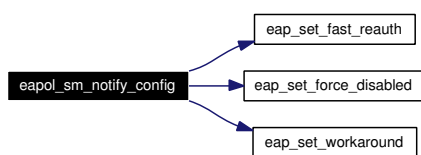
config Pointer to current network configuration

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 pointer. *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 1301 of file eapol_sm.c.

Here is the call graph for this function:



6.64.3.9 void eapol_sm_notify_ctrl_attached (struct eapol_sm * sm)

Notification of attached monitor.

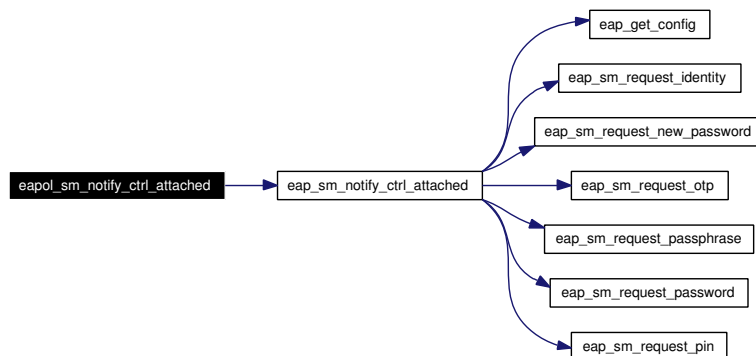
Parameters:

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 1473 of file eapol_sm.c.

Here is the call graph for this function:



6.64.3.10 void eapol_sm_notify_ctrl_response (struct eapol_sm * sm)

Notification of received user input.

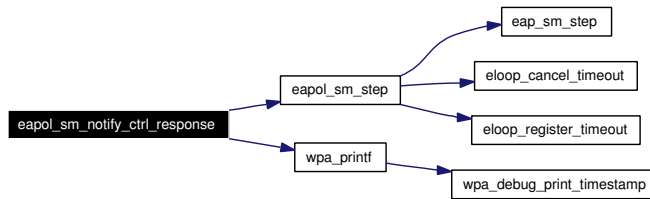
Parameters:

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 1489 of file eapol_sm.c.

Here is the call graph for this function:



6.64.3.11 void eapol_sm_notify_eap_fail (struct eapol_sm * sm, Boolean fail)

Notification of external EAP failure trigger.

Parameters:

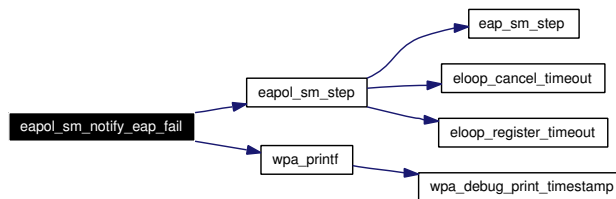
sm Pointer to EAPOL state machine allocated with [eapol_sm_init\(\)](#)

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 1276 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.12 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\(\)](#)

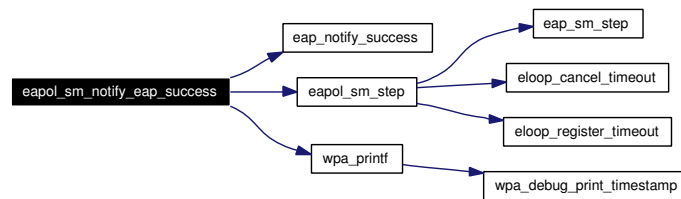
success TRUE = set success, FALSE = clear success

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 1253 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.13 void eapol_sm_notify_logoff (struct [eapol_sm](#) * *sm*, Boolean *logoff*)

Notification of logon/logoff commands.

Parameters:

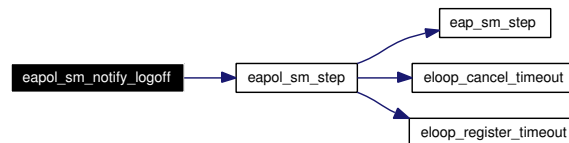
sm Pointer to EAPOL state machine allocated with [eapol_sm_init\(\)](#)

logoff Whether command was logoff

Notify EAPOL state machines that user requested logon/logoff.

Definition at line 1360 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.14 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 1529 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.15 void eapol_sm_notify_pmksa_attempt (struct [eapol_sm](#) * *sm*, int *attempt*)

Notification of PMKSA caching.

Parameters:

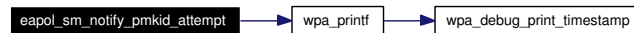
sm Pointer to EAPOL state machine allocated with `eapol_sm_init()`

attempt Whether PMKSA caching is tried

Notify EAPOL state machines whether PMKSA caching is used.

Definition at line 1395 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.16 void eapol_sm_notify_portControl (struct `eapol_sm` * *sm*, `PortControl` *portControl*)

Notification of portControl changes.

Parameters:

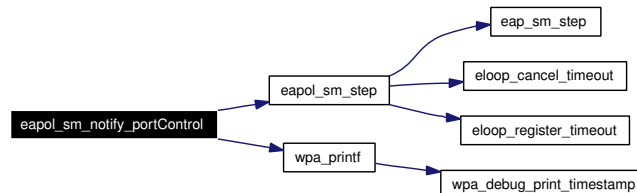
sm Pointer to EAPOL state machine allocated with `eapol_sm_init()`

portControl New value for portControl variable

Notify EAPOL state machines that portControl variable has changed.

Definition at line 1454 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.17 void eapol_sm_notify_portEnabled (struct `eapol_sm` * *sm*, Boolean *enabled*)

Notification about portEnabled change.

Parameters:

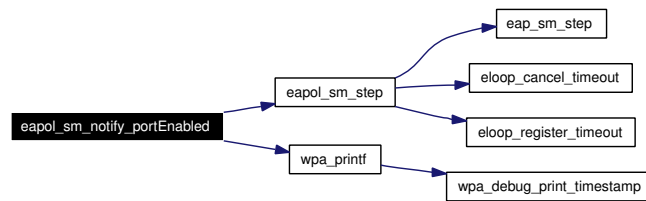
sm Pointer to EAPOL state machine allocated with `eapol_sm_init()`

enabled New portEnabled value

Notify EAPOL station machine about new portEnabled value.

Definition at line 1211 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.18 void eapol_sm_notify_portValid (struct [eapol_sm](#) * *sm*, Boolean *valid*)

Notification about portValid change.

Parameters:

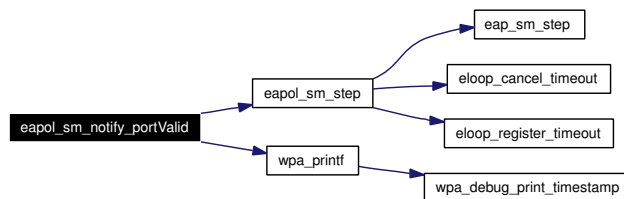
sm Pointer to EAPOL state machine allocated with [eapol_sm_init\(\)](#)

valid New portValid value

Notify EAPOL station machine about new portValid value.

Definition at line 1230 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.19 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 1196 of file `eapol_sm.c`.

6.64.3.20 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 1437 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.21 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 1512 of file `eapol_sm.c`.

6.64.3.22 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)

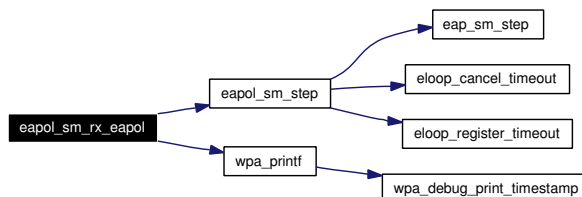
len Length of the EAPOL frame

Returns:

1 = EAPOL frame processed, 0 = not for EAPOL state machine, -1 failure

Definition at line 1097 of file `eapol_sm.c`.

Here is the call graph for this function:



6.64.3.23 void eapol_sm_step (struct eapol_sm * sm)

EAPOL state machine step function.

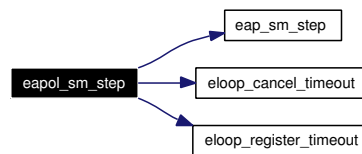
Parameters:

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 846 of file eapol_sm.c.

Here is the call graph for this function:



6.65 eapol_sm.h File Reference

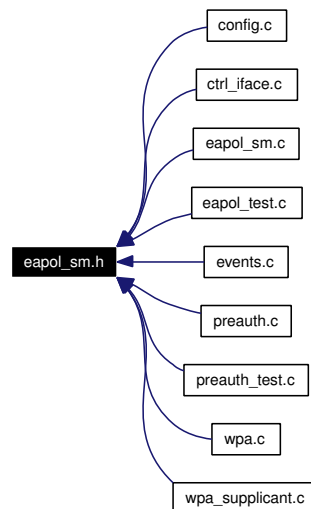
WPA Supplicant / EAPOL state machines.

```
#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_scared_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.

6.65.1 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](#).

6.65.2 Function Documentation

6.65.2.1 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 962 of file `eapol_sm.c`.

6.65.2.2 void eapol_sm_deinit (struct eapol_sm * sm)

Deinitialize EAPOL state machine.

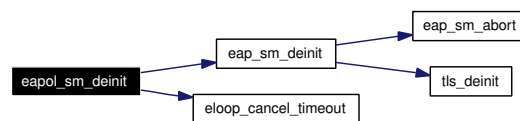
Parameters:

sm Pointer to EAPOL state machine allocated with [eapol_sm_init\(\)](#)

Deinitialize and free EAPOL state machine.

Definition at line 1742 of file eapol_sm.c.

Here is the call graph for this function:

**6.65.2.3 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

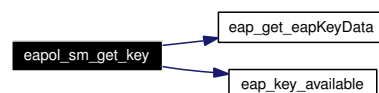
Returns:

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 1335 of file eapol_sm.c.

Here is the call graph for this function:

**6.65.2.4 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 1040 of file eapol_sm.c.

6.65.2.5 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

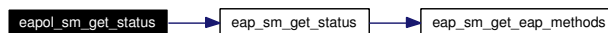
Returns:

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 992 of file eapol_sm.c.

Here is the call graph for this function:



6.65.2.6 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 1692 of file eapol_sm.c.

Here is the call graph for this function:

6.65.2.9 void eapol_sm_notify_ctrl_attached (struct eapol_sm * sm)

Notification of attached monitor.

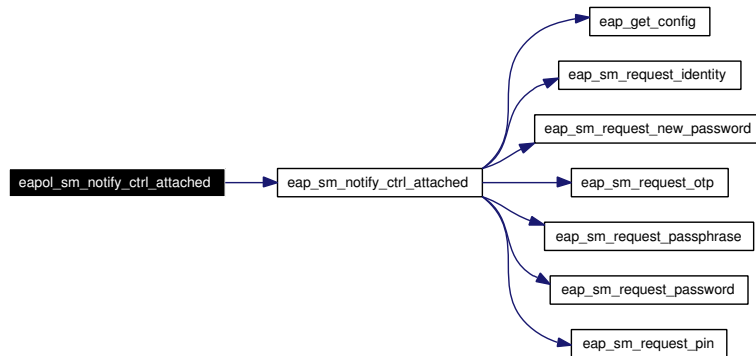
Parameters:

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 1473 of file `eapol_sm.c`.

Here is the call graph for this function:



6.65.2.10 void eapol_sm_notify_ctrl_response (struct eapol_sm * sm)

Notification of received user input.

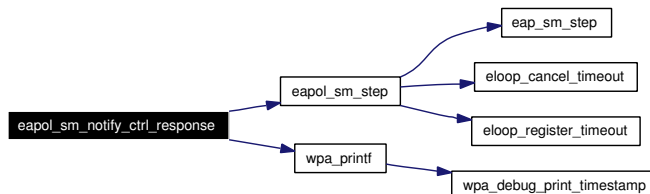
Parameters:

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 1489 of file `eapol_sm.c`.

Here is the call graph for this function:



6.65.2.11 void eapol_sm_notify_eap_fail (struct eapol_sm * sm, Boolean fail)

Notification of external EAP failure trigger.

Parameters:

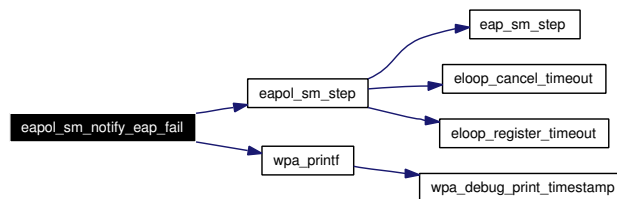
sm Pointer to EAPOL state machine allocated with `eapol_sm_init()`

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 1276 of file `eapol_sm.c`.

Here is the call graph for this function:



6.65.2.12 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()`

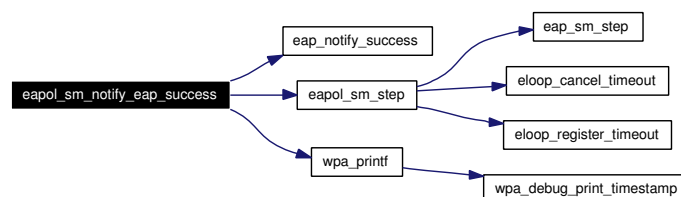
success TRUE = set success, FALSE = clear success

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 1253 of file `eapol_sm.c`.

Here is the call graph for this function:



6.65.2.13 void eapol_sm_notify_logoff (struct `eapol_sm` * *sm*, Boolean *logoff*)

Notification of logon/logoff commands.

Parameters:

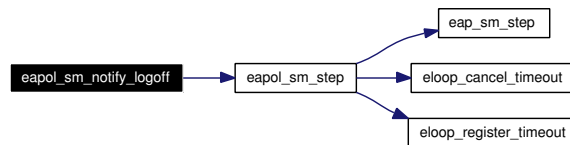
sm Pointer to EAPOL state machine allocated with `eapol_sm_init()`

logoff Whether command was logoff

Notify EAPOL state machines that user requested logon/logoff.

Definition at line 1360 of file eapol_sm.c.

Here is the call graph for this function:



6.65.2.14 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 1529 of file eapol_sm.c.

Here is the call graph for this function:



6.65.2.15 void eapol_sm_notify_pmksa_attempt (struct eapol_sm * sm, int attempt)

Notification of PMKSA caching.

Parameters:

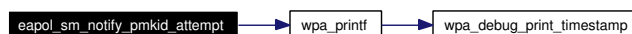
sm Pointer to EAPOL state machine allocated with [eapol_sm_init\(\)](#)

attempt Whether PMKSA caching is tried

Notify EAPOL state machines whether PMKSA caching is used.

Definition at line 1395 of file eapol_sm.c.

Here is the call graph for this function:



6.65.2.16 void eapol_sm_notify_portControl (struct eapol_sm * sm, PortControl portControl)

Notification of portControl changes.

Parameters:

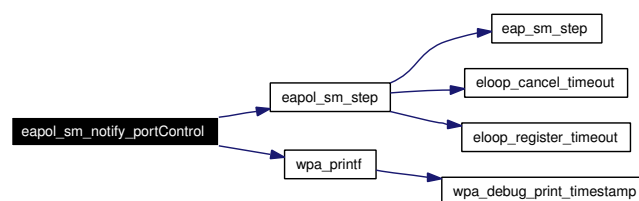
sm Pointer to EAPOL state machine allocated with `eapol_sm_init()`

portControl New value for portControl variable

Notify EAPOL state machines that portControl variable has changed.

Definition at line 1454 of file eapol_sm.c.

Here is the call graph for this function:

**6.65.2.17 void eapol_sm_notify_portEnabled (struct eapol_sm * sm, Boolean enabled)**

Notification about portEnabled change.

Parameters:

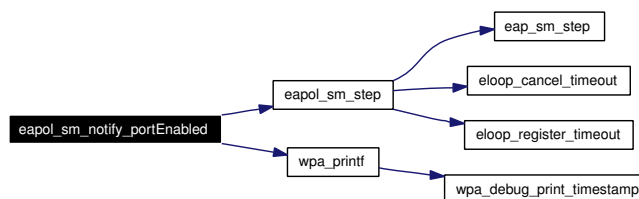
sm Pointer to EAPOL state machine allocated with `eapol_sm_init()`

enabled New portEnabled value

Notify EAPOL station machine about new portEnabled value.

Definition at line 1211 of file eapol_sm.c.

Here is the call graph for this function:

**6.65.2.18 void eapol_sm_notify_portValid (struct eapol_sm * sm, Boolean valid)**

Notification about portValid change.

Parameters:

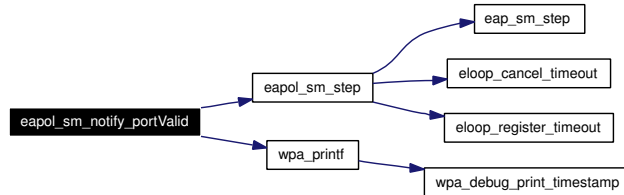
sm Pointer to EAPOL state machine allocated with `eapol_sm_init()`

valid New portValid value

Notify EAPOL station machine about new portValid value.

Definition at line 1230 of file eapol_sm.c.

Here is the call graph for this function:



6.65.2.19 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 1196 of file eapol_sm.c.

6.65.2.20 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 1437 of file eapol_sm.c.

Here is the call graph for this function:



6.65.2.21 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 1512 of file eapol_sm.c.

6.65.2.22 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)

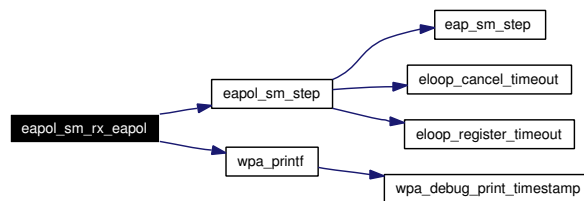
len Length of the EAPOL frame

Returns:

1 = EAPOL frame processed, 0 = not for EAPOL state machine, -1 failure

Definition at line 1097 of file eapol_sm.c.

Here is the call graph for this function:



6.65.2.23 void eapol_sm_step (struct eapol_sm * sm)

EAPOL state machine step function.

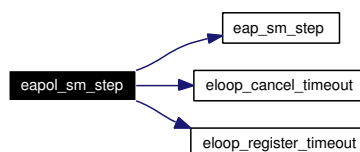
Parameters:

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 846 of file eapol_sm.c.

Here is the call graph for this function:

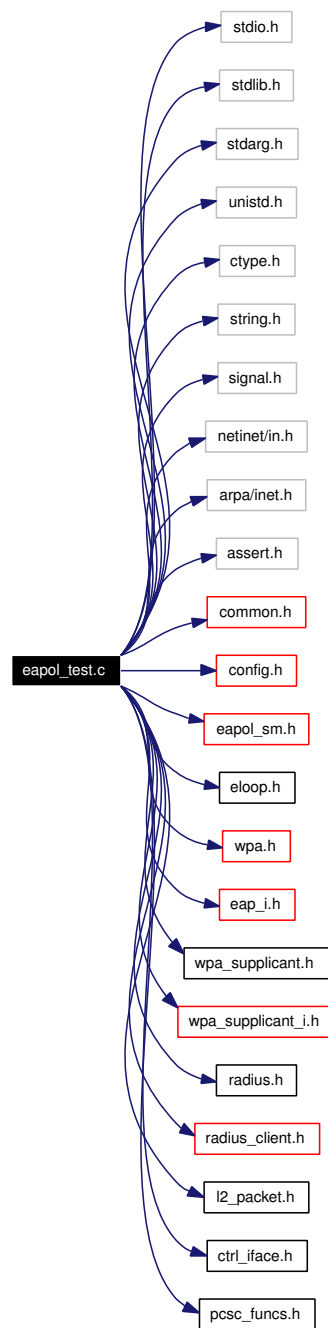


6.66 eapol_test.c File Reference

WPA Supplicant - test code.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <unistd.h>
#include <ctype.h>
#include <string.h>
#include <signal.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <assert.h>
#include "common.h"
#include "config.h"
#include "eapol_sm.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 **main** (int argc, char *argv[])

Variables

- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- [wpa_driver_ops](#) * **wpa_supplicant_drivers** [] = { }

6.66.1 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.

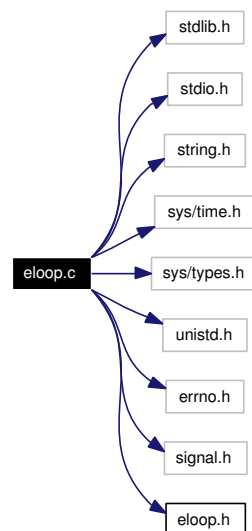
Definition in file [eapol_test.c](#).

6.67 eloop.c File Reference

Event loop based on select() loop.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>
#include <errno.h>
#include <signal.h>
#include "eloop.h"
```

Include dependency graph for eloop.c:



Functions

- void [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)
Register handler for read events.
- 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)

Register timeout.

- int `eloop_cancel_timeout` (void(*handler)(void *eloop_ctx, void *sock_ctx), void *eloop_data, void *user_data)

Cancel timeouts.

- int `eloop_register_signal` (int sig, void(*handler)(int sig, void *eloop_ctx, void *signal_ctx), void *user_data)

Register handler for 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.

6.67.1 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](#).

6.67.2 Function Documentation

6.67.2.1 int `eloop_cancel_timeout` (void(*) (void *eloop_ctx, void *sock_ctx) 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 178 of file eloop.c.

6.67.2.2 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 378 of file eloop.c.

6.67.2.3 void eloop_init (void * user_data)

Initialize global event loop data.

Parameters:

user_data Pointer to global data passed as eloop_ctx to signal handlers

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 74 of file eloop.c.

6.67.2.4 int eloop_register_read_sock (int sock, void(*) (int sock, void *eloop_ctx, void *sock_ctx) 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.

Definition at line 81 of file eloop.c.

6.67.2.5 int eloop_register_signal (int sig, void(*) (int sig, void *eloop_ctx, void *signal_ctx) 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 273 of file `eloop.c`.

6.67.2.6 int `eloop_register_timeout` (unsigned int *secs*, unsigned int *usecs*, void(*) (void **eloop_ctx*, void **timeout_ctx*) *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 131 of file `eloop.c`.

6.67.2.7 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 299 of file `eloop.c`.

6.67.2.8 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 372 of file `eloop.c`.

6.67.2.9 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 393 of file eloop.c.

6.67.2.10 void eloop_unregister_read_sock (int sock)

Unregister handler for read events.

Parameters:

sock File descriptor number for the socket

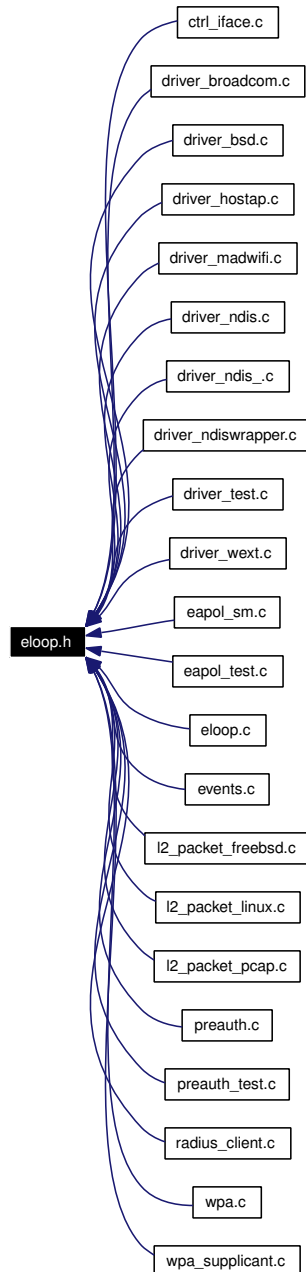
Unregister a read socket notifier that was previously registered with [eloop_register_read_sock\(\)](#).

Definition at line 108 of file eloop.c.

6.68 eloop.h File Reference

Event loop.

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



Defines

- `#define ELOOP_ALL_CTX (void *) -1`
eloop_cancel_timeout() magic number to match all timeouts

Functions

- void `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)
Register handler for read events.
- 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)
Register timeout.
- int `eloop_cancel_timeout` (void(*handler)(void *eloop_ctx, void *sock_ctx), void *eloop_data, void *user_data)
Cancel timeouts.
- int `eloop_register_signal` (int sig, void(*handler)(int sig, void *eloop_ctx, void *signal_ctx), void *user_data)
Register handler for 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.

6.68.1 Detailed Description

Event 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.

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. [elooop.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 [elooop.h](#).

6.68.2 Function Documentation

6.68.2.1 `int elooop_cancel_timeout (void(*) (void *elooop_ctx, void *sock_ctx) handler, void *elooop_data, void * user_data)`

Cancel timeouts.

Parameters:

handler Matching callback function

elooop_data Matching `elooop_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,elooop_data,user_data>` timeouts registered with [elooop_register_timeout\(\)](#). `ELOOP_ALL_CTX` can be used as a wildcard for cancelling all timeouts regardless of `elooop_data/user_data`.

Definition at line 178 of file `elooop.c`.

6.68.2.2 `void elooop_destroy (void)`

Free any resources allocated for the event loop.

After calling [elooop_destroy\(\)](#), other `elooop_*` functions must not be called before re-running [elooop_init\(\)](#).

Definition at line 378 of file `elooop.c`.

6.68.2.3 `void elooop_init (void * user_data)`

Initialize global event loop data.

Parameters:

user_data Pointer to global data passed as `elooop_ctx` to signal handlers

This function must be called before any other `elooop_*` function. `user_data` can be used to configure a global (to the process) pointer that will be passed as `elooop_ctx` parameter to signal handlers.

Definition at line 74 of file `elooop.c`.

6.68.2.4 `int elooop_register_read_sock (int sock, void(*) (int sock, void *elooop_ctx, void *sock_ctx) handler, void * elooop_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.

Definition at line 81 of file eloop.c.

6.68.2.5 int eloop_register_signal (int sig, void(*)(int sig, void *eloop_ctx, void *signal_ctx) 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 273 of file eloop.c.

6.68.2.6 int eloop_register_timeout (unsigned int secs, unsigned int usecs, void(*)(void *eloop_ctx, void *timeout_ctx) 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 131 of file eloop.c.

6.68.2.7 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 299 of file `eloop.c`.

6.68.2.8 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 372 of file `eloop.c`.

6.68.2.9 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 393 of file `eloop.c`.

6.68.2.10 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()`.

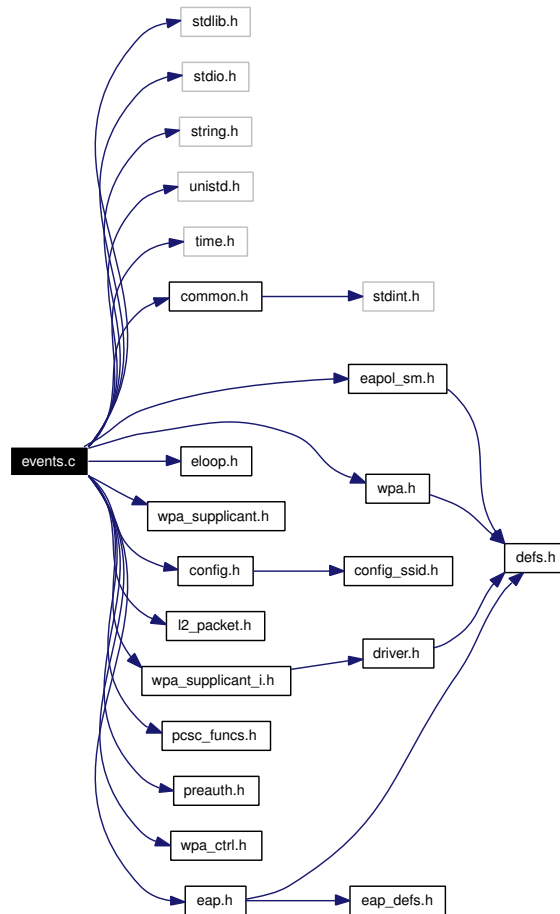
Definition at line 108 of file `eloop.c`.

6.69 events.c File Reference

WPA Supplicant - Driver event processing.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <time.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 "wpa_ctrl.h"
#include "eap.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.

6.69.1 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](#).

6.69.2 Function Documentation

6.69.2.1 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\(\)](#)

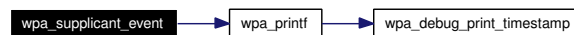
event event type (defined above)

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 732 of file [events.c](#).

Here is the call graph for this function:



6.69.2.2 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

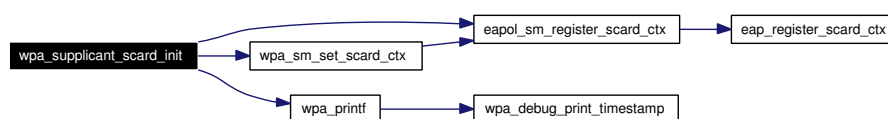
Returns:

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 173 of file [events.c](#).

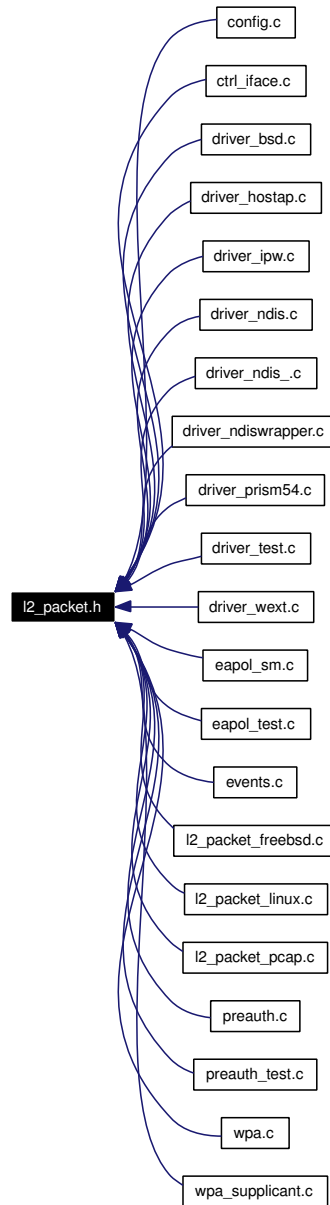
Here is the call graph for this function:



6.70 l2_packet.h File Reference

WPA Supplicant - Layer2 packet interface definition.

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



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` **packed**

6.70.1 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](#).

6.70.2 Function Documentation

6.70.2.1 void l2_packet_deinit (struct l2_packet_data * l2)

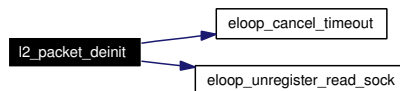
Deinitialize l2_packet interface.

Parameters:

l2 Pointer to internal l2_packet data from [l2_packet_init\(\)](#)

Definition at line 232 of file l2_packet_freebsd.c.

Here is the call graph for this function:



6.70.2.2 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

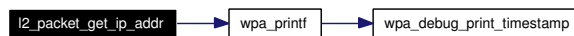
Returns:

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 242 of file l2_packet_freebsd.c.

Here is the call graph for this function:



6.70.2.3 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 51 of file l2_packet_freebsd.c.

6.70.2.4 `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 199 of file l2_packet_freebsd.c.

6.70.2.5 `void l2_packet_notify_auth_start (struct l2_packet_data * l2)`

Notify l2_packet about start of authentication.

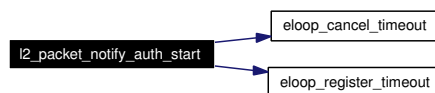
Parameters:

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 278 of file l2_packet_freebsd.c.

Here is the call graph for this function:



6.70.2.6 `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

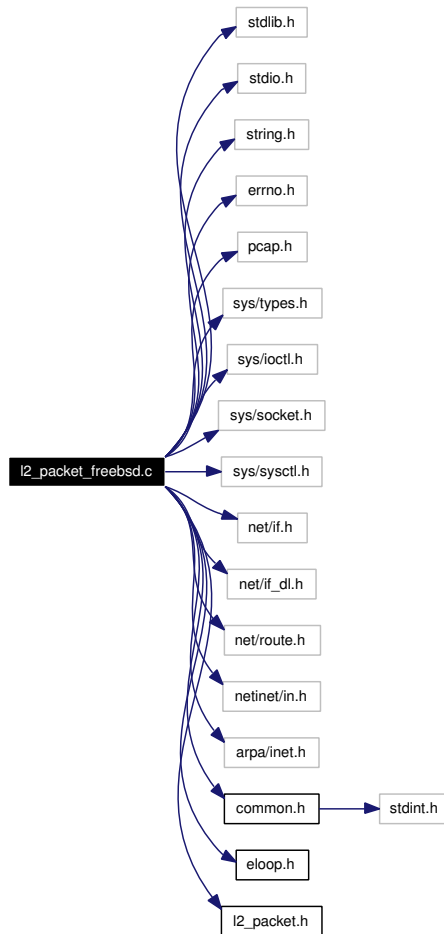
Definition at line 58 of file `l2_packet_freebsd.c`.

6.71 l2_packet_freebsd.c File Reference

WPA Supplicant - Layer2 packet handling with FreeBSD.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <pcap.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <sys/sysctl.h>
#include <net/if.h>
#include <net/if_dl.h>
#include <net/route.h>
#include <netinet/in.h>
#include <arpa/inet.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.

6.71.1 Detailed Description

WPA Supplicant - Layer2 packet handling with FreeBSD.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <jkmaline@cc.hut.fi> Copyright (c) 2005, Sam Lefler <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](#).

6.71.2 Function Documentation

6.71.2.1 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 232 of file [l2_packet_freebsd.c](#).

6.71.2.2 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

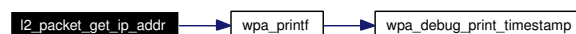
Returns:

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 242 of file [l2_packet_freebsd.c](#).

Here is the call graph for this function:



6.71.2.3 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 51 of file l2_packet_freebsd.c.

6.71.2.4 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 199 of file l2_packet_freebsd.c.

6.71.2.5 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 increase 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 278 of file l2_packet_freebsd.c.

6.71.2.6 `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

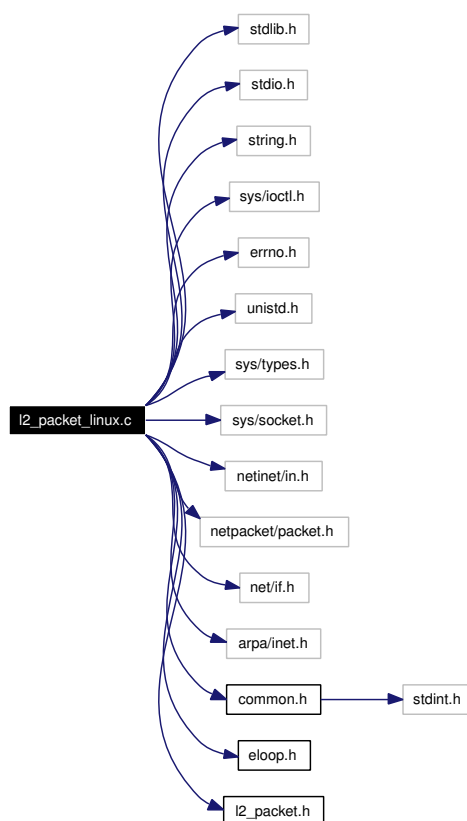
Definition at line 58 of file l2_packet_freebsd.c.

6.72 l2_packet_linux.c File Reference

WPA Supplicant - Layer2 packet handling with Linux packet sockets.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netpacket/packet.h>
#include <net/if.h>
#include <arpa/inet.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.

6.72.1 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](#).

6.72.2 Function Documentation

6.72.2.1 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 162 of file `l2_packet_linux.c`.

Here is the call graph for this function:



6.72.2.2 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 176 of file `l2_packet_linux.c`.

6.72.2.3 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 47 of file l2_packet_linux.c.

6.72.2.4 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

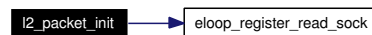
Returns:

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 102 of file l2_packet_linux.c.

Here is the call graph for this function:

**6.72.2.5 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 increase 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 203 of file `l2_packet_linux.c`.

6.72.2.6 `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

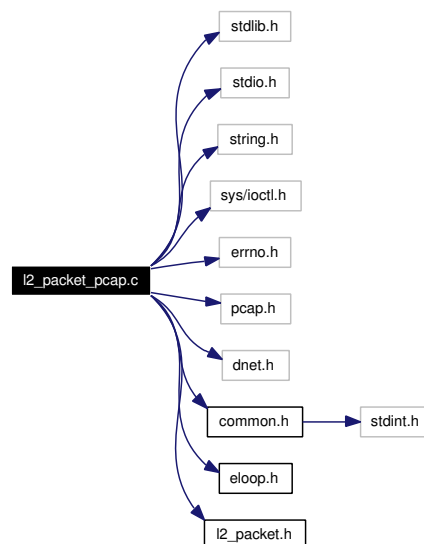
Definition at line 54 of file `l2_packet_linux.c`.

6.73 l2_packet_pcap.c File Reference

WPA Supplicant - Layer2 packet handling with libpcap/libdnet and WinPcap.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/ioctl.h>
#include <errno.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.

6.73.1 Detailed Description

WPA Supplicant - Layer2 packet handling with libpcap/libdnet and WinPcap.

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_pcap.c](#).

6.73.2 Function Documentation

6.73.2.1 void l2_packet_deinit (struct l2_packet_data * l2)

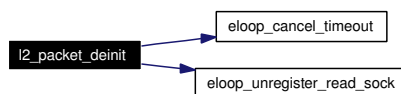
Deinitialize l2_packet interface.

Parameters:

l2 Pointer to internal l2_packet data from [l2_packet_init\(\)](#)

Definition at line 320 of file [l2_packet_pcap.c](#).

Here is the call graph for this function:



6.73.2.2 `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

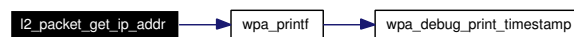
Returns:

- 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 338 of file l2_packet_pcap.c.

Here is the call graph for this function:

**6.73.2.3** `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 57 of file l2_packet_pcap.c.

6.73.2.4 `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 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 283 of file *l2_packet_pcap.c*.

6.73.2.5 void l2_packet_notify_auth_start (struct l2_packet_data * l2)

Notify *l2_packet* about start of authentication.

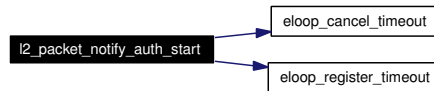
Parameters:

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 increase 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 374 of file *l2_packet_pcap.c*.

Here is the call graph for this function:



6.73.2.6 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

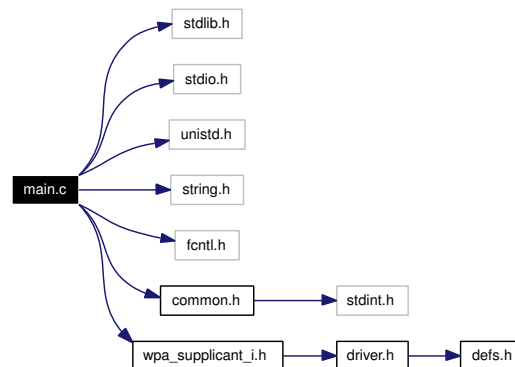
Definition at line 91 of file *l2_packet_pcap.c*.

6.74 main.c File Reference

WPA Supplicant / main() function for UNIX like OSes and MinGW.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.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_license**
- [wpa_driver_ops](#) * **wpa_supplicant_drivers** []

6.74.1 Detailed Description

WPA Supplicant / main() function for UNIX like OSes and MinGW.

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.

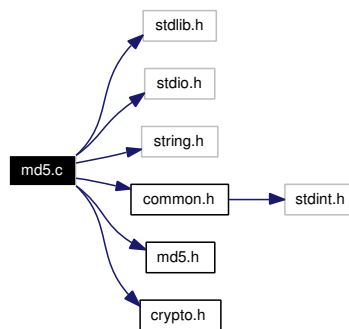
Definition in file [main.c](#).

6.75 md5.c File Reference

MD5 hash implementation and interface functions.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "md5.h"
#include "crypto.h"
```

Include dependency graph for md5.c:



Defines

- #define **byteReverse**(buf, len)
- #define **F1**(x, y, z) (z ^ (x & (y ^ z)))
- #define **F2**(x, y, z) F1(z, x, y)
- #define **F3**(x, y, z) (x ^ y ^ z)
- #define **F4**(x, y, z) (y ^ (x | ~z))
- #define **MD5STEP**(f, w, x, y, z, data, s) (w += f(x, y, z) + data, w = w<<s | w>>(32-s), w += x)

Typedefs

- typedef MD5Context **MD5_CTX**

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).
- void **md5_vector** (size_t num_elem, const u8 *addr[], const size_t *len, u8 *mac)
MD5 hash for data vector.

6.75.1 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](#).

6.75.2 Function Documentation

6.75.2.1 void hmac_md5 (const u8 * key, size_t key_len, const u8 * data, size_t data_len, u8 * mac)

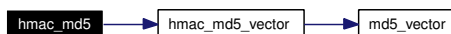
HMAC-MD5 over data buffer (RFC 2104).

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 109 of file md5.c.

Here is the call graph for this function:



6.75.2.2 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 35 of file md5.c.

Here is the call graph for this function:



6.75.2.3 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

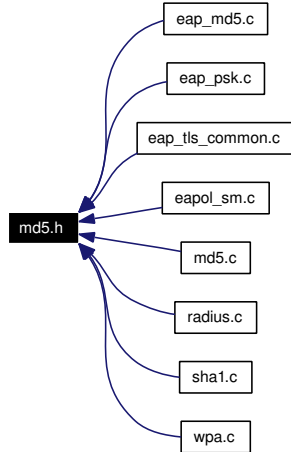
mac Buffer for the hash

Definition at line 141 of file md5.c.

6.76 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).

6.76.1 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](#).

6.76.2 Function Documentation

6.76.2.1 void hmac_md5 (const u8 * key, size_t key_len, const u8 * data, size_t data_len, u8 * mac)

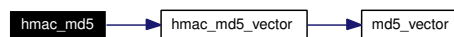
HMAC-MD5 over data buffer (RFC 2104).

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 109 of file md5.c.

Here is the call graph for this function:



6.76.2.2 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 35 of file md5.c.

Here is the call graph for this function:

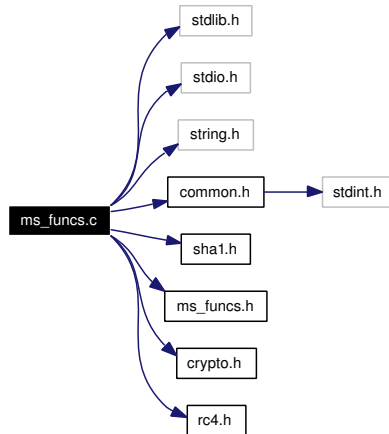


6.77 ms_funcs.c File Reference

WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "sha1.h"
#include "ms_funcs.h"
#include "crypto.h"
#include "rc4.h"
```

Include dependency graph for ms_funcs.c:



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_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_asymmetric_start_key](#) (const u8 *master_key, u8 *session_key, size_t session_key_len, int is_send, int is_server)
GetAsymmetricStartKey() - 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.

6.77.1 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](#).

6.77.2 Function Documentation

6.77.2.1 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 104 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.2 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)

password_len Length of password

nt_response 24-octet NT-Response (IN)

peer_challenge 16-octet PeerChallenge (IN)

auth_challenge 16-octet AuthenticatorChallenge (IN)

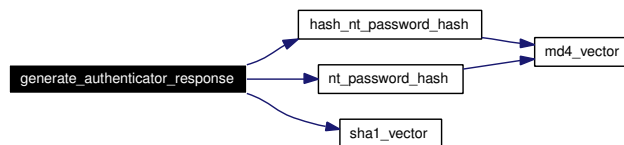
username 0-to-256-char UserName (IN)

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 156 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.3 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

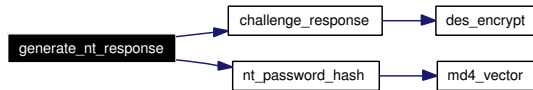
password 0-to-256-unicode-char Password (IN; ASCII)

password_len Length of password

response 24-octet Response (OUT)

Definition at line 128 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.4 void `get_asymmetric_start_key` (const u8 * *master_key*, u8 * *session_key*, size_t *session_key_len*, int *is_send*, int *is_server*)

GetAsymmetricStartKey() - RFC 3079, Sect. 3.4.

Parameters:

master_key 16-octet MasterKey (IN)

session_key 8-to-16 octet SessionKey (OUT)

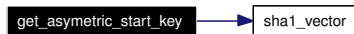
session_key_len SessionKeyLength (Length of session_key) (IN)

is_send IsSend (IN, BOOLEAN)

is_server IsServer (IN, BOOLEAN)

Definition at line 254 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.5 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 224 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.6 void hash_nt_password_hash (const u8 * password_hash, u8 * password_hash_hash)

HashNtPasswordHash() - RFC 2759, Sect. 8.4.

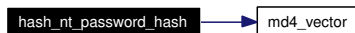
Parameters:

password_hash 16-octet PasswordHash (IN)

password_hash_hash 16-octet PasswordHashHash (OUT)

Definition at line 90 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.7 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

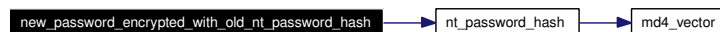
old_password 0-to-256-unicode-char OldPassword (IN; ASCII)

old_password_len Length of old_password

encrypted_pw_block 516-octet EncryptedPwBlock (OUT)

Definition at line 358 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.8 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)

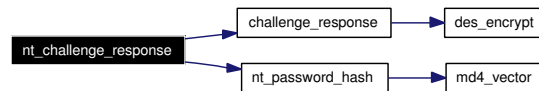
password 0-to-256-unicode-char Password (IN; ASCII)

password_len Length of password

response 24-octet Response (OUT)

Definition at line 208 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.9 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 63 of file ms_funcs.c.

Here is the call graph for this function:



6.77.2.10 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 396 of file ms_funcs.c.

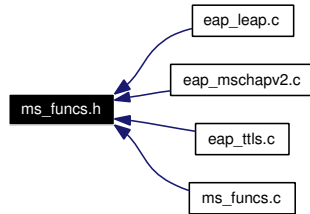
Here is the call graph for this function:



6.78 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_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 [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_asymmetric_start_key](#) (const u8 *master_key, u8 *session_key, size_t session_key_len, int is_send, int is_server)
GetAsymmetricStartKey() - 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.

6.78.1 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](#).

6.78.2 Function Documentation

6.78.2.1 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 104 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.2 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)

password_len Length of password

nt_response 24-octet NT-Response (IN)

peer_challenge 16-octet PeerChallenge (IN)

auth_challenge 16-octet AuthenticatorChallenge (IN)

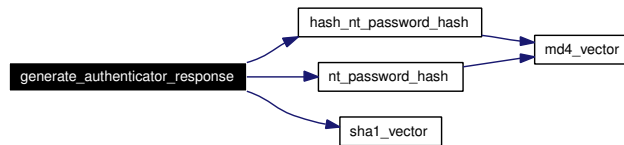
username 0-to-256-char UserName (IN)

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 156 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.3 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_challenge 16-octet PeerChallenge (IN)

username 0-to-256-char UserName (IN)

username_len Length of username

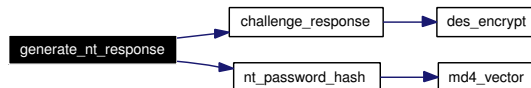
password 0-to-256-unicode-char Password (IN; ASCII)

password_len Length of password

response 24-octet Response (OUT)

Definition at line 128 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.4 void get_asymmetric_start_key (const u8 * *master_key*, u8 * *session_key*, size_t *session_key_len*, int *is_send*, int *is_server*)

GetAsymmetricStartKey() - 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 254 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.5 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 224 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.6 void `hash_nt_password_hash` (const u8 * *password_hash*, u8 * *password_hash_hash*)

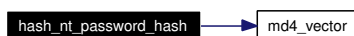
HashNtPasswordHash() - RFC 2759, Sect. 8.4.

Parameters:

password_hash 16-octet PasswordHash (IN)
password_hash_hash 16-octet PasswordHashHash (OUT)

Definition at line 90 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.7 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`)

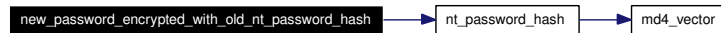
NewPasswordEncryptedWithOldNtPasswordHash() - RFC 2759, Sect. 8.9.

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 358 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.8 void `nt_challenge_response` (`const u8 * challenge`, `const u8 * password`, `size_t password_len`, `u8 * response`)

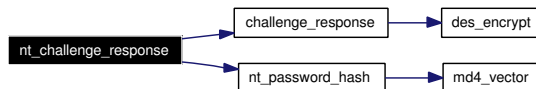
NtChallengeResponse() - RFC 2433, Sect. A.5.

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 208 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.9 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 63 of file ms_funcs.c.

Here is the call graph for this function:



6.78.2.10 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*

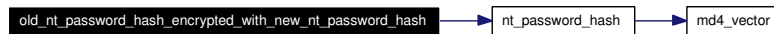
old_password 0-to-256-unicode-char OldPassword (IN; ASCII)

old_password_len Length of *old_password*

encrypted_password_ash 16-octet EncryptedPasswordHash (OUT)

Definition at line 396 of file ms_funcs.c.

Here is the call graph for this function:

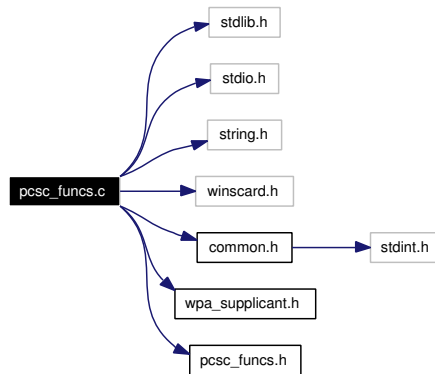


6.79 pcsc_funcs.c File Reference

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <winscard.h>
#include "common.h"
#include "wpa_supplicant.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_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 **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

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.

6.79.1 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](#).

6.79.2 Function Documentation

6.79.2.1 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 364 of file psc_funcs.c.

Here is the call graph for this function:



6.79.2.2 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

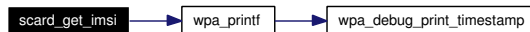
Returns:

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 590 of file psc_funcs.c.

Here is the call graph for this function:



6.79.2.3 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

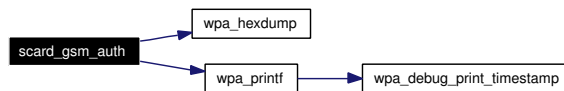
Returns:

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 671 of file pcsc_funcs.c.

Here is the call graph for this function:



6.79.2.4 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 203 of file pcsc_funcs.c.

Here is the call graph for this function:



6.79.2.5 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 334 of file pcsc_funcs.c.

Here is the call graph for this function:



6.79.2.6 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

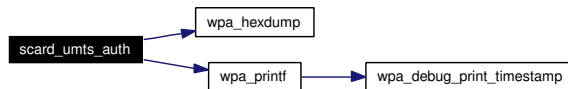
Returns:

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 770 of file pcsc_funcs.c.

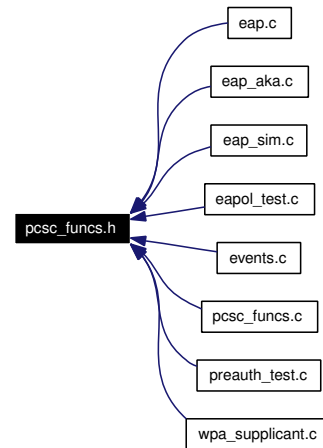
Here is the call graph for this function:



6.80 psc_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_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_ums_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** }

6.80.1 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.

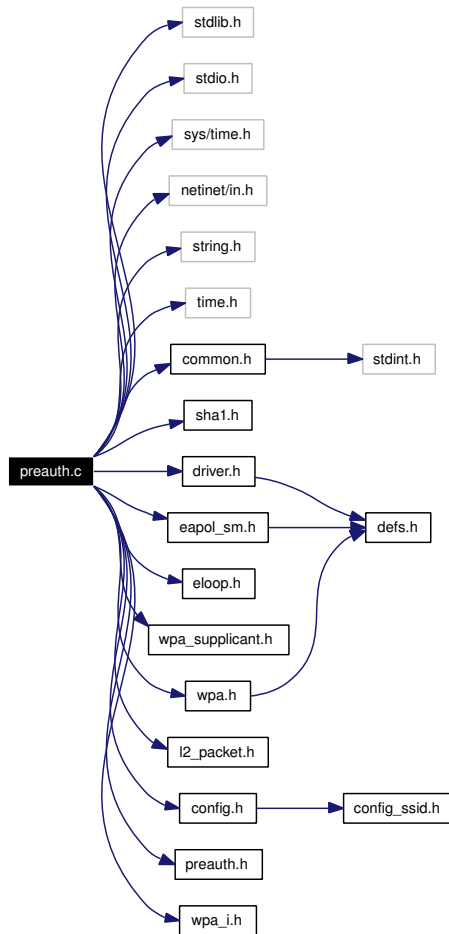
Definition in file [pcsc_funcs.h](#).

6.81 preauth.c File Reference

WPA Supplicant - RSN pre-authentication and PMKSA caching.

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <string.h>
#include <time.h>
#include "common.h"
#include "sha1.h"
#include "wpa.h"
#include "driver.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "config.h"
#include "l2_packet.h"
#include "eapol_sm.h"
#include "preauth.h"
#include "wpa_i.h"
```

Include dependency graph for preauth.c:



Defines

- #define **PMKID_CANDIDATE_PRIO_SCAN** 1000

Functions

- `rsn_pmksa_cache * pmksa_cache_add` (struct `wpa_sm *sm`, 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_free` (struct `wpa_sm *sm`)
Free all entries in PMKSA cache.
- `rsn_pmksa_cache * pmksa_cache_get` (struct `wpa_sm *sm`, const u8 *aa, const u8 *pmkid)
Fetch a PMKSA cache entry.
- void `pmksa_cache_notify_reconfig` (struct `wpa_sm *sm`)
Reconfiguration notification for PMKSA cache.
- `rsn_pmksa_cache * 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.
- 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.

6.81.1 Detailed Description

WPA Supplicant - RSN pre-authentication and PMKSA caching.

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](#).

6.81.2 Function Documentation

6.81.2.1 `struct rsn_pmksa_cache*` `pmksa_cache_add` (`struct wpa_sm` * `sm`, `const u8` * `pmk`, `size_t` `pmk_len`, `const u8` * `aa`, `const u8` * `spa`, `struct wpa_ssid` * `ssid`)

Add a PMKSA cache entry.

Parameters:

- sm* Pointer to WPA state machine data from `wpa_sm_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

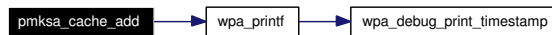
Returns:

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 177 of file `preauth.c`.

Here is the call graph for this function:



6.81.2.2 `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 404 of file `preauth.c`.

6.81.2.3 `void` `pmksa_cache_free` (`struct wpa_sm` * `sm`)

Free all entries in PMKSA cache.

Parameters:

- sm* Pointer to WPA state machine data from `wpa_sm_init()`

Definition at line 271 of file `preauth.c`.

6.81.2.4 `struct rsn_pmksa_cache* pmksa_cache_get (struct wpa_sm * sm, const u8 * aa, const u8 * pmkid)`

Fetch a PMKSA cache entry.

Parameters:

- sm* Pointer to WPA state machine data from `wpa_sm_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 298 of file preauth.c.

6.81.2.5 `struct rsn_pmksa_cache* 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 391 of file preauth.c.

6.81.2.6 `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 453 of file preauth.c.

6.81.2.7 `void pmksa_cache_notify_reconfig (struct wpa_sm * sm)`

Reconfiguration notification for PMKSA cache.

Parameters:

- sm* Pointer to WPA state machine data from `wpa_sm_init()`

Clear references to old data structures when `wpa_supplicant` is reconfigured.

Definition at line 320 of file preauth.c.

6.81.2.8 `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:

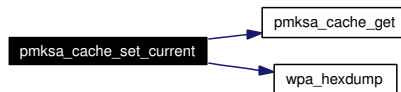
- sm* Pointer to WPA state machine data from `wpa_sm_init()`
- pmkid* PMKID for selecting PMKSA or NULL if not used
- bssid* BSSID for PMKSA or NULL if not used
- ssid* The network configuration for the current network
- try_opportunistic* Whether to allow opportunistic PMKSA caching

Returns:

- 0 if PMKSA was found or -1 if no matching entry was found

Definition at line 422 of file `preauth.c`.

Here is the call graph for this function:



6.81.2.9 `void pmksa_candidate_add (struct wpa_sm * sm, const u8 * bssid, int prio, int preauth)`

Add a new PMKSA candidate.

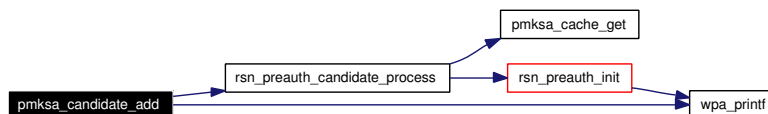
Parameters:

- 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 787 of file `preauth.c`.

Here is the call graph for this function:



6.81.2.10 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 487 of file preauth.c.

6.81.2.11 void rsn_preauth_candidate_process (struct wpa_sm * sm)

Process PMKSA candidates.

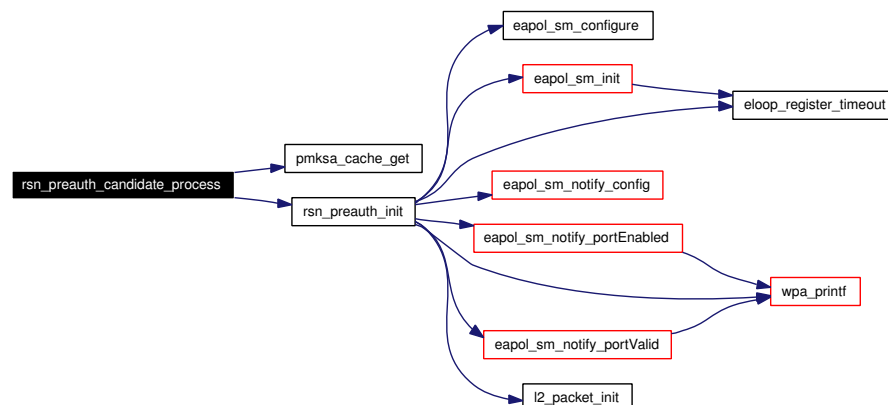
Parameters:

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 723 of file preauth.c.

Here is the call graph for this function:

**6.81.2.12 void rsn_preauth_deinit (struct wpa_sm * sm)**

Abort RSN pre-authentication.

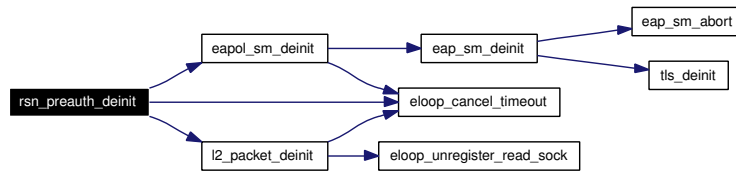
Parameters:

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 699 of file preauth.c.

Here is the call graph for this function:



6.81.2.13 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 924 of file preauth.c.

Here is the call graph for this function:



6.81.2.14 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\(\)](#)

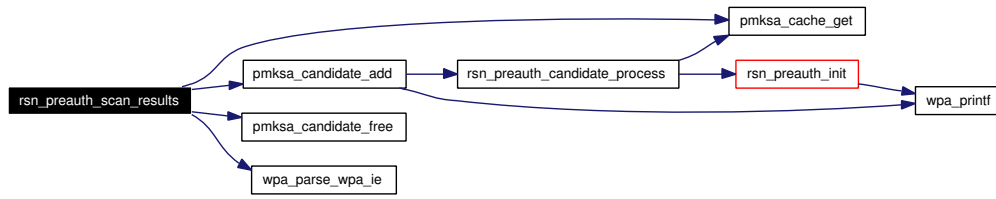
Definition at line 948 of file preauth.c.

6.81.2.15 int rsn_preauth_init (struct wpa_sm * sm, const u8 * dst, struct wpa_ssid * config)

Start new RSN pre-authentication.

Parameters:

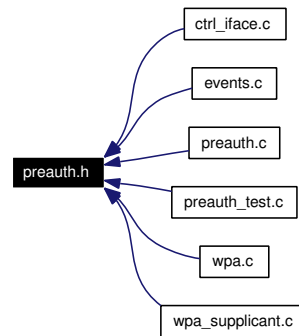
- sm* Pointer to WPA state machine data from [wpa_sm_init\(\)](#)
- dst* Authenticator address (BSSID) with which to preauthenticate
- config* Current network configuration



6.82 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_cache_free](#) (struct [wpa_sm](#) *sm)
Free all entries in PMKSA cache.
- [rsn_pmksa_cache](#) * [pmksa_cache_get](#) (struct [wpa_sm](#) *sm, 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.
- void [pmksa_candidate_free](#) (struct [wpa_sm](#) *sm)
Free all entries in PMKSA candidate list.
- [rsn_pmksa_cache](#) * [pmksa_cache_add](#) (struct [wpa_sm](#) *sm, 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 [wpa_sm](#) *sm)
Reconfiguration notification for PMKSA cache.
- [rsn_pmksa_cache](#) * [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 [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.

6.82.1 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](#).

6.82.2 Function Documentation

6.82.2.1 struct [rsn_pmksa_cache](#)* [pmksa_cache_add](#) (struct [wpa_sm](#) *sm, const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, struct [wpa_ssid](#) *ssid)

Add a PMKSA cache entry.

Parameters:

sm Pointer to WPA state machine data from [wpa_sm_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

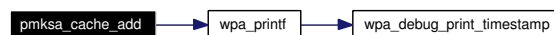
Returns:

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 177 of file preauth.c.

Here is the call graph for this function:



6.82.2.2 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 404 of file preauth.c.

6.82.2.3 void pmksa_cache_free (struct wpa_sm * sm)

Free all entries in PMKSA cache.

Parameters:

sm Pointer to WPA state machine data from [wpa_sm_init\(\)](#)

Definition at line 271 of file preauth.c.

6.82.2.4 struct rsn_pmksa_cache* pmksa_cache_get (struct wpa_sm * sm, const u8 * aa, const u8 * pmkid)

Fetch a PMKSA cache entry.

Parameters:

sm Pointer to WPA state machine data from [wpa_sm_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 298 of file preauth.c.

6.82.2.5 `struct rsn_pmksa_cache*` `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 391 of file `preauth.c`.

6.82.2.6 `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 453 of file `preauth.c`.

6.82.2.7 `void` `pmksa_cache_notify_reconfig` (`struct wpa_sm * sm`)

Reconfiguration notification for PMKSA cache.

Parameters:

sm Pointer to WPA state machine data from `wpa_sm_init()`

Clear references to old data structures when `wpa_supplicant` is reconfigured.

Definition at line 320 of file `preauth.c`.

6.82.2.8 `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

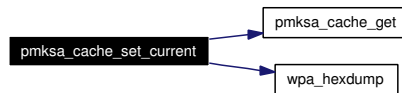
try_opportunistic Whether to allow opportunistic PMKSA caching

Returns:

0 if PMKSA was found or -1 if no matching entry was found

Definition at line 422 of file preauth.c.

Here is the call graph for this function:



6.82.2.9 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

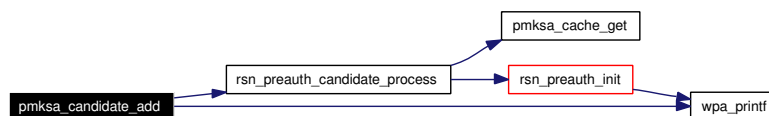
prio Priority (the smaller number, the higher priority)

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 787 of file preauth.c.

Here is the call graph for this function:



6.82.2.10 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 487 of file preauth.c.

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 924 of file preauth.c.

Here is the call graph for this function:

**6.82.2.14 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 948 of file preauth.c.

6.82.2.15 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

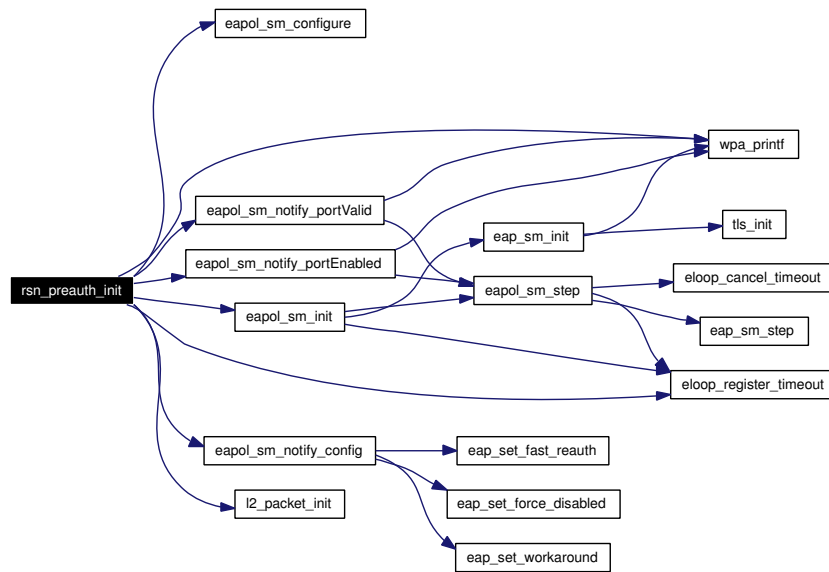
Returns:

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 620 of file preauth.c.

Here is the call graph for this function:



6.82.2.16 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\(\)](#)

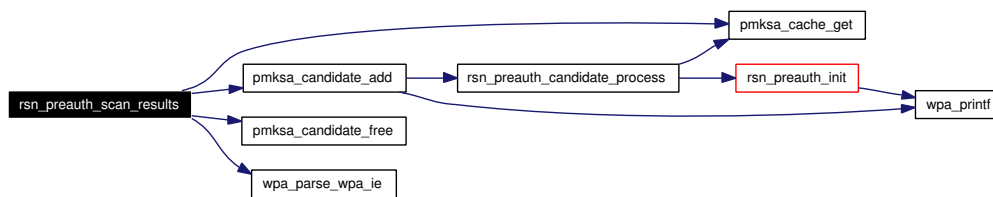
results Scan results

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 863 of file preauth.c.

Here is the call graph for this function:

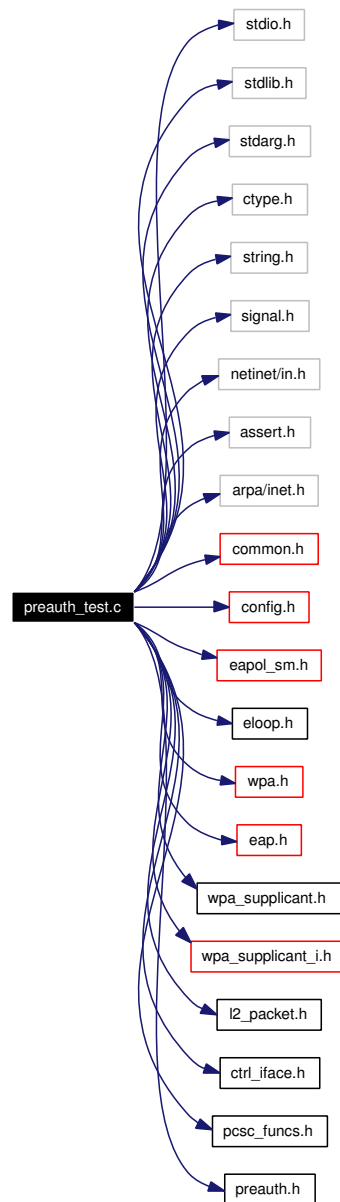


6.83 preauth_test.c File Reference

WPA Supplicant - test code for pre-authentication.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <ctype.h>
#include <string.h>
#include <signal.h>
#include <netinet/in.h>
#include <assert.h>
#include <arpa/inet.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 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** [] = { }

6.83.1 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.

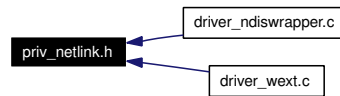
IEEE 802.1X Supplicant test code (to be used in place of [wpa_supplicant.c](#). Not used in production version.

Definition in file [preauth_test.c](#).

6.84 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 **IFLA_IFNAME** 3
- #define **IFLA_WIRELESS** 11
- #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 **NLMSG_ALIGNTO** 4
- #define **NLMSG_ALIGN**(len) (((len) + NLMSG_ALIGNTO - 1) & ~(NLMSG_ALIGNTO - 1))
- #define **NLMSG_LENGTH**(len) ((len) + NLMSG_ALIGN(sizeof(struct nlmsg_hdr)))
- #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)

6.84.1 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](#).

6.84.2 Define Documentation

6.84.2.1 #define RTA_NEXT(rta, attrlen)

Value:

```
((attrlen) -= RTA_ALIGN((rta)->rta_len), \  
(struct rtattr *) (((char *) (rta)) + RTA_ALIGN((rta)->rta_len)))
```

Definition at line 47 of file priv_netlink.h.

6.84.2.2 #define RTA_OK(rta, len)

Value:

```
((len) > 0 && (rta)->rta_len >= sizeof(struct rtattr) && \  
(rta)->rta_len <= (len))
```

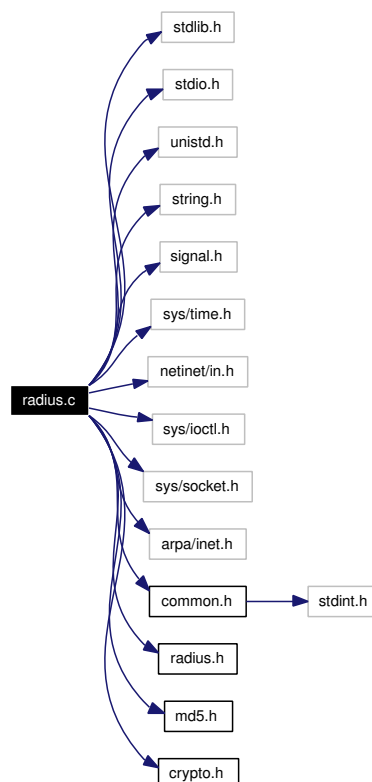
Definition at line 44 of file priv_netlink.h.

6.85 radius.c File Reference

Host AP (software wireless LAN access point) user space daemon for Host AP kernel driver / RADIUS client.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <arpa/inet.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_attr) / sizeof(radius_attr[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, 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)

6.85.1 Detailed Description

Host AP (software wireless LAN access point) user space daemon for Host AP kernel driver / RADIUS client.

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.

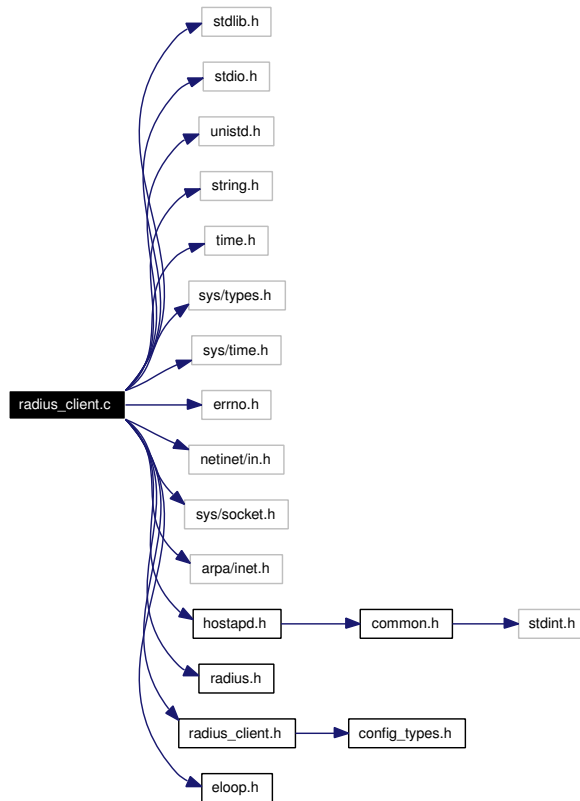
Definition in file [radius.c](#).

6.86 radius_client.c File Reference

Host AP (software wireless LAN access point) user space daemon for Host AP kernel driver / RADIUS client.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <sys/time.h>
#include <errno.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.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, RadiusRx-Result(*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, u8 *addr)
- u8 **radius_client_get_id** (struct radius_client_data *radius)
- void **radius_client_flush** (struct radius_client_data *radius)
- 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)

6.86.1 Detailed Description

Host AP (software wireless LAN access point) user space daemon for Host AP kernel driver / RADIUS client.

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.

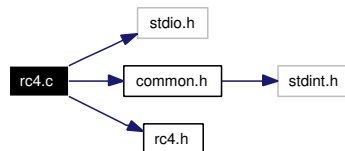
Definition in file [radius_client.c](#).

6.87 rc4.c File Reference

RC4 stream cipher.

```
#include <stdio.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.

6.87.1 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](#).

6.87.2 Function Documentation

6.87.2.1 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 85 of file rc4.c.

Here is the call graph for this function:

**6.87.2.2 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

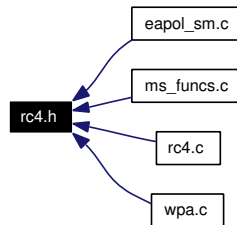
Generate RC4 pseudo random stream for the given key, skip beginning of the stream, and XOR the end result with the data buffer to perform RC4 encryption/decryption.

Definition at line 35 of file rc4.c.

6.88 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.

6.88.1 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](#).

6.88.2 Function Documentation

6.88.2.1 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 85 of file rc4.c.

Here is the call graph for this function:



6.88.2.2 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.

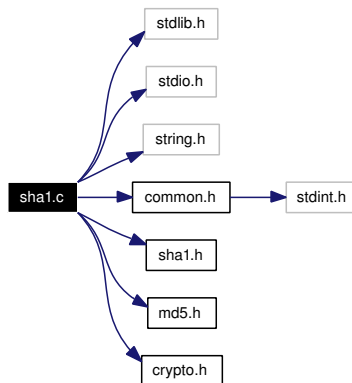
Definition at line 35 of file rc4.c.

6.89 sha1.c File Reference

SHA1 hash implementation and interface functions.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "sha1.h"
#include "md5.h"
#include "crypto.h"
```

Include dependency graph for sha1.c:



Defines

- #define **SHA1HANDSOFF**
- #define **rol**(value, bits) (((value) << (bits)) | ((value) >> (32 - (bits))))
- #define **blk0**(i)
- #define **blk**(i)
- #define **R0**(v, w, x, y, z, i)
- #define **R1**(v, w, x, y, z, i)
- #define **R2**(v, w, x, y, z, i) z += (w ^ x ^ y) + blk(i) + 0x6ED9EBA1 + rol(v, 5); w = rol(w, 30);
- #define **R3**(v, w, x, y, z, i)
- #define **R4**(v, w, x, y, z, i)

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.
- void `sha1_vector` (size_t num_elem, const u8 *addr[], const size_t *len, u8 *mac)
SHA-1 hash for data vector.
- void `sha1_transform` (u8 *state, const u8 data[64])
Perform one SHA-1 transform step.

6.89.1 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`.

6.89.2 Define Documentation

6.89.2.1 #define blk(i)

Value:

```
(block->l[i & 15] = rol(block->l[(i + 13) & 15] ^ \
    block->l[(i + 8) & 15] ^ block->l[(i + 2) & 15] ^ block->l[i & 15], 1))
```

Definition at line 522 of file `sha1.c`.

6.89.2.2 #define blk0(i)**Value:**

```
(block->l[i] = (rol(block->l[i], 24) & 0xFF00FF00) | \
              (rol(block->l[i], 8) & 0x00FF00FF))
```

Definition at line 517 of file sha1.c.

6.89.2.3 #define R0(v, w, x, y, z, i)**Value:**

```
z += ((w & (x ^ y)) ^ y) + blk0(i) + 0x5A827999 + rol(v, 5); \
w = rol(w, 30);
```

Definition at line 526 of file sha1.c.

6.89.2.4 #define R1(v, w, x, y, z, i)**Value:**

```
z += ((w & (x ^ y)) ^ y) + blk(i) + 0x5A827999 + rol(v, 5); \
w = rol(w, 30);
```

Definition at line 529 of file sha1.c.

6.89.2.5 #define R3(v, w, x, y, z, i)**Value:**

```
z += (((w | x) & y) | (w & x)) + blk(i) + 0x8F1BBCDC + rol(v, 5); \
w = rol(w, 30);
```

Definition at line 534 of file sha1.c.

6.89.2.6 #define R4(v, w, x, y, z, i)**Value:**

```
z += (w ^ x ^ y) + blk(i) + 0xCA62C1D6 + rol(v, 5); \
w=rol(w, 30);
```

Definition at line 537 of file sha1.c.

6.89.3 Function Documentation**6.89.3.1 void hmac_sha1 (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)**

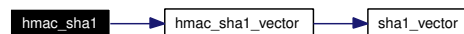
HMAC-SHA1 over data buffer (RFC 2104).

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 109 of file sha1.c.

Here is the call graph for this function:



6.89.3.2 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 36 of file sha1.c.

Here is the call graph for this function:



6.89.3.3 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
iterations 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 358 of file sha1.c.

6.89.3.4 `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).

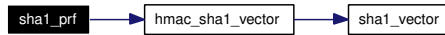
Parameters:

- 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 130 of file sha1.c.

Here is the call graph for this function:



6.89.3.5 `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).

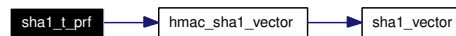
Parameters:

- 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 182 of file sha1.c.

Here is the call graph for this function:



6.89.3.6 void sha1_transform (u8 * state, const u8 data[64])

Perform one SHA-1 transform step.

Parameters:

state SHA-1 state

data Input data for the SHA-1 transform

This function is used to implement 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 and as such, access to just part of the SHA-1 is needed.

Definition at line 424 of file sha1.c.

6.89.3.7 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

Definition at line 400 of file sha1.c.

6.89.3.8 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

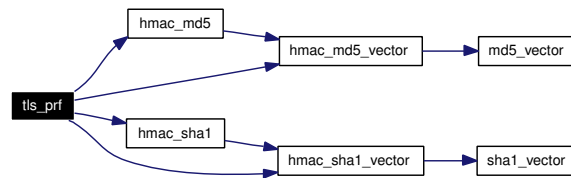
out Buffer for the generated pseudo-random key

outlen Number of bytes of key to generate

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 237 of file sha1.c.

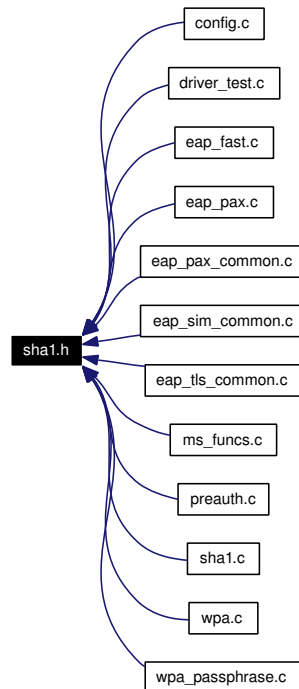
Here is the call graph for this function:



6.90 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.

6.90.1 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](#).

6.90.2 Function Documentation

6.90.2.1 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

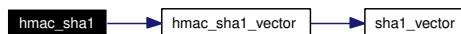
data Pointers to the data area

data_len Length of the data area

mac Buffer for the hash (20 bytes)

Definition at line 109 of file sha1.c.

Here is the call graph for this function:



6.90.2.2 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 36 of file sha1.c.

Here is the call graph for this function:



6.90.2.3 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
iterations 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 358 of file sha1.c.

6.90.2.4 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).

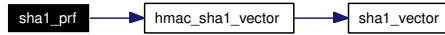
Parameters:

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 130 of file sha1.c.

Here is the call graph for this function:



6.90.2.5 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).

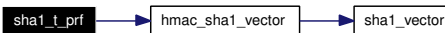
Parameters:

- 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 182 of file sha1.c.

Here is the call graph for this function:



6.90.2.6 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).

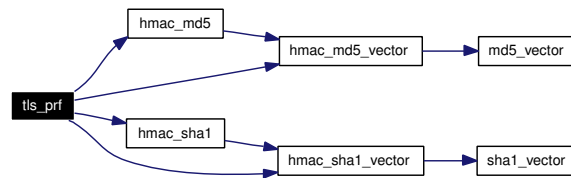
Parameters:

- 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

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 237 of file sha1.c.

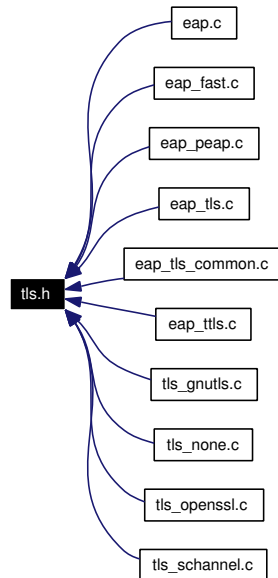
Here is the call graph for this function:



6.91 tls.h File Reference

WPA Supplicant / SSL/TLS interface definition.

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



Enumerations

- enum { **TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED** = -3, **TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED** = -2 }

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 data.
- 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_ca_cert` (void *tls_ctx, const char *ca_cert)
Set trusted CA certificate for all TLS connections.
- 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_global_client_cert` (void *tls_ctx, const char *client_cert)
Set client certificate for all TLS connections.
- int `tls_global_private_key` (void *tls_ctx, const char *private_key, const char *private_key_passwd)
Set private key for all TLS connections.
- 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.
- u8 * `tls_connection_handshake` (void *tls_ctx, struct `tls_connection` *conn, const u8 *in_data, size_t in_len, size_t *out_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_anon_dh` (void *tls_ctx, struct `tls_connection` *conn)

Configure TLS connection to use anonymous DH.

- 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.

6.91.1 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](#).

6.91.2 Function Documentation

6.91.2.1 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

6.91.2.2 `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

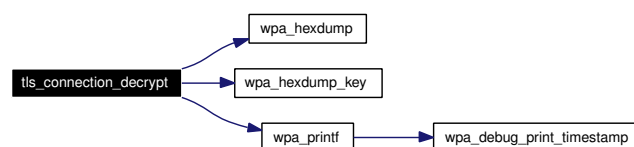
Returns:

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 691 of file `tls_gnutls.c`.

Here is the call graph for this function:



6.91.2.3 `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 239 of file `tls_gnutls.c`.

6.91.2.4 `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 753 of file `tls_gnutls.c`.

6.91.2.5 `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

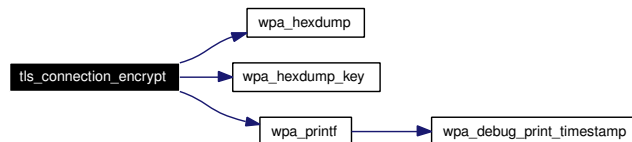
Returns:

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 673 of file `tls_gnutls.c`.

Here is the call graph for this function:



6.91.2.6 `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 254 of file `tls_gnutls.c`.

6.91.2.7 `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 772 of file `tls_gnutls.c`.

6.91.2.8 `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 2130 of file `tls_openssl.c`.

6.91.2.9 `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 495 of file `tls_gnutls.c`.

6.91.2.10 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 780 of file [tls_gnutls.c](#).

6.91.2.11 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 788 of file [tls_gnutls.c](#).

6.91.2.12 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

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.

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

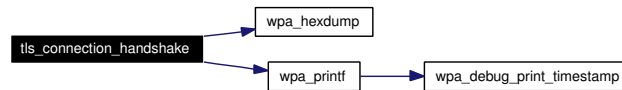
This function is used during TLS handshake. The first call is done with *in_data* == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with *in_data* pointing to the TLS 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 603 of file `tls_gnutls.c`.

Here is the call graph for this function:



6.91.2.13 `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 207 of file `tls_gnutls.c`.

6.91.2.14 `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 719 of file `tls_gnutls.c`.

6.91.2.15 `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 663 of file `tls_gnutls.c`.

6.91.2.16 `int tls_connection_set_anon_dh (void * tls_ctx, struct tls_connection * conn)`

Configure TLS connection to use anonymous DH.

Parameters:

tls_ctx TLS context data from [tls_init\(\)](#)

conn Connection context data from [tls_connection_init\(\)](#)

Returns:

0 on success, -1 on failure

TODO: consider changing this to more generic routine for configuring allowed ciphers

Definition at line 737 of file `tls_gnutls.c`.

6.91.2.17 `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

6.91.2.18 `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\(\)](#)

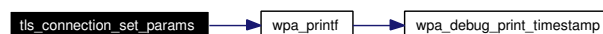
params Connection parameters

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 376 of file `tls_gnutls.c`.

Here is the call graph for this function:



6.91.2.19 `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 468 of file `tls_gnutls.c`.

6.91.2.20 `int tls_connection_shutdown (void * tls_ctx, struct tls_connection * conn)`

Shutdown TLS connection data.

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 260 of file `tls_gnutls.c`.

6.91.2.21 `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 146 of file `tls_gnutls.c`.

6.91.2.22 `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 744 of file `tls_gnutls.c`.

6.91.2.23 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 154 of file `tls_gnutls.c`.

6.91.2.24 int tls_global_ca_cert (void * *tls_ctx*, const char * *ca_cert*)

Set trusted CA certificate for all TLS connections.

Parameters:

tls_ctx TLS context data from [tls_init\(\)](#)

ca_cert File name for CA certificate in PEM or DER format NULL to allow all subjects

Returns:

0 on success, -1 on failure

Definition at line 454 of file `tls_gnutls.c`.

6.91.2.25 int tls_global_client_cert (void * *tls_ctx*, const char * *client_cert*)

Set client certificate for all TLS connections.

Parameters:

tls_ctx TLS context data from [tls_init\(\)](#)

client_cert File name for client certificate in PEM or DER format

Returns:

0 on success, -1 on failure

Definition at line 480 of file `tls_gnutls.c`.

6.91.2.26 `int tls_global_private_key (void * tls_ctx, const char * private_key, const char * private_key_passwd)`

Set private key for all TLS connections.

Parameters:

tls_ctx TLS context data from `tls_init()`

private_key File name for client private key in PEM or DER format

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

Returns:

0 on success, -1 on failure

Definition at line 487 of file `tls_gnutls.c`.

6.91.2.27 `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 461 of file `tls_gnutls.c`.

6.91.2.28 `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.

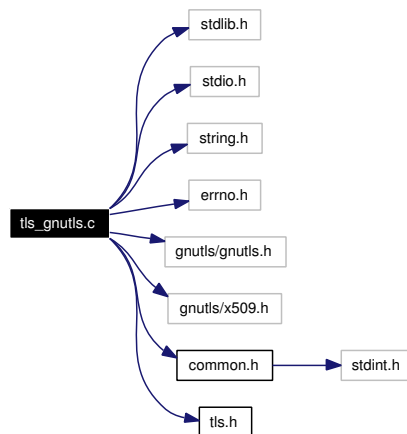
Definition at line 111 of file `tls_gnutls.c`.

6.92 tls_gnutls.c File Reference

WPA Supplicant / SSL/TLS interface functions for openssl.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.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`

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 data.
- `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_ca_cert` (`void *_ssl_ctx`, `const char *ca_cert`)
Set trusted CA certificate for all TLS connections.
- `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_global_client_cert` (`void *_ssl_ctx`, `const char *client_cert`)
Set client certificate for all TLS connections.
- `int` `tls_global_private_key` (`void *_ssl_ctx`, `const char *private_key`, `const char *private_key_passwd`)
Set private key for all TLS connections.
- `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.
- `u8 *` `tls_connection_handshake` (`void *ssl_ctx`, `struct tls_connection *conn`, `const u8 *in_data`, `size_t in_len`, `size_t *out_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_anon_dh` (void *ssl_ctx, struct tls_connection *conn)
Configure TLS connection to use anonymous DH.
- `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.

Variables

- `int wpa_debug_show_keys`

6.92.1 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](#).

6.92.2 Function Documentation

- 6.92.2.1** `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

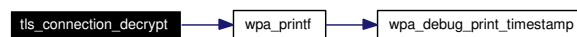
Returns:

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 691 of file `tls_gnutls.c`.

Here is the call graph for this function:

**6.92.2.2 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 239 of file `tls_gnutls.c`.

6.92.2.3 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 753 of file `tls_gnutls.c`.

6.92.2.4 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 673 of file `tls_gnutls.c`.

6.92.2.5 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 254 of file `tls_gnutls.c`.

6.92.2.6 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 772 of file `tls_gnutls.c`.

6.92.2.7 `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 495 of file `tls_gnutls.c`.

6.92.2.8 `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 780 of file `tls_gnutls.c`.

6.92.2.9 `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 788 of file `tls_gnutls.c`.

6.92.2.10 `u8* tls_connection_handshake (void * tls_ctx, struct tls_connection * conn, const u8 * in_data, size_t in_len, size_t * out_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.

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

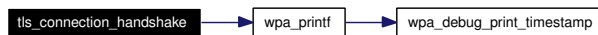
This function is used during TLS handshake. The first call is done with `in_data == NULL` and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with `in_data` pointing to the TLS 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 603 of file `tls_gnutls.c`.

Here is the call graph for this function:

**6.92.2.11 struct tls_connection* tls_connection_init (void * tls_ctx)**

Initialize a new TLS connection.

Parameters:

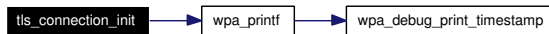
tls_ctx TLS context data from `tls_init()`

Returns:

Connection context data, `conn` for other function calls

Definition at line 207 of file `tls_gnutls.c`.

Here is the call graph for this function:

**6.92.2.12 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 719 of file `tls_gnutls.c`.

6.92.2.13 `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 663 of file `tls_gnutls.c`.

6.92.2.14 `int tls_connection_set_anon_dh (void * tls_ctx, struct tls_connection * conn)`

Configure TLS connection to use anonymous DH.

Parameters:

tls_ctx TLS context data from `tls_init()`

conn Connection context data from `tls_connection_init()`

Returns:

0 on success, -1 on failure

TODO: consider changing this to more generic routine for configuring allowed ciphers

Definition at line 737 of file `tls_gnutls.c`.

6.92.2.15 `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()`

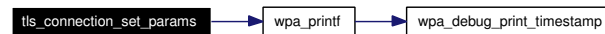
params Connection parameters

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 376 of file `tls_gnutls.c`.

Here is the call graph for this function:



6.92.2.16 `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 468 of file `tls_gnutls.c`.

6.92.2.17 `int tls_connection_shutdown (void * tls_ctx, struct tls_connection * conn)`

Shutdown TLS connection data.

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 260 of file `tls_gnutls.c`.

6.92.2.18 `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 146 of file `tls_gnutls.c`.

6.92.2.19 `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 744 of file `tls_gnutls.c`.

6.92.2.20 `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 154 of file `tls_gnutls.c`.

6.92.2.21 `int tls_global_ca_cert (void * tls_ctx, const char * ca_cert)`

Set trusted CA certificate for all TLS connections.

Parameters:

- tls_ctx* TLS context data from `tls_init()`
- ca_cert* File name for CA certificate in PEM or DER format NULL to allow all subjects

Returns:

- 0 on success, -1 on failure

Definition at line 454 of file `tls_gnutls.c`.

6.92.2.22 int tls_global_client_cert (void * *tls_ctx*, const char * *client_cert*)

Set client certificate for all TLS connections.

Parameters:

tls_ctx TLS context data from [tls_init\(\)](#)

client_cert File name for client certificate in PEM or DER format

Returns:

0 on success, -1 on failure

Definition at line 480 of file `tls_gnutls.c`.

6.92.2.23 int tls_global_private_key (void * *tls_ctx*, const char * *private_key*, const char * *private_key_passwd*)

Set private key for all TLS connections.

Parameters:

tls_ctx TLS context data from [tls_init\(\)](#)

private_key File name for client private key in PEM or DER format

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

Returns:

0 on success, -1 on failure

Definition at line 487 of file `tls_gnutls.c`.

6.92.2.24 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 461 of file `tls_gnutls.c`.

6.92.2.25 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 111 of file `tls_gnutls.c`.

Here is the call graph for this function:

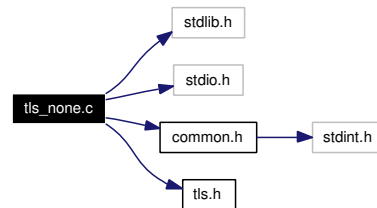


6.93 `tls_none.c` File Reference

WPA Supplicant / SSL/TLS interface functions for no TLS case.

```
#include <stdlib.h>
#include <stdio.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.

6.93.1 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](#).

6.93.2 Function Documentation

6.93.2.1 `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 27 of file `tls_none.c`.

6.93.2.2 `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.

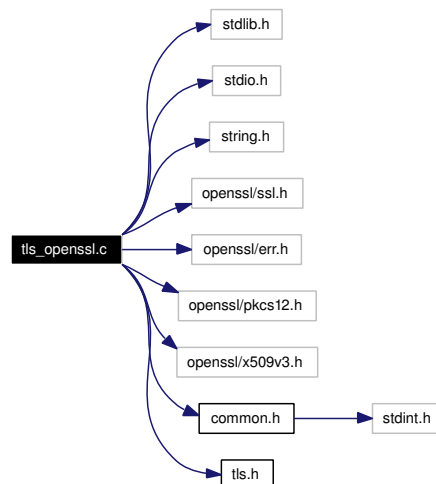
Definition at line 22 of file `tls_none.c`.

6.94 tls_openssl.c File Reference

WPA Supplicant / SSL/TLS interface functions for openssl.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.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 data.
- int `tls_global_ca_cert` (void *_ssl_ctx, const char *ca_cert)
Set trusted CA certificate for all TLS connections.
- 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_global_client_cert` (void *_ssl_ctx, const char *client_cert)
Set client certificate for all TLS connections.
- int `tls_global_private_key` (void *_ssl_ctx, const char *private_key, const char *private_key_passwd)
Set private key for all TLS connections.
- 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.
- u8 * `tls_connection_handshake` (void *ssl_ctx, struct `tls_connection` *conn, const u8 *in_data, size_t in_len, size_t *out_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_anon_dh](#) (void *ssl_ctx, struct tls_connection *conn)
Configure TLS connection to use anonymous DH.
- 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_connection_get_keyblock_size](#) (void *tls_ctx, struct tls_connection *conn)
Get TLS key_block size.

6.94.1 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](#).

6.94.2 Function Documentation

- 6.94.2.1** 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 1902 of file `tls_openssl.c`.

6.94.2.2 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 902 of file `tls_openssl.c`.

6.94.2.3 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 2019 of file `tls_openssl.c`.

6.94.2.4 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 1868 of file `tls_openssl.c`.

6.94.2.5 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 915 of file `tls_openssl.c`.

6.94.2.6 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 2048 of file `tls_openssl.c`.

6.94.2.7 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 2130 of file `tls_openssl.c`.

6.94.2.8 `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 1730 of file `tls_openssl.c`.

6.94.2.9 `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 2056 of file `tls_openssl.c`.

6.94.2.10 `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 2064 of file `tls_openssl.c`.

6.94.2.11 `u8* tls_connection_handshake (void * tls_ctx, struct tls_connection * conn, const u8 * in_data, size_t in_len, size_t * out_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.

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

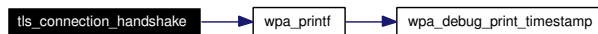
This function is used during TLS handshake. The first call is done with *in_data* == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with *in_data* pointing to the TLS 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 1753 of file `tls_openssl.c`.

Here is the call graph for this function:

**6.94.2.12 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 855 of file `tls_openssl.c`.

6.94.2.13 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 1932 of file `tls_openssl.c`.

6.94.2.14 `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.

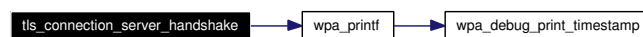
Returns:

pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

Definition at line 1817 of file `tls_openssl.c`.

Here is the call graph for this function:

**6.94.2.15** `int tls_connection_set_anon_dh (void * tls_ctx, struct tls_connection * conn)`

Configure TLS connection to use anonymous DH.

Parameters:

- tls_ctx* TLS context data from `tls_init()`
- conn* Connection context data from `tls_connection_init()`

Returns:

0 on success, -1 on failure

TODO: consider changing this to more generic routine for configuring allowed ciphers

Definition at line 1988 of file `tls_openssl.c`.

6.94.2.16 `int tls_connection_set_params (void * tls_ctx, struct tls_connection * conn, const struct tls_connection_params * params)`

Set TLS connection parameters.

Parameters:

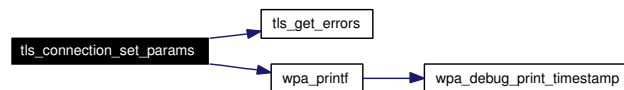
- tls_ctx* TLS context data from `tls_init()`
- conn* Connection context data from `tls_connection_init()`
- params* Connection parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRIV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRIV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

Definition at line 2072 of file tls_openssl.c.

Here is the call graph for this function:

**6.94.2.17 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 1206 of file tls_openssl.c.

6.94.2.18 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection data.

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 921 of file tls_openssl.c.

6.94.2.19 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 742 of file `tls_openssl.c`.

6.94.2.20 `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 2003 of file `tls_openssl.c`.

6.94.2.21 `int tls_get_errors (void * tls_ctx)`

Process pending errors.

Parameters:

tls_ctx TLS context data from `tls_init()`

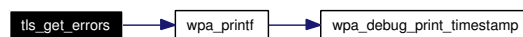
Returns:

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

Definition at line 841 of file `tls_openssl.c`.

Here is the call graph for this function:



6.94.2.22 `int tls_global_ca_cert (void * tls_ctx, const char * ca_cert)`

Set trusted CA certificate for all TLS connections.

Parameters:

tls_ctx TLS context data from `tls_init()`

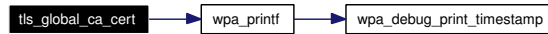
ca_cert File name for CA certificate in PEM or DER format NULL to allow all subjects

Returns:

0 on success, -1 on failure

Definition at line 1135 of file `tls_openssl.c`.

Here is the call graph for this function:

**6.94.2.23 int tls_global_client_cert (void * *tls_ctx*, const char * *client_cert*)**

Set client certificate for all TLS connections.

Parameters:

tls_ctx TLS context data from `tls_init()`

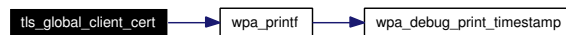
client_cert File name for client certificate in PEM or DER format

Returns:

0 on success, -1 on failure

Definition at line 1277 of file `tls_openssl.c`.

Here is the call graph for this function:

**6.94.2.24 int tls_global_private_key (void * *tls_ctx*, const char * *private_key*, const char * *private_key_passwd*)**

Set private key for all TLS connections.

Parameters:

tls_ctx TLS context data from `tls_init()`

private_key File name for client private key in PEM or DER format

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

Returns:

0 on success, -1 on failure

Definition at line 1609 of file `tls_openssl.c`.

6.94.2.25 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 1160 of file `tls_openssl.c`.

6.94.2.26 void* `tls_init` (const struct `tls_config` * *conf*)

Initialize TLS library.

Parameters:

conf Configuration data for TLS library

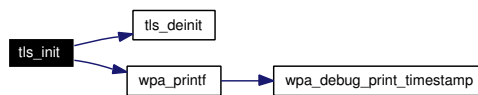
Returns:

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 698 of file `tls_openssl.c`.

Here is the call graph for this function:

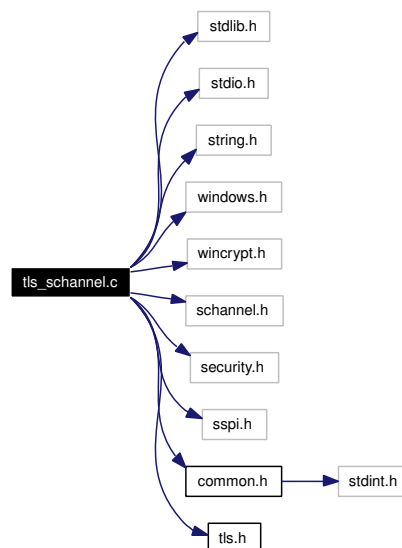


6.95 tls_schannel.c File Reference

WPA Supplicant / SSL/TLS interface functions for Microsoft Schannel.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.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_connection`

Defines

- #define `SECURITY_WIN32`
- #define `SECPKG_ATTR_EAP_KEY_BLOCK` 0x5b

Typedefs

- typedef `_SecPkgContext_EapKeyBlock` `SecPkgContext_EapKeyBlock`

- typedef `_SecPkgContext_EapKeyBlock` * **P**SecPkgContext_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 data.
- int `tls_global_ca_cert` (void *_ssl_ctx, const char *ca_cert)
Set trusted CA certificate for all TLS connections.
- 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_global_client_cert` (void *_ssl_ctx, const char *client_cert)
Set client certificate for all TLS connections.
- int `tls_global_private_key` (void *_ssl_ctx, const char *private_key, const char *private_key_passwd)
Set private key for all TLS connections.
- 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.
- u8 * `tls_connection_handshake` (void *ssl_ctx, struct `tls_connection` *conn, const u8 *in_data, size_t in_len, size_t *out_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_anon_dh](#) (void *ssl_ctx, struct tls_connection *conn)
Configure TLS connection to use anonymous DH.
- 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.

6.95.1 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](#).

6.95.2 Function Documentation

6.95.2.1 `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

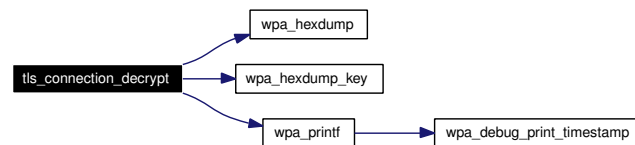
Returns:

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 551 of file `tls_schannel.c`.

Here is the call graph for this function:



6.95.2.2 `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 140 of file `tls_schannel.c`.

6.95.2.3 `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 660 of file `tls_schannel.c`.

6.95.2.4 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

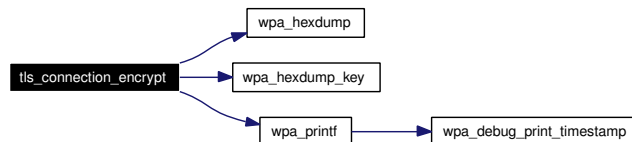
Returns:

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 463 of file `tls_schannel.c`.

Here is the call graph for this function:



6.95.2.5 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 149 of file `tls_schannel.c`.

6.95.2.6 `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 680 of file `tls_schannel.c`.

6.95.2.7 `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 203 of file `tls_schannel.c`.

6.95.2.8 `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 688 of file `tls_schannel.c`.

6.95.2.9 `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 696 of file `tls_schannel.c`.

6.95.2.10 `u8* tls_connection_handshake (void * tls_ctx, struct tls_connection * conn, const u8 * in_data, size_t in_len, size_t * out_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.

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

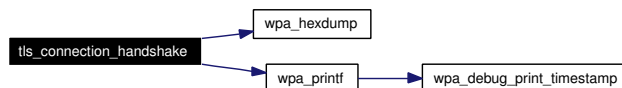
This function is used during TLS handshake. The first call is done with `in_data == NULL` and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with `in_data` pointing to the TLS 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 319 of file `tls_schannel.c`.

Here is the call graph for this function:



6.95.2.11 `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 126 of file `tls_schannel.c`.

6.95.2.12 `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 632 of file `tls_schannel.c`.

6.95.2.13 `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 454 of file `tls_schannel.c`.

6.95.2.14 `int tls_connection_set_anon_dh (void *tls_ctx, struct tls_connection *conn)`

Configure TLS connection to use anonymous DH.

Parameters:

tls_ctx TLS context data from `tls_init()`
conn Connection context data from `tls_connection_init()`

Returns:

0 on success, -1 on failure

TODO: consider changing this to more generic routine for configuring allowed ciphers

Definition at line 647 of file `tls_schannel.c`.

6.95.2.15 `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\(\)](#)

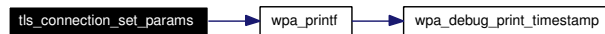
params Connection parameters

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 704 of file `tls_schannel.c`.

Here is the call graph for this function:



6.95.2.16 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 183 of file `tls_schannel.c`.

6.95.2.17 int tls_connection_shutdown (void * *tls_ctx*, struct `tls_connection` * *conn*)

Shutdown TLS connection data.

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 155 of file `tls_schannel.c`.

6.95.2.18 `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 109 of file `tls_schannel.c`.

6.95.2.19 `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 653 of file `tls_schannel.c`.

6.95.2.20 `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 120 of file `tls_schannel.c`.

6.95.2.21 `int tls_global_ca_cert (void * tls_ctx, const char * ca_cert)`

Set trusted CA certificate for all TLS connections.

Parameters:

tls_ctx TLS context data from `tls_init()`

ca_cert File name for CA certificate in PEM or DER format NULL to allow all subjects

Returns:

0 on success, -1 on failure

Definition at line 171 of file `tls_schannel.c`.

6.95.2.22 int tls_global_client_cert (void * *tls_ctx*, const char * *client_cert*)

Set client certificate for all TLS connections.

Parameters:

tls_ctx TLS context data from `tls_init()`

client_cert File name for client certificate in PEM or DER format

Returns:

0 on success, -1 on failure

Definition at line 190 of file `tls_schannel.c`.

6.95.2.23 int tls_global_private_key (void * *tls_ctx*, const char * *private_key*, const char * *private_key_passwd*)

Set private key for all TLS connections.

Parameters:

tls_ctx TLS context data from `tls_init()`

private_key File name for client private key in PEM or DER format

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

Returns:

0 on success, -1 on failure

Definition at line 196 of file `tls_schannel.c`.

6.95.2.24 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 177 of file `tls_schannel.c`.

6.95.2.25 `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 93 of file `tls_schannel.c`.

6.96 win_if_list.c File Reference

win_if_list - Display network interfaces with description (for Windows)

```
#include "pcap.h"
```

```
#include <winsock.h>
```

Include dependency graph for win_if_list.c:



Functions

- `int main (int argc, char *argv[])`

6.96.1 Detailed Description

win_if_list - Display network interfaces with description (for Windows)

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 small tool is for the Windows build to provide an easy way of fetching a list of available network interfaces.

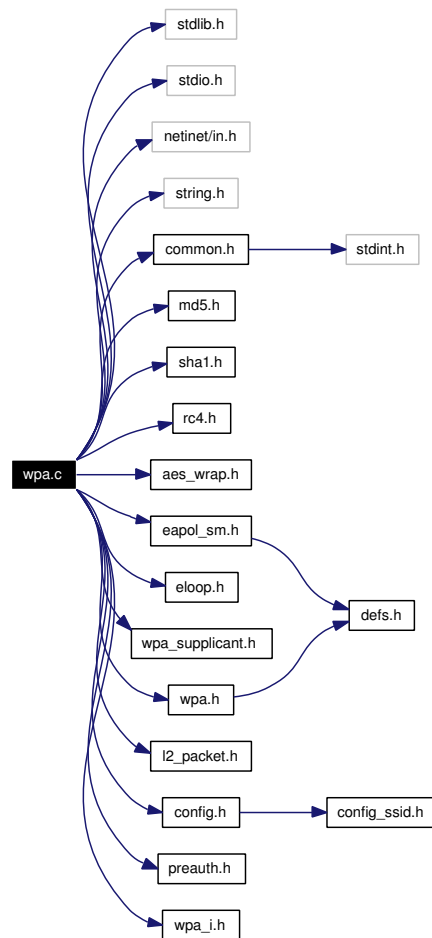
Definition in file [win_if_list.c](#).

6.97 wpa.c File Reference

WPA Supplicant - WPA state machine and EAPOL-Key processing.

```
#include <stdlib.h>
#include <stdio.h>
#include <netinet/in.h>
#include <string.h>
#include "common.h"
#include "md5.h"
#include "sha1.h"
#include "rc4.h"
#include "aes_wrap.h"
#include "wpa.h"
#include "eloop.h"
#include "wpa_supplicant.h"
#include "config.h"
#include "l2_packet.h"
#include "eapol_sm.h"
#include "preauth.h"
#include "wpa_i.h"
```

Include dependency graph for wpa.c:



Defines

- #define **WPA_KEY_INFO_TYPE_MASK** (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 **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.
- 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 configuration 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)
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](#) **packed**

6.97.1 Detailed Description

WPA Supplicant - WPA state machine and EAPOL-Key processing.

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.c](#).

6.97.2 Function Documentation

6.97.2.1 `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 504 of file wpa.c.

6.97.2.2 `void wpa_sm_aborted_cached (struct wpa_sm * sm)`

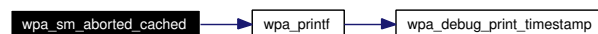
Notify WPA that PMKSA caching was aborted.

Parameters:

- sm* Pointer to WPA state machine data from [wpa_sm_init\(\)](#)

Definition at line 1905 of file wpa.c.

Here is the call graph for this function:



6.97.2.3 `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 2296 of file wpa.c.

Here is the call graph for this function:



6.97.2.4 `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 2200 of file wpa.c.

6.97.2.5 `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 2536 of file wpa.c.

6.97.2.6 `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 2575 of file wpa.c.

6.97.2.7 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 2272 of file wpa.c.

6.97.2.8 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

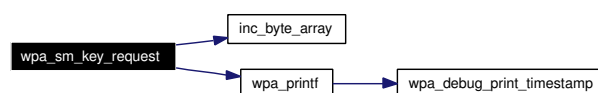
Returns:

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 796 of file wpa.c.

Here is the call graph for this function:

**6.97.2.9 void wpa_sm_notify_assoc (struct wpa_sm * sm, const u8 * bssid)**

Notify WPA state machine about association.

Parameters:

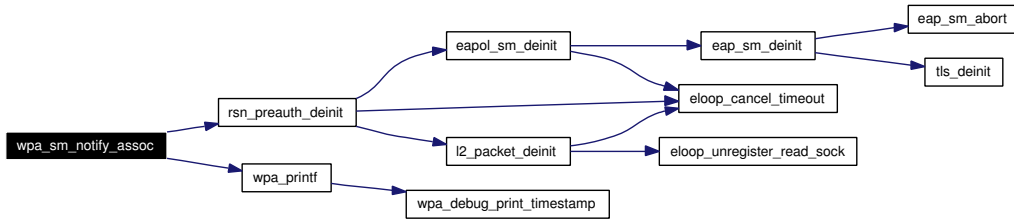
sm Pointer to WPA state machine data from [wpa_sm_init\(\)](#)

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 2318 of file wpa.c.

Here is the call graph for this function:



6.97.2.10 void wpa_sm_notify_disassoc (struct wpa_sm * sm)

Notify WPA state machine about disassociation.

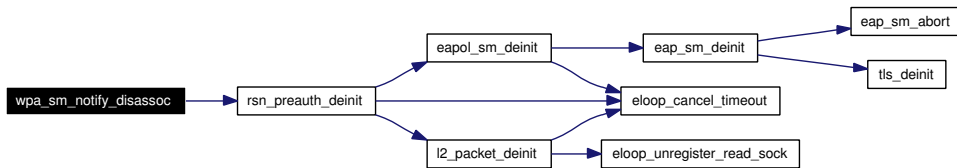
Parameters:

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 2341 of file wpa.c.

Here is the call graph for this function:



6.97.2.11 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

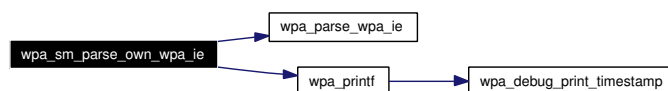
Returns:

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 2753 of file wpa.c.

Here is the call graph for this function:



6.97.2.12 `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:

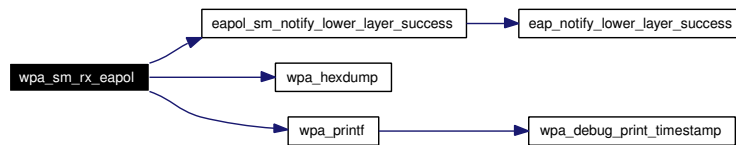
- 1 = WPA EAPOL-Key processed, 0 = not a WPA EAPOL-Key, -1 failure

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 1965 of file wpa.c.

Here is the call graph for this function:

**6.97.2.13** `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

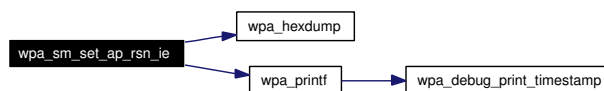
Returns:

- 0 on success, -1 on failure

Inform WPA state machine about the RSN IE used in Beacon / Probe Response frame.

Definition at line 2719 of file wpa.c.

Here is the call graph for this function:



6.97.2.14 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

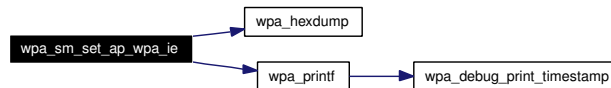
Returns:

- 0 on success, -1 on failure

Inform WPA state machine about the WPA IE used in Beacon / Probe Response frame.

Definition at line 2684 of file wpa.c.

Here is the call graph for this function:



6.97.2.15 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

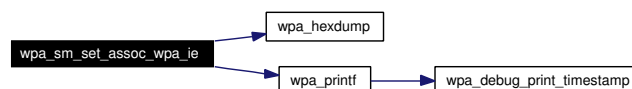
Returns:

- 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 2649 of file wpa.c.

Here is the call graph for this function:



6.97.2.16 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

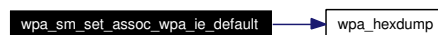
Returns:

- 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 2603 of file wpa.c.

Here is the call graph for this function:

**6.97.2.17 void wpa_sm_set_config (struct [wpa_sm](#) * *sm*, struct [wpa_ssid](#) * *config*)**

Notification of current configuration 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 pointer.

Definition at line 2430 of file wpa.c.

6.97.2.18 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 2469 of file wpa.c.

6.97.2.19 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 2397 of file wpa.c.

6.97.2.20 void `wpa_sm_set_ifname` (struct `wpa_sm` * *sm*, const char * *ifname*)

Set network interface name.

Parameters:

sm Pointer to WPA state machine data from `wpa_sm_init()`

ifname Interface name

Definition at line 2456 of file wpa.c.

6.97.2.21 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 2443 of file wpa.c.

6.97.2.22 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 2484 of file wpa.c.

6.97.2.23 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 2358 of file wpa.c.

6.97.2.24 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 2376 of file wpa.c.

6.97.2.25 void wpa_sm_set_scared_ctx (struct wpa_sm * sm, void * scared_ctx)

Set context pointer for smartcard callbacks.

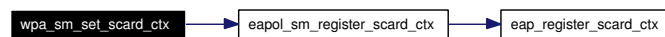
Parameters:

sm Pointer to WPA state machine data from [wpa_sm_init\(\)](#)

scared_ctx Context pointer for smartcard related callback functions

Definition at line 2410 of file wpa.c.

Here is the call graph for this function:



6.98 wpa.h File Reference

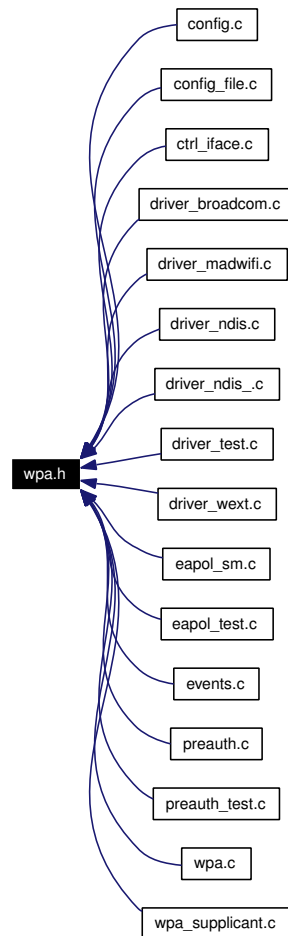
[wpa_supplicant](#) - WPA definitions

```
#include "defs.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 **EAPOL_VERSION** 2
- #define **WPA_CAPABILITY_PREAUTH** BIT(0)
- #define **GENERIC_INFO_ELEM** 0xdd
- #define **RSN_INFO_ELEM** 0x30
- #define **PMKID_LEN** 16

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 }
- 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** }

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_scared_ctx](#) (struct [wpa_sm](#) *sm, void *scared_ctx)

Set context pointer for smartcard callbacks.

- void `wpa_sm_set_config` (struct `wpa_sm` *sm, struct `wpa_ssid` *config)
Notification of current configuration 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)
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_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.

Variables

- `ieee802_1x_hdr` **packed**

6.98.1 Detailed Description

[wpa_supplicant](#) - WPA 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 [wpa.h](#).

6.98.2 Function Documentation

6.98.2.1 `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 504 of file [wpa.c](#).

6.98.2.2 `void wpa_sm_aborted_cached` (`struct wpa_sm * sm`)

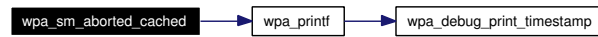
Notify WPA that PMKSA caching was aborted.

Parameters:

- sm* Pointer to WPA state machine data from [wpa_sm_init\(\)](#)

Definition at line 1905 of file wpa.c.

Here is the call graph for this function:



6.98.2.3 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 2296 of file wpa.c.

Here is the call graph for this function:



6.98.2.4 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 2200 of file wpa.c.

6.98.2.5 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 2536 of file wpa.c.

6.98.2.6 `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 2575 of file wpa.c.

6.98.2.7 `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 2272 of file wpa.c.

6.98.2.8 `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

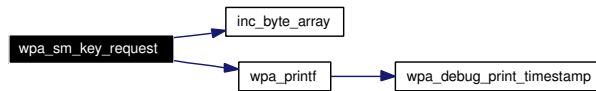
Returns:

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 796 of file wpa.c.

Here is the call graph for this function:



6.98.2.9 void wpa_sm_notify_assoc (struct wpa_sm * sm, const u8 * bssid)

Notify WPA state machine about association.

Parameters:

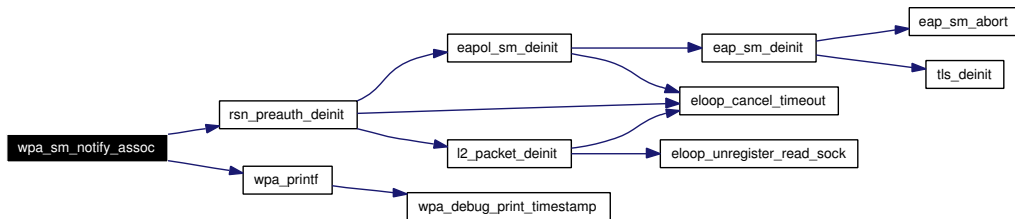
sm Pointer to WPA state machine data from [wpa_sm_init\(\)](#)

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 2318 of file wpa.c.

Here is the call graph for this function:



6.98.2.10 void wpa_sm_notify_disassoc (struct wpa_sm * sm)

Notify WPA state machine about disassociation.

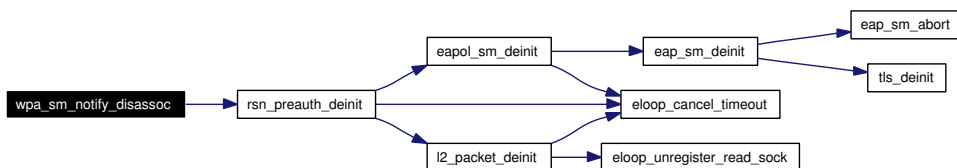
Parameters:

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 2341 of file wpa.c.

Here is the call graph for this function:



6.98.2.11 `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

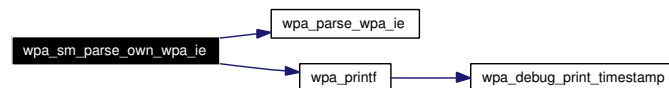
Returns:

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 2753 of file wpa.c.

Here is the call graph for this function:

**6.98.2.12** `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:

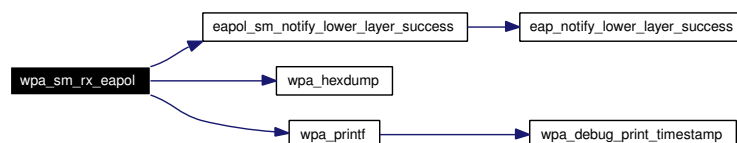
1 = WPA EAPOL-Key processed, 0 = not a WPA EAPOL-Key, -1 failure

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 1965 of file wpa.c.

Here is the call graph for this function:



6.98.2.13 `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

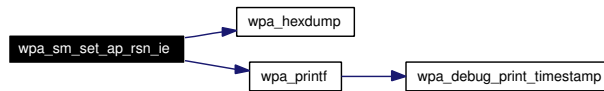
Returns:

- 0 on success, -1 on failure

Inform WPA state machine about the RSN IE used in Beacon / Probe Response frame.

Definition at line 2719 of file wpa.c.

Here is the call graph for this function:

**6.98.2.14** `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

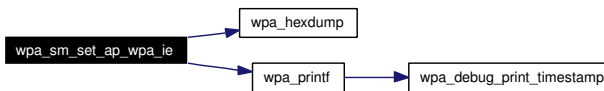
Returns:

- 0 on success, -1 on failure

Inform WPA state machine about the WPA IE used in Beacon / Probe Response frame.

Definition at line 2684 of file wpa.c.

Here is the call graph for this function:

**6.98.2.15** `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

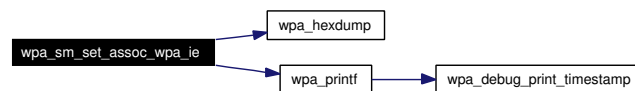
Returns:

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 2649 of file wpa.c.

Here is the call graph for this function:



6.98.2.16 `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

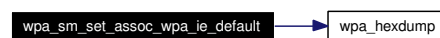
Returns:

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 2603 of file wpa.c.

Here is the call graph for this function:



6.98.2.17 `void wpa_sm_set_config (struct wpa_sm * sm, struct wpa_ssid * config)`

Notification of current configuration 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 pointer.

Definition at line 2430 of file wpa.c.

6.98.2.18 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 2469 of file wpa.c.

6.98.2.19 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 2397 of file wpa.c.

6.98.2.20 void wpa_sm_set_ifname (struct wpa_sm * sm, const char * ifname)

Set network interface name.

Parameters:

- sm* Pointer to WPA state machine data from [wpa_sm_init\(\)](#)
- ifname* Interface name

Definition at line 2456 of file wpa.c.

6.98.2.21 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 2443 of file wpa.c.

6.98.2.22 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 2484 of file wpa.c.

6.98.2.23 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 2358 of file wpa.c.

6.98.2.24 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 2376 of file wpa.c.

6.98.2.25 void wpa_sm_set_scard_ctx (struct wpa_sm * sm, void * scard_ctx)

Set context pointer for smartcard callbacks.

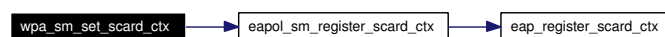
Parameters:

sm Pointer to WPA state machine data from [wpa_sm_init\(\)](#)

scard_ctx Context pointer for smartcard related callback functions

Definition at line 2410 of file wpa.c.

Here is the call graph for this function:

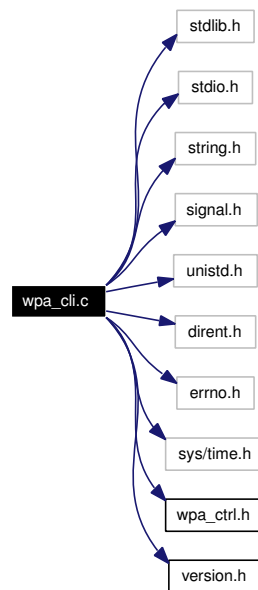


6.99 wpa_cli.c File Reference

WPA Supplicant - command line interface for [wpa_supplicant](#) daemon.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <signal.h>
#include <unistd.h>
#include <dirent.h>
#include <errno.h>
#include <sys/time.h>
#include "wpa_ctrl.h"
#include "version.h"
```

Include dependency graph for wpa_cli.c:



Defines

- #define **max_args** 10

Functions

- int **main** (int argc, char *argv[])

6.99.1 Detailed Description

WPA Supplicant - command line interface for [wpa_supplicant](#) daemon.

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.

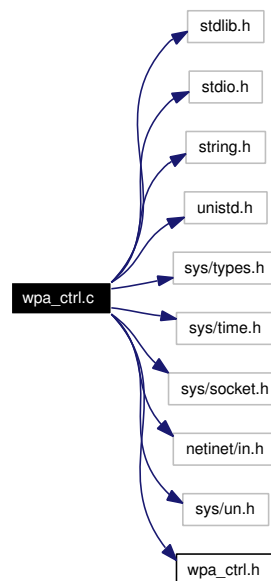
Definition in file [wpa_cli.c](#).

6.100 wpa_ctrl.c File Reference

wpa_supplicant/hostapd control interface library

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/un.h>
#include "wpa_ctrl.h"
```

Include dependency graph for wpa_ctrl.c:



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.

6.100.1 Detailed Description

wpa_supplicant/hostapd control interface library

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 [wpa_ctrl.c](#).

6.100.2 Function Documentation

6.100.2.1 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 200 of file wpa_ctrl.c.

6.100.2.2 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.

Definition at line 128 of file wpa_ctrl.c.

6.100.2.3 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 206 of file wpa_ctrl.c.

6.100.2.4 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.

Definition at line 238 of file wpa_ctrl.c.

6.100.2.5 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.

Definition at line 56 of file wpa_ctrl.c.

6.100.2.6 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:

Non-zero if there are pending messages

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.

Definition at line 224 of file wpa_ctrl.c.

6.100.2.7 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.

Definition at line 212 of file wpa_ctrl.c.

6.100.2.8 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.

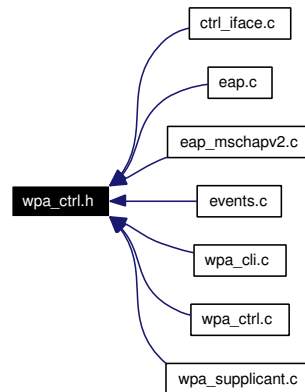
msg_cb can be used to register a callback function that will be called for unsolicited messages received while waiting for the command response. These messages may be received if [wpa_ctrl_request\(\)](#) is called at the same time as wpa_supplicant/hostapd is sending such a message. This can happen 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.

Definition at line 138 of file wpa_ctrl.c.

6.101 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.

6.101.1 Detailed Description

wpa_supplicant/hostapd control interface library

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 [wpa_ctrl.h](#).

6.101.2 Define Documentation

6.101.2.1 `#define WPA_CTRL_REQ "CTRL-REQ"`

Interactive request for identity/password/pin

Definition at line 26 of file [wpa_ctrl.h](#).

6.101.2.2 `#define WPA_CTRL_RSP "CTRL-RSP"`

Response to identity/password/pin request

Definition at line 29 of file [wpa_ctrl.h](#).

6.101.2.3 `#define WPA_EVENT_CONNECTED "CTRL-EVENT-CONNECTED "`

Authentication completed successfully and data connection enabled

Definition at line 33 of file [wpa_ctrl.h](#).

6.101.2.4 `#define WPA_EVENT_DISCONNECTED "CTRL-EVENT-DISCONNECTED "`

Disconnected, data connection is not available

Definition at line 35 of file [wpa_ctrl.h](#).

6.101.2.5 #define WPA_EVENT_EAP_FAILURE "CTRL-EVENT-EAP-FAILURE "

EAP authentication failed (EAP-Failure received)

Definition at line 49 of file wpa_ctrl.h.

6.101.2.6 #define WPA_EVENT_EAP_METHOD "CTRL-EVENT-EAP-METHOD "

EAP method selected

Definition at line 45 of file wpa_ctrl.h.

6.101.2.7 #define WPA_EVENT_EAP_NOTIFICATION "CTRL-EVENT-EAP-NOTIFICATION "

EAP-Request/Notification received

Definition at line 41 of file wpa_ctrl.h.

6.101.2.8 #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.

6.101.2.9 #define WPA_EVENT_EAP_SUCCESS "CTRL-EVENT-EAP-SUCCESS "

EAP authentication completed successfully

Definition at line 47 of file wpa_ctrl.h.

6.101.2.10 #define WPA_EVENT_PASSWORD_CHANGED "CTRL-EVENT-PASSWORD-CHANGED "

Password change was completed successfully

Definition at line 39 of file wpa_ctrl.h.

6.101.2.11 #define WPA_EVENT_TERMINATING "CTRL-EVENT-TERMINATING "

[wpa_supplicant](#) is exiting

Definition at line 37 of file wpa_ctrl.h.

6.101.3 Function Documentation**6.101.3.1 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 200 of file wpa_ctrl.c.

6.101.3.2 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.

Definition at line 128 of file wpa_ctrl.c.

6.101.3.3 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 206 of file wpa_ctrl.c.

6.101.3.4 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.

Definition at line 238 of file wpa_ctrl.c.

6.101.3.5 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.

Definition at line 56 of file `wpa_ctrl.c`.

6.101.3.6 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:

Non-zero if there are pending messages

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.

Definition at line 224 of file `wpa_ctrl.c`.

6.101.3.7 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.

Definition at line 212 of file `wpa_ctrl.c`.

6.101.3.8 `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.

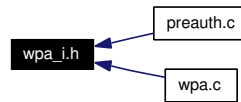
msg_cb can be used to register a callback function that will be called for unsolicited messages received while waiting for the command response. These messages may be received if [wpa_ctrl_request\(\)](#) is called at the same time as wpa_supplicant/hostapd is sending such a message. This can happen 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.

Definition at line 138 of file [wpa_ctrl.c](#).

6.102 wpa_i.h File Reference

[wpa_supplicant](#) - Internal WPA state machine definitions

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



Defines

- `#define WPA_NONCE_LEN 32`
- `#define WPA_REPLAY_COUNTER_LEN 8`

Variables

- [wpa_ptk packed](#)
WPA Pairwise Transient Key.

6.102.1 Detailed Description

[wpa_supplicant](#) - Internal WPA state machine definitions

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 [wpa_i.h](#).

6.102.2 Variable Documentation

6.102.2.1 struct [wpa_ptk packed](#)

WPA Pairwise Transient Key.

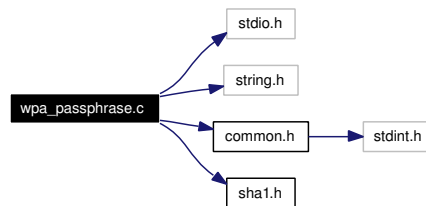
IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy

6.103 wpa_passphrase.c File Reference

WPA Supplicant - ASCII passphrase to WPA PSK tool.

```
#include <stdio.h>
#include <string.h>
#include "common.h"
#include "sha1.h"
```

Include dependency graph for wpa_passphrase.c:



Functions

- int **main** (int argc, char *argv[])

6.103.1 Detailed Description

WPA Supplicant - ASCII passphrase to WPA PSK tool.

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.

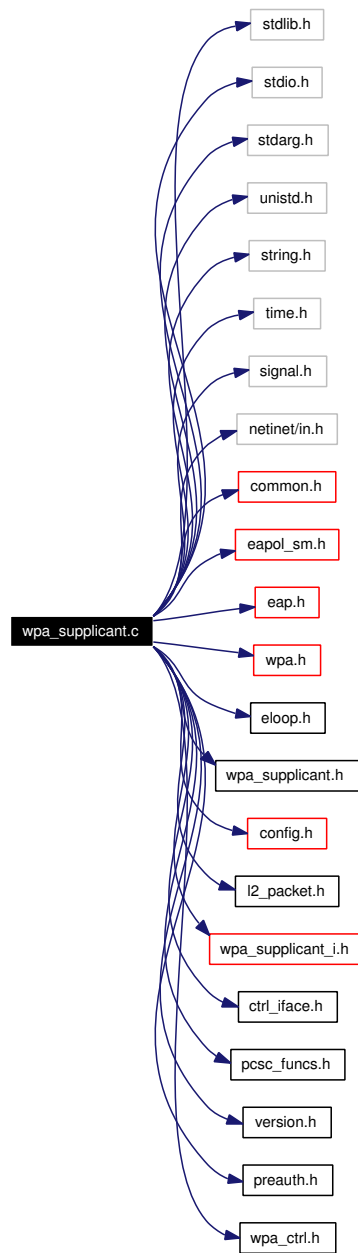
Definition in file [wpa_passphrase.c](#).

6.104 wpa_supplicant.c File Reference

WPA Supplicant.

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <signal.h>
#include <netinet/in.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 "pcsc_funcs.h"
#include "version.h"
#include "preauth.h"
#include "wpa_ctrl.h"
```

Include dependency graph for wpa_supplicant.c:



Defines

- `#define SCAN_AP_LIMIT 128`

Functions

- void `wpa_msg` (struct `wpa_supplicant` *wpa_s, int level, char *fmt,...)
- `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.
- const char * `wpa_ssid_txt` (u8 *ssid, size_t ssid_len)
Convert SSID to a printable string.
- 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_license](#)
- [wpa_driver_ops](#) * [wpa_supplicant_drivers](#) []
- int [wpa_debug_level](#)
- int [wpa_debug_show_keys](#)
- int [wpa_debug_timestamp](#)

6.104.1 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](#).

6.104.2 Function Documentation

6.104.2.1 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:

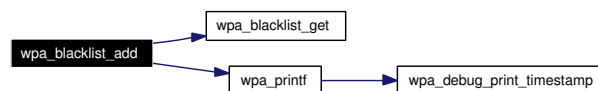
0 on success, -1 on failure

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 433 of file [wpa_supplicant.c](#).

Here is the call graph for this function:



6.104.2.2 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 490 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.3 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 400 of file wpa_supplicant.c.

6.104.2.4 void wpa_clear_keys (struct wpa_supplicant * wpa_s, const u8 * addr)

Clear keys configured for the driver.

Parameters:

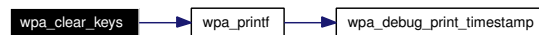
wpa_s Pointer to [wpa_supplicant](#) data

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 767 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.5 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 521 of file wpa_supplicant.c.

6.104.2.6 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

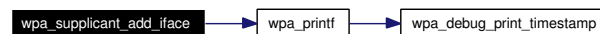
Returns:

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 2211 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.7 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

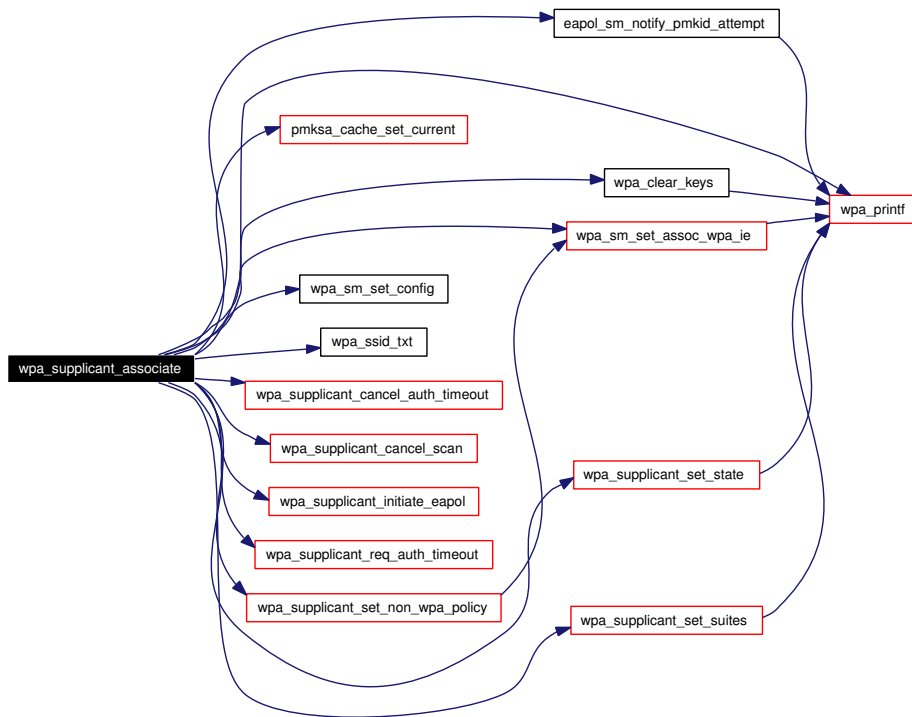
bss Scan results for the selected BSS, or NULL if not available

ssid Configuration data for the selected network

This function is used to request wpa_supplicant to associate with a BSS.

Definition at line 1276 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.8 void wpa_supplicant_cancel_auth_timeout (struct [wpa_supplicant](#) * *wpa_s*)

Cancel authentication timeout.

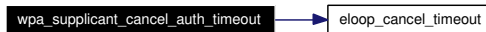
Parameters:

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 618 of file `wpa_supplicant.c`.

Here is the call graph for this function:



6.104.2.9 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 565 of file `wpa_supplicant.c`.

Here is the call graph for this function:



6.104.2.10 void wpa_supplicant_deauthenticate (struct wpa_supplicant * wpa_s, int reason_code)

Deauthenticate the current connection.

Parameters:

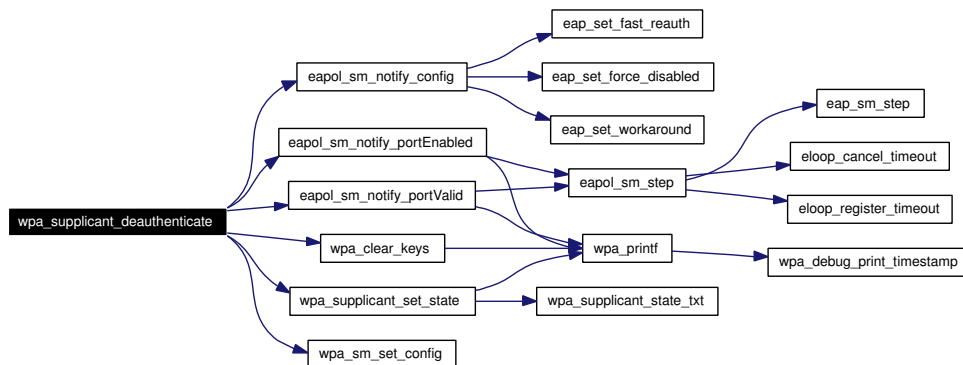
wpa_s Pointer to [wpa_supplicant](#) data

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 1503 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.11 void wpa_supplicant_deinit (struct wpa_global * global)

Deinitialize wpa_supplicant.

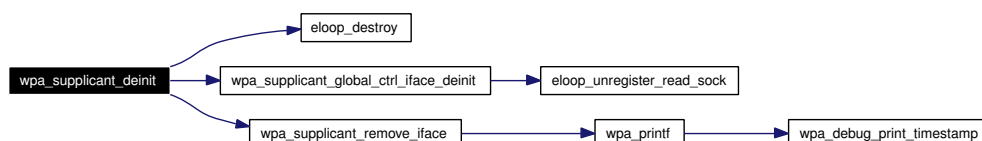
Parameters:

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 2395 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.12 void wpa_supplicant_disassociate (struct wpa_supplicant * wpa_s, int reason_code)

Disassociate the current connection.

Parameters:

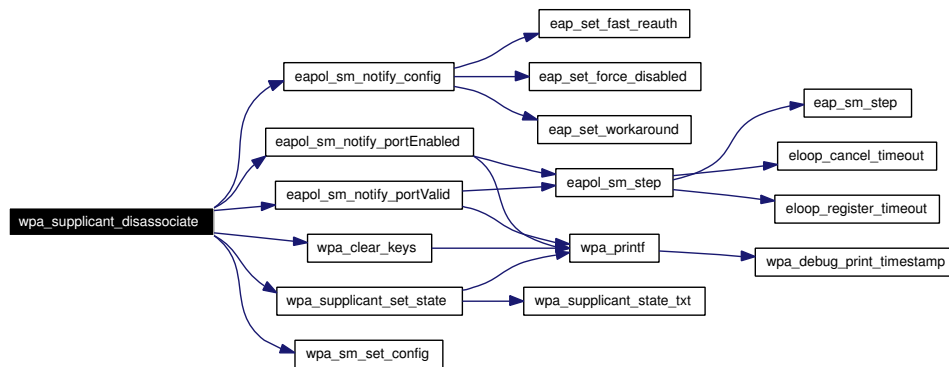
wpa_s Pointer to [wpa_supplicant](#) data

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 1476 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.13 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

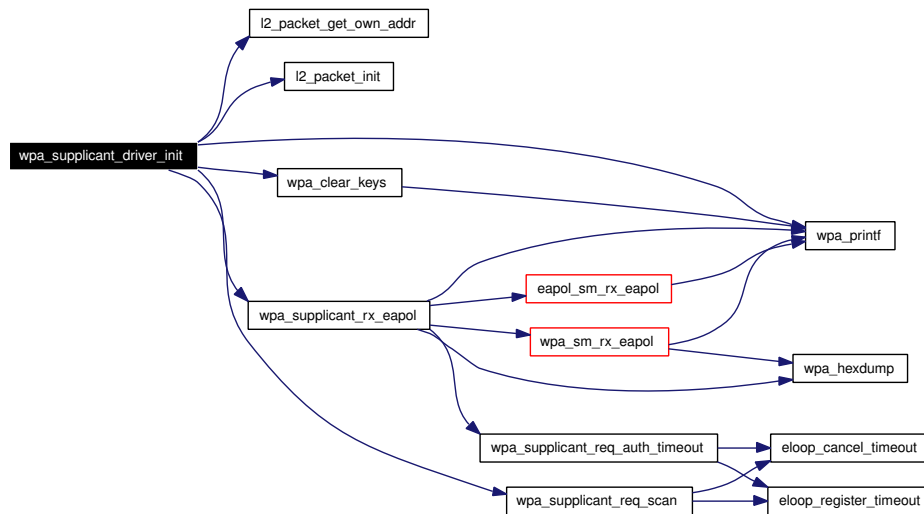
Returns:

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 1837 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.14 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 2288 of file `wpa_supplicant.c`.

6.104.2.15 int `wpa_supplicant_get_scan_results` (struct `wpa_supplicant *` `wpa_s`)

Get scan results.

Parameters:

- `wpa_s` Pointer to `wpa_supplicant` data

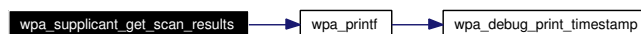
Returns:

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 1530 of file `wpa_supplicant.c`.

Here is the call graph for this function:



6.104.2.16 struct `wpa_ssid*` `wpa_supplicant_get_ssid` (struct `wpa_supplicant *` `wpa_s`)

Get a pointer to the current network structure.

Parameters:

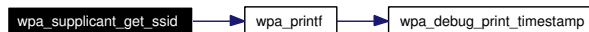
`wpa_s` Pointer to `wpa_supplicant` data

Returns:

A pointer to the current network structure or NULL on failure

Definition at line 1626 of file `wpa_supplicant.c`.

Here is the call graph for this function:



6.104.2.17 `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 862 of file `wpa_supplicant.c`.

6.104.2.18 struct `wpa_global*` `wpa_supplicant_init` (struct `wpa_params *` `params`)

Initialize `wpa_supplicant`.

Parameters:

`params` Parameters for `wpa_supplicant`

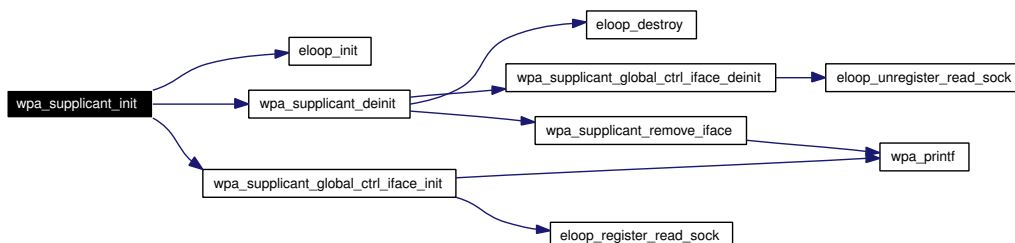
Returns:

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 2311 of file `wpa_supplicant.c`.

Here is the call graph for this function:



6.104.2.19 void wpa_supplicant_initiate_eapol (struct wpa_supplicant * wpa_s)

Configure EAPOL state machine.

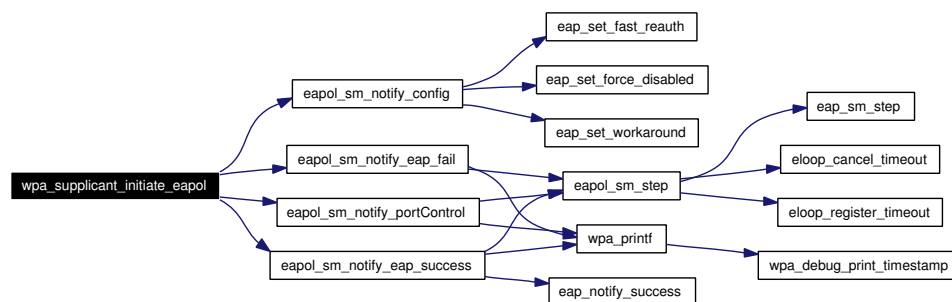
Parameters:

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 634 of file [wpa_supplicant.c](#).

Here is the call graph for this function:

**6.104.2.20 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 893 of file [wpa_supplicant.c](#).

Here is the call graph for this function:

6.104.2.22 void wpa_supplicant_req_auth_timeout (struct wpa_supplicant * wpa_s, int sec, int usec)

Schedule a timeout for authentication.

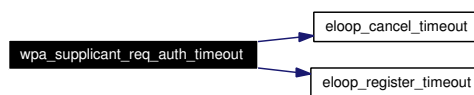
Parameters:

- 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 595 of file wpa_supplicant.c.

Here is the call graph for this function:

**6.104.2.23 void wpa_supplicant_req_scan (struct wpa_supplicant * wpa_s, int sec, int usec)**

Schedule a scan for neighboring access points.

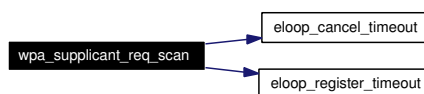
Parameters:

- 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 548 of file wpa_supplicant.c.

Here is the call graph for this function:

**6.104.2.24 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\(\)](#)

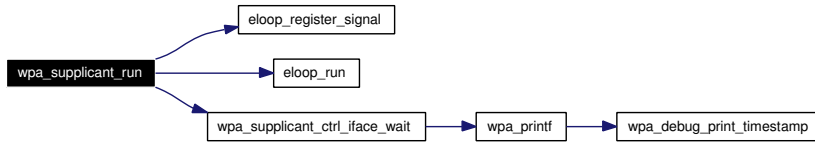
Returns:

- 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 is still in use.

Definition at line 2362 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.25 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

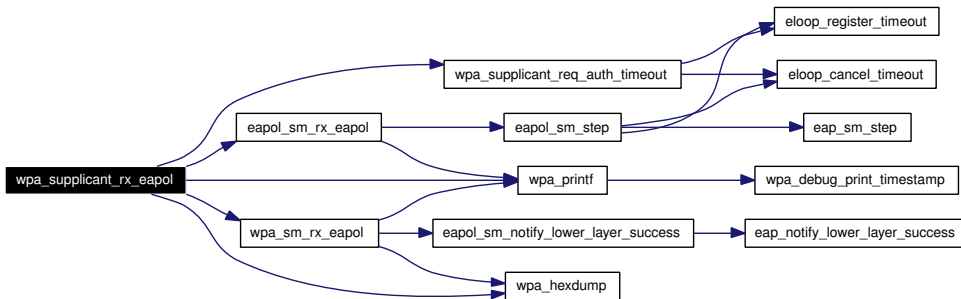
buf EAPOL data starting from the EAPOL header (i.e., no Ethernet header)

len Length of the EAPOL data

This function is called for each received EAPOL frame.

Definition at line 1780 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.26 void wpa_supplicant_set_non_wpa_policy (struct wpa_supplicant * wpa_s, struct wpa_ssid * ssid)

Set WPA parameters to non-WPA mode.

Parameters:

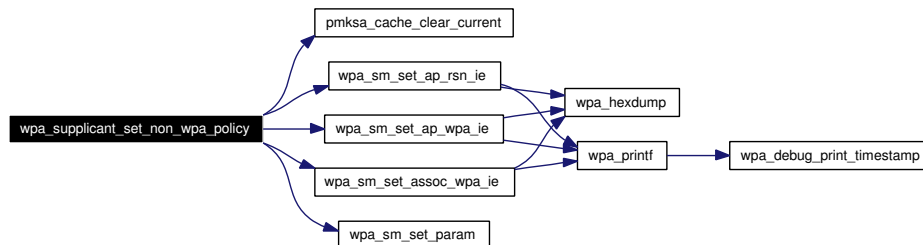
wpa_s Pointer to [wpa_supplicant](#) data

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 684 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.27 void wpa_supplicant_set_state (struct wpa_supplicant * wpa_s, wpa_states state)

Set current connection state.

Parameters:

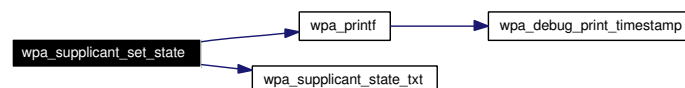
wpa_s Pointer to [wpa_supplicant](#) data

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 836 of file wpa_supplicant.c.

Here is the call graph for this function:



6.104.2.28 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.

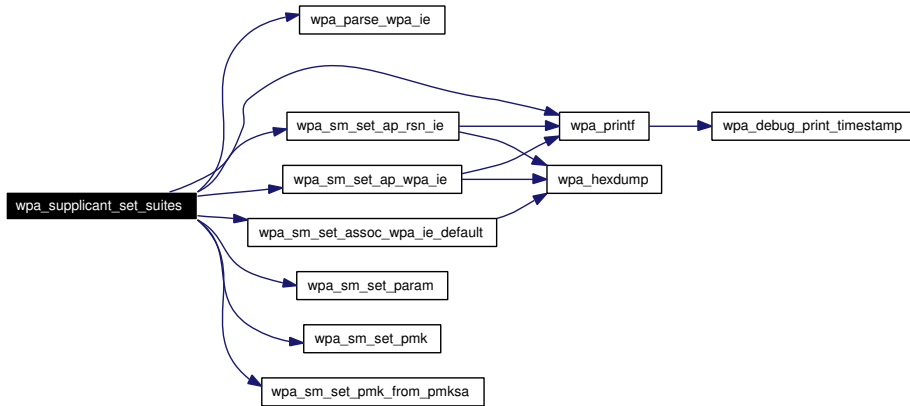
Returns:

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 1146 of file wpa_supplicant.c.

Here is the call graph for this function:

**6.104.2.29 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 802 of file wpa_supplicant.c.

6.104.3 Variable Documentation**6.104.3.1 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 50 of file wpa_supplicant.c.

6.104.3.2 const char* wpa_supplicant_version**Initial value:**

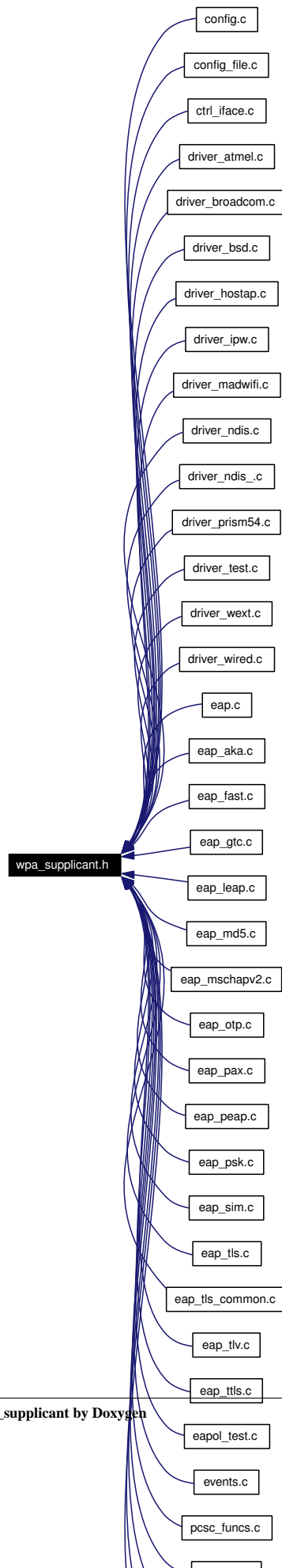
```
"wpa_supplicant v" VERSION_STR "\n"  
"Copyright (c) 2003-2006, Jouni Malinen <jkmaline@cc.hut.fi> and contributors"
```

Definition at line 46 of file wpa_supplicant.c.

6.105 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](#) }

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_msg](#) (struct [wpa_supplicant](#) *wpa_s, int level, char *fmt,...) `__attribute__((format(printf, 3, 4)))`
Conditional printf for default target and ctrl_iface monitors.
- void const char * [wpa_ssid_txt](#) (u8 *ssid, size_t ssid_len)
Convert SSID to a printable string.
- 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.

6.105.1 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](#).

6.105.2 Typedef Documentation

6.105.2.1 typedef enum [wpa_event_type](#) [wpa_event_type](#)

enum [wpa_event_type](#) - Event type for [wpa_supplicant_event\(\)](#) calls

6.105.3 Enumeration Type Documentation

6.105.3.1 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 event 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 for event processing is provided with union `wpa_event_data::michael_mic_failure`. This information is used to request new encryption 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 sent 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.

Definition at line 27 of file `wpa_supplicant.h`.

6.105.4 Function Documentation

6.105.4.1 void wpa_msg (struct wpa_supplicant * wpa_s, int level, char * fmt, ...)

Conditional printf for default target and `ctrl_iface` monitors.

Parameters:

wpa_s pointer to `wpa_supplicant` 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`.

6.105.4.2 void 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 521 of file `wpa_supplicant.c`.

6.105.4.3 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\(\)](#)

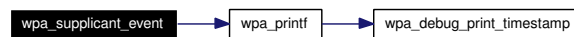
event event type (defined above)

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 732 of file events.c.

Here is the call graph for this function:



6.105.4.4 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

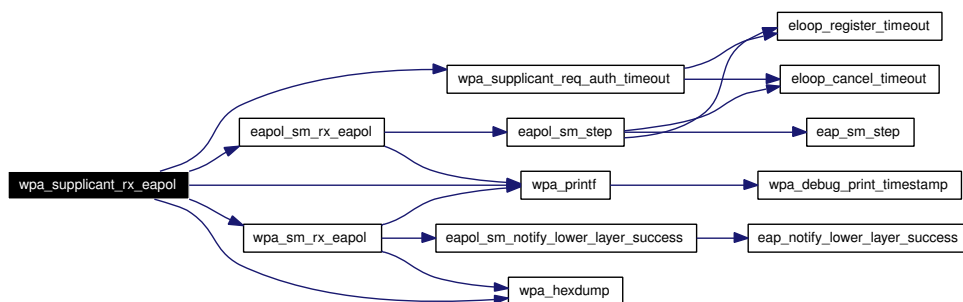
buf EAPOL data starting from the EAPOL header (i.e., no Ethernet header)

len Length of the EAPOL data

This function is called for each received EAPOL frame.

Definition at line 1780 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106 wpa_supplicant_i.h File Reference

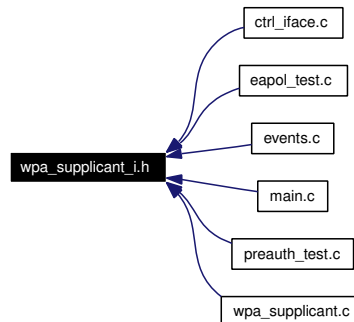
[wpa_supplicant](#) - Internal definitions

```
#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.

6.106.1 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](#).

6.106.2 Function Documentation

6.106.2.1 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:

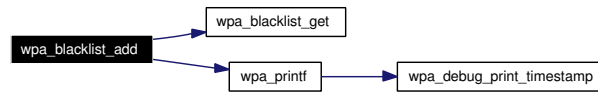
0 on success, -1 on failure

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 433 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.2 void wpa_blacklist_clear (struct wpa_supplicant * wpa_s)

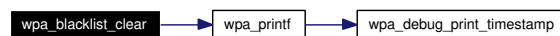
Clear the blacklist of all entries.

Parameters:

wpa_s Pointer to [wpa_supplicant](#) data

Definition at line 490 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.3 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 400 of file wpa_supplicant.c.

6.106.2.4 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 767 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.5 `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

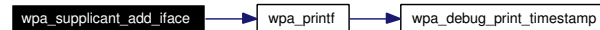
Returns:

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 2211 of file `wpa_supplicant.c`.

Here is the call graph for this function:



6.106.2.6 `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

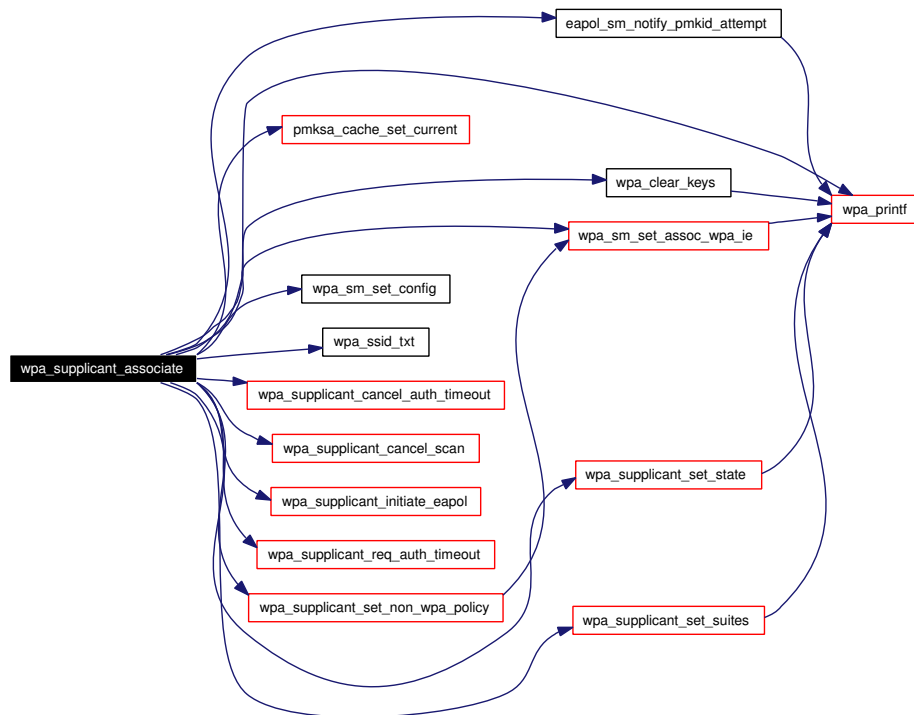
bss Scan results for the selected BSS, or NULL if not available

ssid Configuration data for the selected network

This function is used to request `wpa_supplicant` to associate with a BSS.

Definition at line 1276 of file `wpa_supplicant.c`.

Here is the call graph for this function:



6.106.2.7 void wpa_supplicant_cancel_auth_timeout (struct wpa_supplicant * wpa_s)

Cancel authentication timeout.

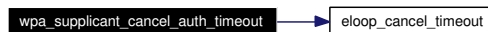
Parameters:

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 618 of file [wpa_supplicant.c](#).

Here is the call graph for this function:



6.106.2.8 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 565 of file [wpa_supplicant.c](#).

Here is the call graph for this function:



6.106.2.9 void wpa_supplicant_deauthenticate (struct wpa_supplicant * wpa_s, int reason_code)

Deauthenticate the current connection.

Parameters:

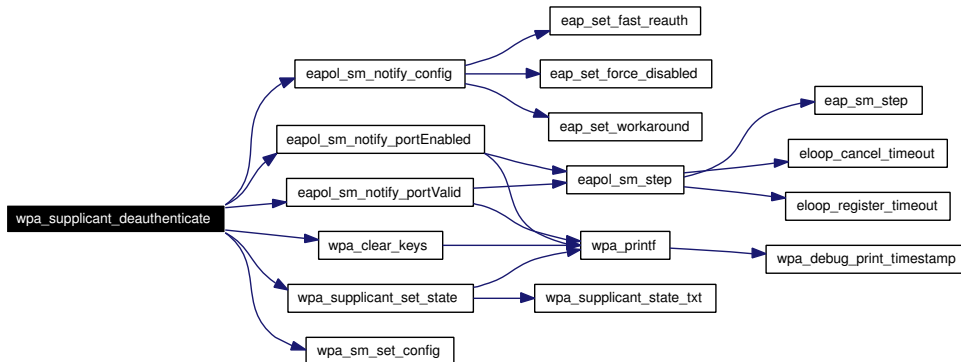
wpa_s Pointer to [wpa_supplicant](#) data

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 1503 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.10 void wpa_supplicant_deinit (struct wpa_global * global)

Deinitialize wpa_supplicant.

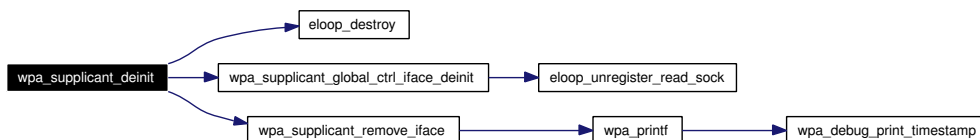
Parameters:

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 2395 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.11 void wpa_supplicant_disassociate (struct wpa_supplicant * wpa_s, int reason_code)

Disassociate the current connection.

Parameters:

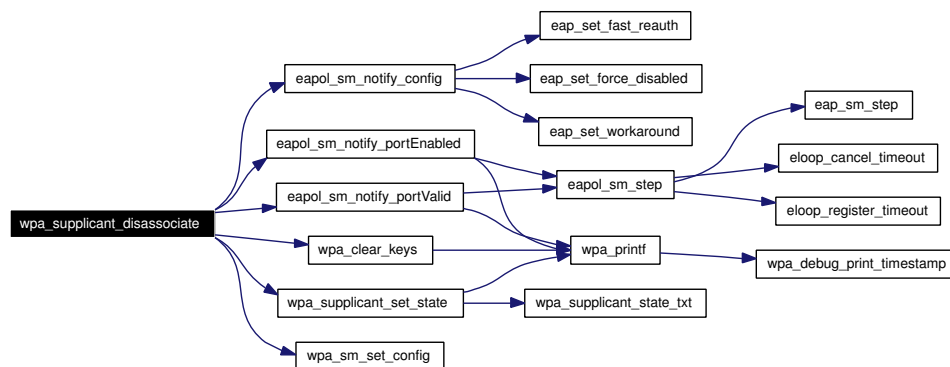
wpa_s Pointer to [wpa_supplicant](#) data

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 1476 of file wpa_supplicant.c.

Here is the call graph for this function:

**6.106.2.12 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

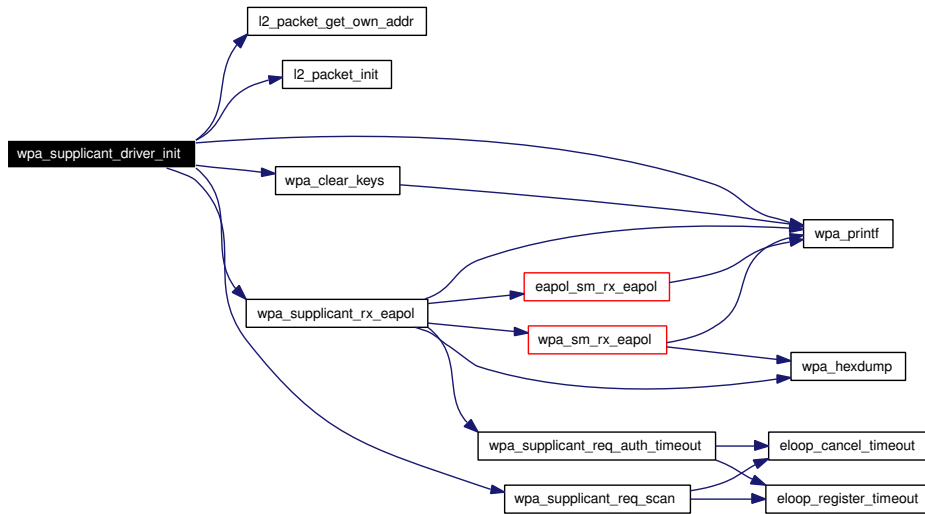
Returns:

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 1837 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.13 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 2288 of file `wpa_supplicant.c`.

6.106.2.14 int `wpa_supplicant_get_scan_results` (struct `wpa_supplicant *` `wpa_s`)

Get scan results.

Parameters:

`wpa_s` Pointer to `wpa_supplicant` data

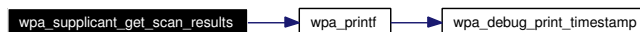
Returns:

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 1530 of file `wpa_supplicant.c`.

Here is the call graph for this function:



6.106.2.15 struct [wpa_ssid](#)* wpa_supplicant_get_ssid (struct [wpa_supplicant](#) * wpa_s)

Get a pointer to the current network structure.

Parameters:

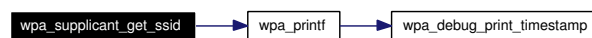
wpa_s Pointer to [wpa_supplicant](#) data

Returns:

A pointer to the current network structure or NULL on failure

Definition at line 1626 of file wpa_supplicant.c.

Here is the call graph for this function:

**6.106.2.16** struct [wpa_global](#)* wpa_supplicant_init (struct [wpa_params](#) * params)

Initialize wpa_supplicant.

Parameters:

params Parameters for wpa_supplicant

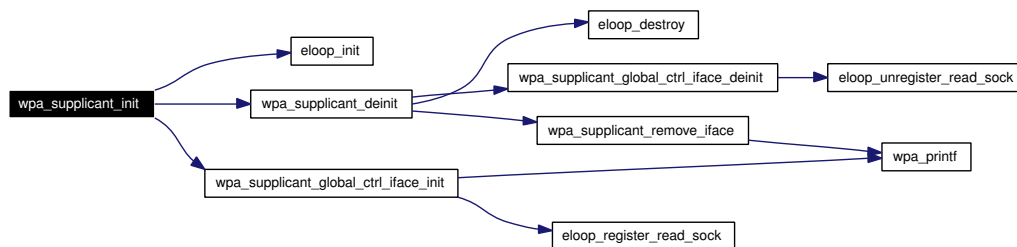
Returns:

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 2311 of file wpa_supplicant.c.

Here is the call graph for this function:

**6.106.2.17** void wpa_supplicant_initiate_eapol (struct [wpa_supplicant](#) * wpa_s)

Configure EAPOL state machine.

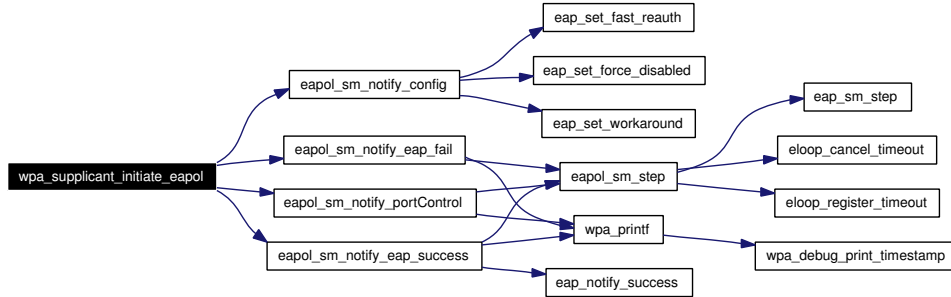
Parameters:

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 634 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.18 int wpa_supplicant_reload_configuration (struct wpa_supplicant * wpa_s)

Reload configuration data.

Parameters:

wpa_s Pointer to [wpa_supplicant](#) data

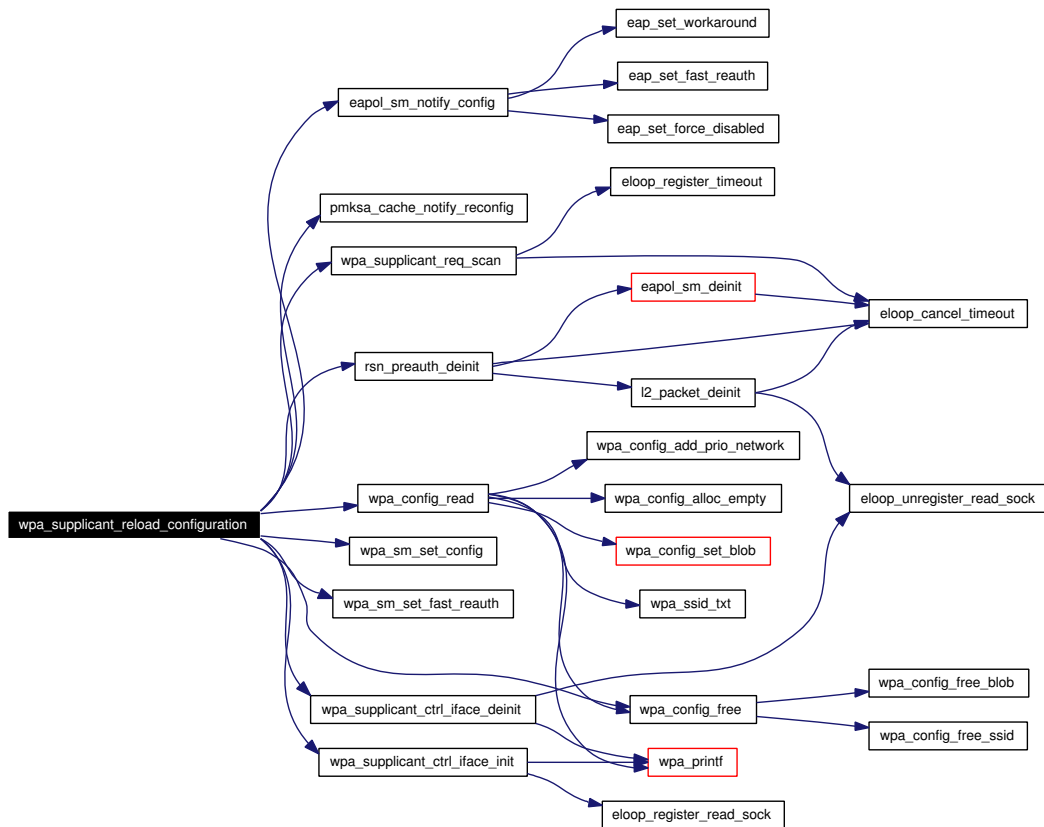
Returns:

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 893 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.19 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

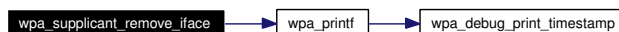
Returns:

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 2255 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.20 void wpa_supplicant_req_auth_timeout (struct wpa_supplicant * wpa_s, int sec, int usec)

Schedule a timeout for authentication.

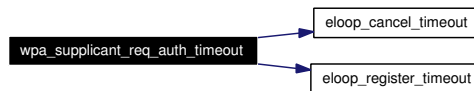
Parameters:

- 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 595 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.21 void wpa_supplicant_req_scan (struct wpa_supplicant * wpa_s, int sec, int usec)

Schedule a scan for neighboring access points.

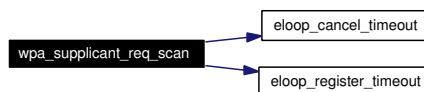
Parameters:

- 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 548 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.22 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\(\)](#)

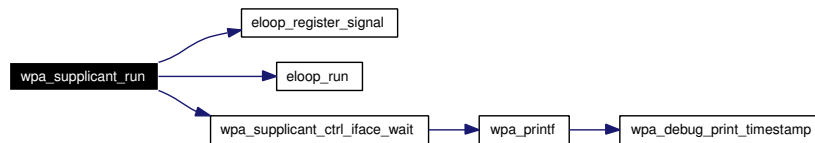
Returns:

- 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 is still in use.

Definition at line 2362 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.23 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

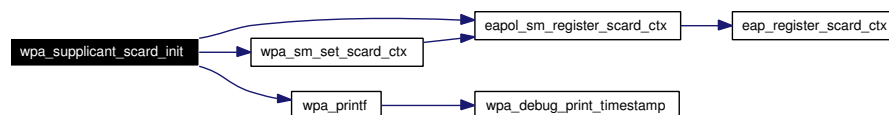
Returns:

- 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 173 of file events.c.

Here is the call graph for this function:



6.106.2.24 void wpa_supplicant_set_non_wpa_policy (struct wpa_supplicant * wpa_s, struct wpa_ssid * ssid)

Set WPA parameters to non-WPA mode.

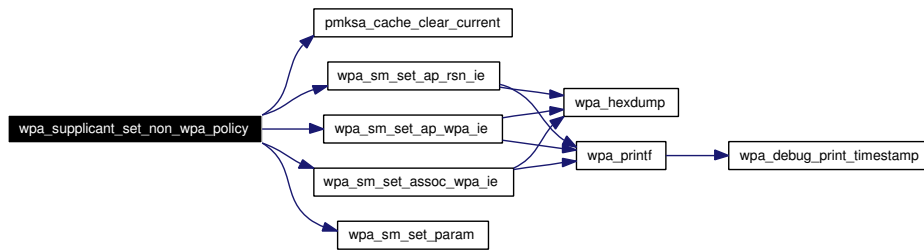
Parameters:

- 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 684 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.25 void wpa_supplicant_set_state (struct wpa_supplicant * wpa_s, wpa_states state)

Set current connection state.

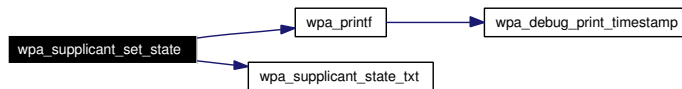
Parameters:

- 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 836 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.26 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.

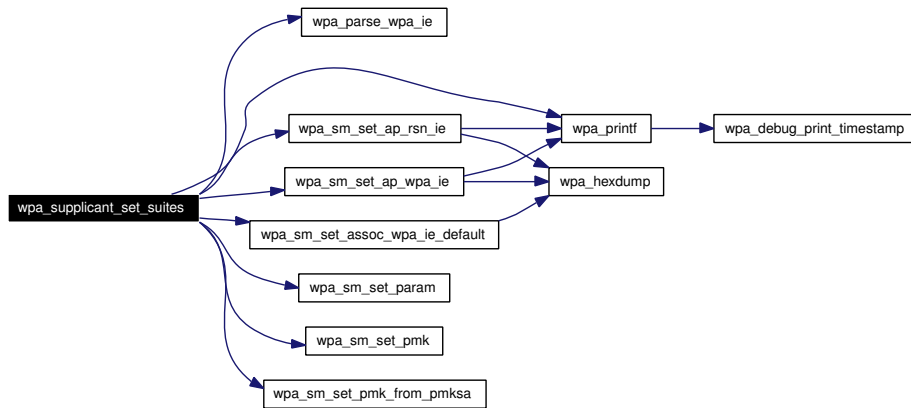
Returns:

- 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 1146 of file wpa_supplicant.c.

Here is the call graph for this function:



6.106.2.27 `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 802 of file wpa_supplicant.c.

Chapter 7

wpa_supplicant Page Documentation

7.1 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](#).

7.1.1 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

7.1.2 Generic helper functions

wpa_supplicant uses generic helper functions some of which are shared with hostapd. The following C files are currently used:

[elooop.c](#) and [elooop.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

7.1.3 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 libcrypt (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

7.1.4 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)

7.1.5 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

7.1.6 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.

7.1.7 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_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

7.1.8 EAPOL supplicant

[eapol_sm.c](#) and [eapol_sm.h](#) EAPOL supplicant state machine and IEEE 802.1X processing

7.1.9 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

7.1.10 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

7.2 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.

7.2.1 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\(\)](#).

7.2.2 Control interface commands

Following commands can be used with [wpa_ctrl_request\(\)](#):

7.2.2.1 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.

7.2.2.2 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

```

7.2.2.3 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

```

7.2.2.4 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
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
```

7.2.2.5 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
```

7.2.2.6 SET <variable> <value>

Set variables:

- EAPOL::heldPeriod
- EAPOL::authPeriod
- EAPOL::startPeriod
- EAPOL::maxStart
- dot11RSNACfgPMKLifetime
- dot11RSNACfgPMKReauthThreshold
- dot11RSNACfgSATimeout

Example command:

```
SET EAPOL::heldPeriod 45
```

7.2.2.7 LOGON

IEEE 802.1X EAPOL state machine logon.

7.2.2.8 LOGOFF

IEEE 802.1X EAPOL state machine logoff.

7.2.2.9 REASSOCIATE

Force reassociation.

7.2.2.10 PREAUTH <BSSID>

Start pre-authentication with the given BSSID.

7.2.2.11 ATTACH

Attach the connection as a monitor for unsolicited events. This can be done with [wpa_ctrl_attach\(\)](#).

7.2.2.12 DETACH

Detach the connection as a monitor for unsolicited events. This can be done with [wpa_ctrl_detach\(\)](#).

7.2.2.13 LEVEL <debug level>

Change debug level.

7.2.2.14 RECONFIGURE

Force wpa_supplicant to re-read its configuration data.

7.2.2.15 TERMINATE

Terminate wpa_supplicant process.

7.2.2.16 BSSID <network id> <BSSID>

Set preferred BSSID for a network. Network id can be received from the LIST_NETWORKS command output.

7.2.2.17 LIST_NETWORKS

List configured networks.

```
network id / ssid / bssid / flags  
0 example network any [CURRENT]
```

(note: fields are separated with tabs)

7.2.2.18 DISCONNECT

Disconnect and wait for REASSOCIATE command before connecting.

7.2.2.19 SCAN

Request a new BSS scan.

7.2.2.20 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)

7.2.2.21 SELECT_NETWORK <network id>

Select a network (disable others). Network id can be received from the LIST_NETWORKS command output.

7.2.2.22 ENABLE_NETWORK <network id>

Enable a network. Network id can be received from the LIST_NETWORKS command output.

7.2.2.23 DISABLE_NETWORK <network id>

Disable a network. Network id can be received from the LIST_NETWORKS command output.

7.2.2.24 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.

7.2.2.25 REMOVE_NETWORK <network id>

Remove a network. Network id can be received from the LIST_NETWORKS command output.

7.2.2.26 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)
- ...

7.2.2.27 GET_NETWORK <network id> <variable>

Get network variables. Network id can be received from the LIST_NETWORKS command output.

7.2.2.28 SAVE_CONFIG

Save the current configuration.

7.2.3 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
```

7.2.3.1 GET_CAPABILITY <option>

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.

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 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
```

7.3 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.

7.3.1 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.

7.3.1.1 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.

7.3.1.2 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)

7.3.1.3 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.

7.3.1.4 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.

7.3.1.5 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 target 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)

7.4 EAP peer implementation

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.

7.4.1 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 build (Makefile) and EAP method table in the beginning of `eap.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.

7.5 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.

7.5.1 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.

7.5.2 l2_packet (link layer access)

wpa_supplicant needs to have access to sending and receiving layer 2 (link layer) packets with two Ether-types: 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.

7.5.3 Event loop

wpa_supplicant uses a single process/thread model and an event loop to provide callbacks on events (registered timeout, received packet, signal). [elooop.h](#) defines the event loop interface. [elooop.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.

7.5.4 Control interface

wpa_supplicant uses a [control interface](#) to allow external processes 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`.

7.5.5 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.

7.6 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.

7.6.1 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 [-nW] -c<conf> [-a<AS IP>] [-p<AS port>] [-s<AS secret>] [-r<count>]
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
  -n = no MPPE keys expected
```

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.

7.6.2 preauth_test - WPA2 pre-authentication and EAP peer testing

preauth_test is similar to eapol_test in the sense that it 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.

7.6.3 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
```

7.6.4 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 can 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

- aborted_cached
 - eapol_ctx, 27
- accept_802_1x_keys
 - eapol_config, 24
- add_pmkid
 - wpa_driver_ops, 48
- aes.c, 91
 - aes_decrypt, 93
 - aes_decrypt_deinit, 94
 - aes_decrypt_init, 94
 - aes_encrypt, 94
 - aes_encrypt_deinit, 94
 - aes_encrypt_init, 95
 - GETU32, 93
 - PUTU32, 93
 - rijndaelKeySetupDec, 95
 - rijndaelKeySetupEnc, 95
 - ROUND, 93
- aes_128_cbc_decrypt
 - aes_wrap.c, 97
 - aes_wrap.h, 104
- aes_128_cbc_encrypt
 - aes_wrap.c, 98
 - aes_wrap.h, 104
- aes_128_ctr_encrypt
 - aes_wrap.c, 98
 - aes_wrap.h, 105
- aes_128_eax_decrypt
 - aes_wrap.c, 99
 - aes_wrap.h, 105
- aes_128_eax_encrypt
 - aes_wrap.c, 99
 - aes_wrap.h, 106
- aes_128_encrypt_block
 - aes_wrap.c, 100
 - aes_wrap.h, 107
- aes_decrypt
 - aes.c, 93
 - crypto.h, 151
- aes_decrypt_deinit
 - aes.c, 94
 - crypto.h, 152
- aes_decrypt_init
 - aes.c, 94
 - crypto.h, 152
- aes_encrypt
 - aes.c, 94
 - crypto.h, 152
- aes_encrypt_deinit
 - aes.c, 94
 - crypto.h, 152
- aes_encrypt_init
 - aes.c, 95
 - crypto.h, 152
- aes_unwrap
 - aes_wrap.c, 100
 - aes_wrap.h, 107
- aes_wrap
 - aes_wrap.c, 101
 - aes_wrap.h, 108
- aes_wrap.c, 96
 - aes_128_cbc_decrypt, 97
 - aes_128_cbc_encrypt, 98
 - aes_128_ctr_encrypt, 98
 - aes_128_eax_decrypt, 99
 - aes_128_eax_encrypt, 99
 - aes_128_encrypt_block, 100
 - aes_unwrap, 100
 - aes_wrap, 101
 - omac1_aes_128, 101
- aes_wrap.h, 103
 - aes_128_cbc_decrypt, 104
 - aes_128_cbc_encrypt, 104
 - aes_128_ctr_encrypt, 105
 - aes_128_eax_decrypt, 105
 - aes_128_eax_encrypt, 106
 - aes_128_encrypt_block, 107
 - aes_unwrap, 107
 - aes_wrap, 108
 - omac1_aes_128, 108
- altsubject_match
 - wpa_ssid, 77
- altsubject_match2
 - wpa_ssid, 78
- anonymous_identity
 - wpa_ssid, 78
- ap_scan
 - wpa_config, 37
- assoc_info
 - wpa_event_data, 58

- associate
 - wpa_driver_ops, 49
- auth_alg
 - wpa_driver_associate_params, 44
 - wpa_ssid, 78
- AVP_PAD
 - eap_ttls.h, 303
- base64.c, 110
 - base64_decode, 110
 - base64_encode, 111
- base64.h, 112
 - base64_decode, 112
 - base64_encode, 112
- base64_decode
 - base64.c, 110
 - base64.h, 112
- base64_encode
 - base64.c, 111
 - base64.h, 112
- beacon_ies
 - wpa_event_data::assoc_info, 59
- blk
 - sha1.c, 425
- blk0
 - sha1.c, 425
- bssid
 - wpa_driver_associate_params, 44
 - wpa_event_data::pmkid_candidate, 63
 - wpa_ssid, 78
- ca_cert
 - wpa_ssid, 78
- ca_cert2
 - wpa_ssid, 79
- ca_path
 - wpa_ssid, 79
- ca_path2
 - wpa_ssid, 79
- cb
 - eapol_ctx, 27
- challenge_response
 - ms_funcs.c, 373
 - ms_funcs.h, 379
- client_cert
 - wpa_ssid, 79
- client_cert2
 - wpa_ssid, 79
- common.c, 114
 - hexstr2bin, 115
 - hwaddr_aton, 116
 - inc_byte_array, 116
 - wpa_debug_print_timestamp, 116
 - wpa_hexdump, 116
 - wpa_hexdump_ascii, 117
 - wpa_hexdump_ascii_key, 117
 - wpa_hexdump_key, 117
 - wpa_printf, 118
- common.h, 119
 - hexstr2bin, 123
 - hwaddr_aton, 123
 - inc_byte_array, 123
 - wpa_debug_print_timestamp, 124
 - WPA_GET_BE32, 122
 - wpa_hexdump, 124
 - wpa_hexdump_ascii, 124
 - wpa_hexdump_ascii_key, 124
 - wpa_hexdump_key, 125
 - wpa_printf, 125
 - WPA_PUT_BE16, 122
 - WPA_PUT_LE16, 123
- config.c, 126
 - FUNC, 128
 - INT, 128
 - INT_RANGE, 128
 - STR, 128
 - STR_RANGE, 128
 - wpa_config_add_network, 129
 - wpa_config_add_prio_network, 129
 - wpa_config_alloc_empty, 129
 - wpa_config_allowed_eap_method, 129
 - wpa_config_free, 130
 - wpa_config_free_blob, 130
 - wpa_config_free_ssid, 130
 - wpa_config_get, 130
 - wpa_config_get_blob, 131
 - wpa_config_get_network, 131
 - wpa_config_remove_blob, 131
 - wpa_config_remove_network, 132
 - wpa_config_set, 132
 - wpa_config_set_blob, 132
 - wpa_config_set_network_defaults, 133
 - wpa_config_update_psk, 133
- config.h, 134
 - wpa_config_add_network, 136
 - wpa_config_add_prio_network, 136
 - wpa_config_alloc_empty, 136
 - wpa_config_free, 136
 - wpa_config_free_blob, 137
 - wpa_config_free_ssid, 137
 - wpa_config_get, 137
 - wpa_config_get_blob, 137
 - wpa_config_get_network, 138
 - wpa_config_read, 138
 - wpa_config_remove_blob, 139
 - wpa_config_remove_network, 139
 - wpa_config_set, 139
 - wpa_config_set_blob, 140

- wpa_config_set_network_defaults, 140
 - wpa_config_update_psk, 140
 - wpa_config_write, 141
- config_file.c, 142
 - wpa_config_read, 143
 - wpa_config_write, 143
- config_ssid.h, 145
 - DEFAULT_EAPOL_FLAGS, 146
 - DEFAULT_GROUP, 146
 - wpa_config_allowed_eap_method, 147
- confname
 - wpa_interface, 65
- crypto.c, 148
 - des_encrypt, 149
 - md4_vector, 149
- crypto.h, 150
 - aes_decrypt, 151
 - aes_decrypt_deinit, 152
 - aes_decrypt_init, 152
 - aes_encrypt, 152
 - aes_encrypt_deinit, 152
 - aes_encrypt_init, 152
 - des_encrypt, 153
 - md4_vector, 153
 - md5_vector, 153
 - sha1_transform, 153
 - sha1_vector, 154
- crypto_gnutls.c, 155
 - des_encrypt, 155
 - md4_vector, 156
- ctrl_iface.c, 157
 - wpa_supplicant_ctrl_iface_deinit, 159
 - wpa_supplicant_ctrl_iface_init, 159
 - wpa_supplicant_ctrl_iface_send, 160
 - wpa_supplicant_ctrl_iface_wait, 160
 - wpa_supplicant_global_ctrl_iface_deinit, 161
 - wpa_supplicant_global_ctrl_iface_init, 161
- ctrl_iface.h, 162
- ctrl_interface
 - wpa_config, 37
 - wpa_interface, 65
- deauthenticate
 - wpa_driver_ops, 49
- DEFAULT_EAPOL_FLAGS
 - config_ssid.h, 146
- DEFAULT_GROUP
 - config_ssid.h, 146
- defs.h, 163
 - WPA_4WAY_HANDSHAKE, 164
 - WPA_ASSOCIATED, 164
 - WPA_ASSOCIATING, 164
 - WPA_COMPLETED, 164
 - WPA_DISCONNECTED, 164
 - WPA_GROUP_HANDSHAKE, 164
 - WPA_INACTIVE, 164
 - WPA_SCANNING, 164
 - wpa_states, 164
- deinit
 - eap_method, 14
 - wpa_driver_ops, 49
- deinit_for_reauth
 - eap_method, 14
- des_encrypt
 - crypto.c, 149
 - crypto.h, 153
 - crypto_gnutls.c, 155
- desc
 - wpa_driver_ops, 49
- dh_file
 - wpa_ssid, 80
- dh_file2
 - wpa_ssid, 80
- disabled
 - wpa_ssid, 80
- disassociate
 - wpa_driver_ops, 49
- dot11RSNAConfigPMKLifetime
 - wpa_config, 38
- dot11RSNAConfigPMKReauthThreshold
 - wpa_config, 38
- dot11RSNAConfigSATimeout
 - wpa_config, 38
- driver.h, 165
- driver_atmel.c, 167
 - wpa_driver_atmel_ops, 168
- driver_broadcom.c, 169
 - wpa_driver_broadcom_ops, 171
- driver_bsd.c, 172
 - LE_READ_4, 174
 - wpa_driver_bsd_ops, 174
- driver_hostap.c, 175
 - wpa_driver_hostap_ops, 176
- driver_hostap.h, 177
- driver_ipw.c, 179
 - wpa_driver_ipw_ops, 180
- driver_madwifi.c, 182
 - wpa_driver_madwifi_ops, 184
- driver_ndis.c, 185
 - wpa_driver_ndis_ops, 188
- driver_ndis.h, 189
- driver_ndis_c, 190
- driver_ndiswrapper.c, 192
 - wpa_driver_ndiswrapper_ops, 193
- driver_param
 - wpa_config, 38
 - wpa_interface, 65

- driver_prism54.c, 195
 - wpa_driver_prism54_ops, 196
- driver_test.c, 197
 - wpa_driver_test_ops, 198
- driver_wext.c, 199
 - wpa_driver_wext_deinit, 202
 - wpa_driver_wext_get_bssid, 202
 - wpa_driver_wext_get_ifflags, 202
 - wpa_driver_wext_get_scan_results, 202
 - wpa_driver_wext_get_ssid, 203
 - wpa_driver_wext_init, 203
 - wpa_driver_wext_ops, 207
 - wpa_driver_wext_scan, 204
 - wpa_driver_wext_scan_timeout, 204
 - wpa_driver_wext_set_bssid, 205
 - wpa_driver_wext_set_freq, 205
 - wpa_driver_wext_set_ifflags, 205
 - wpa_driver_wext_set_key, 205
 - wpa_driver_wext_set_mode, 206
 - wpa_driver_wext_set_ssid, 206
- driver_wext.h, 208
 - wpa_driver_wext_deinit, 209
 - wpa_driver_wext_get_bssid, 210
 - wpa_driver_wext_get_ifflags, 210
 - wpa_driver_wext_get_scan_results, 210
 - wpa_driver_wext_get_ssid, 211
 - wpa_driver_wext_init, 211
 - wpa_driver_wext_scan, 211
 - wpa_driver_wext_scan_timeout, 212
 - wpa_driver_wext_set_bssid, 212
 - wpa_driver_wext_set_freq, 213
 - wpa_driver_wext_set_ifflags, 213
 - wpa_driver_wext_set_key, 213
 - wpa_driver_wext_set_mode, 214
 - wpa_driver_wext_set_ssid, 214
- driver_wired.c, 215
 - wpa_driver_wired_ops, 216
- drivers.c, 217
- eap.c, 218
 - eap_get_config, 222
 - eap_get_config_blob, 222
 - eap_get_eapKeyData, 222
 - eap_get_eapRespData, 223
 - eap_get_name, 223
 - eap_get_names, 223
 - eap_get_phase2_type, 224
 - eap_get_phase2_types, 224
 - eap_get_type, 224
 - eap_hdr_validate, 225
 - eap_key_available, 225
 - eap_notify_lower_layer_success, 225
 - eap_notify_success, 226
 - eap_register_scard_ctx, 226
 - eap_set_config_blob, 226
 - eap_set_fast_reauth, 226
 - eap_set_force_disabled, 227
 - eap_set_workaround, 227
 - eap_sm_abort, 227
 - eap_sm_buildIdentity, 227
 - eap_sm_deinit, 228
 - eap_sm_get_eap_methods, 228
 - eap_sm_get_status, 228
 - eap_sm_init, 229
 - eap_sm_notify_ctrl_attached, 229
 - eap_sm_request_identity, 230
 - eap_sm_request_new_password, 230
 - eap_sm_request_otp, 230
 - eap_sm_request_passphrase, 231
 - eap_sm_request_password, 231
 - eap_sm_request_pin, 231
 - eap_sm_step, 232
 - SM_ENTRY, 222
 - SM_STATE, 222
- eap.h, 233
 - eap_get_config, 237
 - eap_get_eapKeyData, 237
 - eap_get_eapRespData, 237
 - eap_get_name, 238
 - eap_get_names, 238
 - eap_get_phase2_type, 238
 - eap_get_phase2_types, 239
 - eap_get_type, 239
 - eap_key_available, 239
 - eap_notify_lower_layer_success, 239
 - eap_notify_success, 240
 - eap_register_scard_ctx, 240
 - eap_set_fast_reauth, 240
 - eap_set_force_disabled, 240
 - eap_set_workaround, 241
 - eap_sm_abort, 241
 - eap_sm_buildIdentity, 241
 - eap_sm_deinit, 242
 - eap_sm_get_eap_methods, 242
 - eap_sm_get_status, 242
 - eap_sm_init, 243
 - eap_sm_notify_ctrl_attached, 243
 - eap_sm_request_identity, 244
 - eap_sm_request_new_password, 244
 - eap_sm_request_otp, 244
 - eap_sm_request_passphrase, 245
 - eap_sm_request_password, 245
 - eap_sm_request_pin, 245
 - eap_sm_step, 246
 - EAPOL_altAccept, 236
 - EAPOL_altReject, 236
 - eapol_bool_var, 236
 - EAPOL_eapFail, 236

- EAPOL_eapNoResp, 236
- EAPOL_eapReq, 236
- EAPOL_eapResp, 236
- EAPOL_eapRestart, 236
- EAPOL_eapSuccess, 236
- EAPOL_idleWhile, 237
- eapol_int_var, 236
- EAPOL_portEnabled, 236
- eap_aka.c, 247
 - eap_method_aka, 248
- eap_config, 11
 - opensc_engine_path, 11
 - pkcs11_engine_path, 11
 - pkcs11_module_path, 12
- eap_defs.h, 249
- eap_fast.c, 251
 - eap_method_fast, 252
- eap_get_config
 - eap.c, 222
 - eap.h, 237
- eap_get_config_blob
 - eap.c, 222
 - eap_i.h, 256
- eap_get_eapKeyData
 - eap.c, 222
 - eap.h, 237
- eap_get_eapRespData
 - eap.c, 223
 - eap.h, 237
- eap_get_name
 - eap.c, 223
 - eap.h, 238
- eap_get_names
 - eap.c, 223
 - eap.h, 238
- eap_get_phase2_type
 - eap.c, 224
 - eap.h, 238
- eap_get_phase2_types
 - eap.c, 224
 - eap.h, 239
- eap_get_type
 - eap.c, 224
 - eap.h, 239
- eap_gtc.c, 253
 - eap_method_gtc, 254
- eap_hdr_validate
 - eap.c, 225
 - eap_i.h, 256
- eap_i.h, 255
 - eap_get_config_blob, 256
 - eap_hdr_validate, 256
 - eap_set_config_blob, 257
- eap_key_available
 - eap.c, 225
 - eap.h, 239
- eap_leap.c, 258
 - eap_method_leap, 259
- eap_md5.c, 260
 - eap_method_md5, 261
- eap_method, 13
 - deinit, 14
 - deinit_for_reauth, 14
 - get_identity, 14
 - get_status, 14
 - getKey, 15
 - has_reauth_data, 15
 - init, 15
 - init_for_reauth, 16
 - isKeyAvailable, 16
 - process, 16
- eap_method_aka
 - eap_aka.c, 248
- eap_method_fast
 - eap_fast.c, 252
- eap_method_gtc
 - eap_gtc.c, 254
- eap_method_leap
 - eap_leap.c, 259
- eap_method_md5
 - eap_md5.c, 261
- eap_method_mschapv2
 - eap_mschapv2.c, 263
- eap_method_otp
 - eap_otp.c, 265
- eap_method_pax
 - eap_pax.c, 267
- eap_method_peap
 - eap_peap.c, 276
- eap_method_psk
 - eap_psk.c, 278
- eap_method_ret, 18
- eap_method_sim
 - eap_sim.c, 282
- eap_method_tls
 - eap_tls.c, 288
- eap_method_ttls
 - eap_ttls.c, 301
- eap_methods
 - wpa_ssid, 80
- eap_mschapv2.c, 262
 - eap_method_mschapv2, 263
- eap_notify_lower_layer_success
 - eap.c, 225
 - eap.h, 239
- eap_notify_success
 - eap.c, 226
 - eap.h, 240

- eap_otp.c, 264
 - eap_method_otp, 265
- eap_pax.c, 266
 - eap_method_pax, 267
- eap_pax_common.c, 268
 - eap_pax_initial_key_derivation, 269
 - eap_pax_kdf, 269
 - eap_pax_mac, 270
- eap_pax_common.h, 271
 - eap_pax_initial_key_derivation, 272
 - eap_pax_kdf, 272
 - eap_pax_mac, 273
- eap_pax_initial_key_derivation
 - eap_pax_common.c, 269
 - eap_pax_common.h, 272
- eap_pax_kdf
 - eap_pax_common.c, 269
 - eap_pax_common.h, 272
- eap_pax_mac
 - eap_pax_common.c, 270
 - eap_pax_common.h, 273
- eap_peap.c, 275
 - eap_method_peap, 276
- eap_psk.c, 277
 - eap_method_psk, 278
- eap_psk_common.c, 279
- eap_psk_common.h, 280
- eap_register_scard_ctx
 - eap.c, 226
 - eap.h, 240
- eap_set_config_blob
 - eap.c, 226
 - eap_i.h, 257
- eap_set_fast_reauth
 - eap.c, 226
 - eap.h, 240
- eap_set_force_disabled
 - eap.c, 227
 - eap.h, 240
- eap_set_workaround
 - eap.c, 227
 - eap.h, 241
- eap_sim.c, 281
 - eap_method_sim, 282
- eap_sim_common.c, 283
- eap_sim_common.h, 285
- eap_sm, 19
- eap_sm_abort
 - eap.c, 227
 - eap.h, 241
- eap_sm_buildIdentity
 - eap.c, 227
 - eap.h, 241
- eap_sm_deinit
 - eap.c, 228
 - eap.h, 242
- eap_sm_get_eap_methods
 - eap.c, 228
 - eap.h, 242
- eap_sm_get_status
 - eap.c, 228
 - eap.h, 242
- eap_sm_init
 - eap.c, 229
 - eap.h, 243
- eap_sm_notify_ctrl_attached
 - eap.c, 229
 - eap.h, 243
- eap_sm_request_identity
 - eap.c, 230
 - eap.h, 244
- eap_sm_request_new_password
 - eap.c, 230
 - eap.h, 244
- eap_sm_request_otp
 - eap.c, 230
 - eap.h, 244
- eap_sm_request_passphrase
 - eap.c, 231
 - eap.h, 245
- eap_sm_request_password
 - eap.c, 231
 - eap.h, 245
- eap_sm_request_pin
 - eap.c, 231
 - eap.h, 245
- eap_sm_step
 - eap.c, 232
 - eap.h, 246
- eap_tls.c, 287
 - eap_method_tls, 288
- eap_tls_common.c, 289
 - eap_tls_data_reassemble, 290
- eap_tls_common.h, 292
 - eap_tls_data_reassemble, 293
- eap_tls_data_reassemble
 - eap_tls_common.c, 290
 - eap_tls_common.h, 293
- eap_tlv.c, 294
 - eap_tlv_build_nak, 295
 - eap_tlv_build_result, 295
 - eap_tlv_process, 295
- eap_tlv.h, 297
 - eap_tlv_build_nak, 298
 - eap_tlv_build_result, 298
 - eap_tlv_process, 298
- eap_tlv_build_nak
 - eap_tlv.c, 295

- eap_tlv.h, 298
- eap_tlv_build_result
 - eap_tlv.c, 295
 - eap_tlv.h, 298
- eap_tlv_process
 - eap_tlv.c, 295
 - eap_tlv.h, 298
- eap_ttls.c, 300
 - eap_method_ttls, 301
- eap_ttls.h, 302
 - AVP_PAD, 303
- eap_workaround
 - wpa_ssid, 80
- EAPOL_altAccept
 - eap.h, 236
- EAPOL_altReject
 - eap.h, 236
- eapol_bool_var
 - eap.h, 236
- eapol_callbacks, 21
 - get_bool, 22
 - get_config, 22
 - get_config_blob, 22
 - get_eapReqData, 22
 - get_int, 22
 - set_bool, 23
 - set_config_blob, 23
 - set_int, 23
- eapol_config, 24
 - accept_802_1x_keys, 24
 - required_keys, 24
- eapol_ctx, 26
 - aborted_cached, 27
 - cb, 27
 - eapol_done_cb, 27
 - eapol_send, 28
 - get_config_blob, 28
 - openc_engine_path, 28
 - pkcs11_engine_path, 28
 - pkcs11_module_path, 28
 - preauth, 28
 - scard_ctx, 29
 - set_config_blob, 29
 - set_wep_key, 29
- eapol_done_cb
 - eapol_ctx, 27
- EAPOL_eapFail
 - eap.h, 236
- EAPOL_eapNoResp
 - eap.h, 236
- EAPOL_eapReq
 - eap.h, 236
- EAPOL_eapResp
 - eap.h, 236
- EAPOL_eapRestart
 - eap.h, 236
- EAPOL_eapSuccess
 - eap.h, 236
- EAPOL_idleWhile
 - eap.h, 237
- eapol_int_var
 - eap.h, 236
- EAPOL_portEnabled
 - eap.h, 236
- eapol_send
 - eapol_ctx, 28
- eapol_sm, 30
- eapol_sm.c, 304
 - eapol_sm_configure, 307
 - eapol_sm_deinit, 308
 - eapol_sm_get_key, 308
 - eapol_sm_get_mib, 308
 - eapol_sm_get_status, 309
 - eapol_sm_init, 309
 - eapol_sm_notify_cached, 310
 - eapol_sm_notify_config, 310
 - eapol_sm_notify_ctrl_attached, 311
 - eapol_sm_notify_ctrl_response, 311
 - eapol_sm_notify_eap_fail, 312
 - eapol_sm_notify_eap_success, 312
 - eapol_sm_notify_logoff, 313
 - eapol_sm_notify_lower_layer_success, 313
 - eapol_sm_notify_pmkid_attempt, 313
 - eapol_sm_notify_portControl, 314
 - eapol_sm_notify_portEnabled, 314
 - eapol_sm_notify_portValid, 315
 - eapol_sm_notify_tx_eapol_key, 315
 - eapol_sm_register_scard_ctx, 315
 - eapol_sm_request_reauth, 316
 - eapol_sm_rx_eapol, 316
 - eapol_sm_step, 316
 - SM_ENTRY, 307
 - SM_STATE, 307
- eapol_sm.h, 318
 - eapol_sm_configure, 320
 - eapol_sm_deinit, 320
 - eapol_sm_get_key, 321
 - eapol_sm_get_mib, 321
 - eapol_sm_get_status, 322
 - eapol_sm_init, 322
 - eapol_sm_notify_cached, 323
 - eapol_sm_notify_config, 323
 - eapol_sm_notify_ctrl_attached, 323
 - eapol_sm_notify_ctrl_response, 324
 - eapol_sm_notify_eap_fail, 324
 - eapol_sm_notify_eap_success, 325
 - eapol_sm_notify_logoff, 325
 - eapol_sm_notify_lower_layer_success, 326

- eapol_sm_notify_pmkid_attempt, 326
- eapol_sm_notify_portControl, 326
- eapol_sm_notify_portEnabled, 327
- eapol_sm_notify_portValid, 327
- eapol_sm_notify_tx_eapol_key, 328
- eapol_sm_register_scard_ctx, 328
- eapol_sm_request_reauth, 328
- eapol_sm_rx_eapol, 329
- eapol_sm_step, 329
- eapol_sm_configure
 - eapol_sm.c, 307
 - eapol_sm.h, 320
- eapol_sm_deinit
 - eapol_sm.c, 308
 - eapol_sm.h, 320
- eapol_sm_get_key
 - eapol_sm.c, 308
 - eapol_sm.h, 321
- eapol_sm_get_mib
 - eapol_sm.c, 308
 - eapol_sm.h, 321
- eapol_sm_get_status
 - eapol_sm.c, 309
 - eapol_sm.h, 322
- eapol_sm_init
 - eapol_sm.c, 309
 - eapol_sm.h, 322
- eapol_sm_notify_cached
 - eapol_sm.c, 310
 - eapol_sm.h, 323
- eapol_sm_notify_config
 - eapol_sm.c, 310
 - eapol_sm.h, 323
- eapol_sm_notify_ctrl_attached
 - eapol_sm.c, 311
 - eapol_sm.h, 323
- eapol_sm_notify_ctrl_response
 - eapol_sm.c, 311
 - eapol_sm.h, 324
- eapol_sm_notify_eap_fail
 - eapol_sm.c, 312
 - eapol_sm.h, 324
- eapol_sm_notify_eap_success
 - eapol_sm.c, 312
 - eapol_sm.h, 325
- eapol_sm_notify_logoff
 - eapol_sm.c, 313
 - eapol_sm.h, 325
- eapol_sm_notify_lower_layer_success
 - eapol_sm.c, 313
 - eapol_sm.h, 326
- eapol_sm_notify_pmkid_attempt
 - eapol_sm.c, 313
 - eapol_sm.h, 326
- eapol_sm_notify_portControl
 - eapol_sm.c, 314
 - eapol_sm.h, 326
- eapol_sm_notify_portEnabled
 - eapol_sm.c, 314
 - eapol_sm.h, 327
- eapol_sm_notify_portValid
 - eapol_sm.c, 315
 - eapol_sm.h, 327
- eapol_sm_notify_tx_eapol_key
 - eapol_sm.c, 315
 - eapol_sm.h, 328
- eapol_sm_register_scard_ctx
 - eapol_sm.c, 315
 - eapol_sm.h, 328
- eapol_sm_request_reauth
 - eapol_sm.c, 316
 - eapol_sm.h, 328
- eapol_sm_rx_eapol
 - eapol_sm.c, 316
 - eapol_sm.h, 329
- eapol_sm_step
 - eapol_sm.c, 316
 - eapol_sm.h, 329
- eapol_test.c, 330
- eapol_version
 - wpa_config, 38
- eappsk_len
 - wpa_ssid, 81
- eloop.c, 333
 - eloop_cancel_timeout, 334
 - eloop_destroy, 335
 - eloop_init, 335
 - eloop_register_read_sock, 335
 - eloop_register_signal, 335
 - eloop_register_timeout, 336
 - eloop_run, 336
 - eloop_terminate, 336
 - eloop_terminated, 336
 - eloop_unregister_read_sock, 337
- eloop.h, 338
 - eloop_cancel_timeout, 340
 - eloop_destroy, 340
 - eloop_init, 340
 - eloop_register_read_sock, 340
 - eloop_register_signal, 341
 - eloop_register_timeout, 341
 - eloop_run, 341
 - eloop_terminate, 342
 - eloop_terminated, 342
 - eloop_unregister_read_sock, 342
- eloop_cancel_timeout
 - eloop.c, 334
 - eloop.h, 340

- eloop_destroy
 - eloop.c, 335
 - eloop.h, 340
- eloop_init
 - eloop.c, 335
 - eloop.h, 340
- eloop_register_read_sock
 - eloop.c, 335
 - eloop.h, 340
- eloop_register_signal
 - eloop.c, 335
 - eloop.h, 341
- eloop_register_timeout
 - eloop.c, 336
 - eloop.h, 341
- eloop_run
 - eloop.c, 336
 - eloop.h, 341
- eloop_terminate
 - eloop.c, 336
 - eloop.h, 342
- eloop_terminated
 - eloop.c, 336
 - eloop.h, 342
- eloop_unregister_read_sock
 - eloop.c, 337
 - eloop.h, 342
- engine
 - wpa_ssid, 81
- engine_id
 - wpa_ssid, 81
- EVENT_ASSOC
 - wpa_supplicant.h, 549
- EVENT_ASSOCINFO
 - wpa_supplicant.h, 549
- EVENT_DISASSOC
 - wpa_supplicant.h, 549
- EVENT_INTERFACE_STATUS
 - wpa_supplicant.h, 549
- EVENT_MICHAEL_MIC_FAILURE
 - wpa_supplicant.h, 549
- EVENT_PMKID_CANDIDATE
 - wpa_supplicant.h, 549
- EVENT_SCAN_RESULTS
 - wpa_supplicant.h, 549
- events.c, 343
 - wpa_supplicant_event, 345
 - wpa_supplicant_scard_init, 345
- fast_reauth
 - wpa_config, 38
- flush_pmkid
 - wpa_driver_ops, 50
- freq
 - wpa_driver_associate_params, 45
- FUNC
 - config.c, 128
- generate_authenticator_response
 - ms_funcs.c, 374
 - ms_funcs.h, 379
- generate_nt_response
 - ms_funcs.c, 374
 - ms_funcs.h, 380
- get_asymmetric_start_key
 - ms_funcs.c, 375
 - ms_funcs.h, 380
- get_bool
 - eapol_callbacks, 22
- get_bssid
 - wpa_driver_ops, 50
- get_capa
 - wpa_driver_ops, 50
- get_config
 - eapol_callbacks, 22
- get_config_blob
 - eapol_callbacks, 22
 - eapol_ctx, 28
- get_eapReqData
 - eapol_callbacks, 22
- get_identity
 - eap_method, 14
- get_ifname
 - wpa_driver_ops, 50
- get_int
 - eapol_callbacks, 22
- get_mac_addr
 - wpa_driver_ops, 51
- get_master_key
 - ms_funcs.c, 375
 - ms_funcs.h, 381
- get_scan_results
 - wpa_driver_ops, 51
- get_ssid
 - wpa_driver_ops, 51
- get_status
 - eap_method, 14
- getKey
 - eap_method, 15
- GETU32
 - aes.c, 93
- has_reauth_data
 - eap_method, 15
- hash_nt_password_hash
 - ms_funcs.c, 375
 - ms_funcs.h, 381
- hexstr2bin

- common.c, 115
- common.h, 123
- hmac_md5
 - md5.c, 368
 - md5.h, 371
- hmac_md5_vector
 - md5.c, 368
 - md5.h, 371
- hmac_sha1
 - sha1.c, 426
 - sha1.h, 432
- hmac_sha1_vector
 - sha1.c, 427
 - sha1.h, 432
- hwaddr_aton
 - common.c, 116
 - common.h, 123
- id
 - wpa_ssid, 81
- inc_byte_array
 - common.c, 116
 - common.h, 123
- index
 - wpa_event_data::pmkid_candidate, 63
- init
 - eap_method, 15
 - wpa_driver_ops, 52
- init_for_reauth
 - eap_method, 16
- INT
 - config.c, 128
- INT_RANGE
 - config.c, 128
- isKeyAvailable
 - eap_method, 16
- key_id
 - wpa_ssid, 81
- key_mgmt
 - wpa_ssid, 81
- l2_packet.h, 346
 - l2_packet_deinit, 348
 - l2_packet_get_ip_addr, 348
 - l2_packet_get_own_addr, 348
 - l2_packet_init, 348
 - l2_packet_notify_auth_start, 349
 - l2_packet_send, 349
- l2_packet_deinit
 - l2_packet.h, 348
 - l2_packet_freebsd.c, 353
 - l2_packet_linux.c, 358
 - l2_packet_pcap.c, 362
- l2_packet_freebsd.c, 351
 - l2_packet_deinit, 353
 - l2_packet_get_ip_addr, 353
 - l2_packet_get_own_addr, 353
 - l2_packet_init, 354
 - l2_packet_notify_auth_start, 354
 - l2_packet_send, 354
- l2_packet_get_ip_addr
 - l2_packet.h, 348
 - l2_packet_freebsd.c, 353
 - l2_packet_linux.c, 358
 - l2_packet_pcap.c, 362
- l2_packet_get_own_addr
 - l2_packet.h, 348
 - l2_packet_freebsd.c, 353
 - l2_packet_linux.c, 358
 - l2_packet_pcap.c, 363
- l2_packet_init
 - l2_packet.h, 348
 - l2_packet_freebsd.c, 354
 - l2_packet_linux.c, 359
 - l2_packet_pcap.c, 363
- l2_packet_linux.c, 356
 - l2_packet_deinit, 358
 - l2_packet_get_ip_addr, 358
 - l2_packet_get_own_addr, 358
 - l2_packet_init, 359
 - l2_packet_notify_auth_start, 359
 - l2_packet_send, 360
- l2_packet_notify_auth_start
 - l2_packet.h, 349
 - l2_packet_freebsd.c, 354
 - l2_packet_linux.c, 359
 - l2_packet_pcap.c, 364
- l2_packet_pcap.c, 361
 - l2_packet_deinit, 362
 - l2_packet_get_ip_addr, 362
 - l2_packet_get_own_addr, 363
 - l2_packet_init, 363
 - l2_packet_notify_auth_start, 364
 - l2_packet_send, 364
- l2_packet_send
 - l2_packet.h, 349
 - l2_packet_freebsd.c, 354
 - l2_packet_linux.c, 360
 - l2_packet_pcap.c, 364
- LE_READ_4
 - driver_bsd.c, 174
- leap
 - wpa_ssid, 82
- main.c, 365
- md4_vector
 - crypto.c, 149

- crypto.h, 153
- crypto_gnutls.c, 156
- md5.c, 367
 - hmac_md5, 368
 - hmac_md5_vector, 368
 - md5_vector, 369
- md5.h, 370
 - hmac_md5, 371
 - hmac_md5_vector, 371
- md5_vector
 - crypto.h, 153
 - md5.c, 369
- mode
 - wpa_ssid, 82
- ms_funcs.c, 372
 - challenge_response, 373
 - generate_authenticator_response, 374
 - generate_nt_response, 374
 - get_asymmetric_start_key, 375
 - get_master_key, 375
 - hash_nt_password_hash, 375
 - new_password_encrypted_with_old_nt_password_hash, 376
 - nt_challenge_response, 376
 - nt_password_hash, 377
 - old_nt_password_hash_encrypted_with_new_nt_password_hash, 377
- ms_funcs.h, 378
 - challenge_response, 379
 - generate_authenticator_response, 379
 - generate_nt_response, 380
 - get_asymmetric_start_key, 380
 - get_master_key, 381
 - hash_nt_password_hash, 381
 - new_password_encrypted_with_old_nt_password_hash, 381
 - nt_challenge_response, 382
 - nt_password_hash, 382
 - old_nt_password_hash_encrypted_with_new_nt_password_hash, 383
- mschapv2_retry
 - wpa_ssid, 82
- name
 - wpa_driver_ops, 52
- new_password
 - wpa_ssid, 82
- new_password_encrypted_with_old_nt_password_hash
 - ms_funcs.c, 376
 - ms_funcs.h, 381
- next
 - wpa_ssid, 82
- non_leap
 - wpa_ssid, 82
- nt_challenge_response
 - ms_funcs.c, 376
 - ms_funcs.h, 382
- nt_password_hash
 - ms_funcs.c, 377
 - ms_funcs.h, 382
- num_prio
 - wpa_config, 39
- old_nt_password_hash_encrypted_with_new_nt_password_hash
 - ms_funcs.c, 377
 - ms_funcs.h, 383
- omac1_aes_128
 - aes_wrap.c, 101
 - aes_wrap.h, 108
- opensc_engine_path
 - eap_config, 11
 - eapol_ctx, 28
 - wpa_config, 39
- otp
 - wpa_ssid, 83
- pac_file
 - wpa_ssid, 83
- packed
 - wpa_i.h, 525
- passphrase
 - wpa_ssid, 83
- pbkdf2_sha1
 - sha1.c, 427
 - sha1.h, 433
- pcsc
 - wpa_ssid, 83
- pcsc_funcs.c, 384
 - scard_deinit, 386
 - scard_get_imsi, 386
 - scard_gsm_auth, 386
 - scard_init, 387
 - scard_set_pin, 387
 - scard_umts_auth, 388
- pcsc_funcs.h, 389
- pending_req_identity
 - wpa_ssid, 83
- pending_req_new_password
 - wpa_ssid, 83
- pending_req_otp
 - wpa_ssid, 84
- pending_req_passphrase
 - wpa_ssid, 84
- pending_req_password
 - wpa_ssid, 84
- pending_req_pin

- wpa_ssid, 84
- phase1
 - wpa_ssid, 84
- phase2
 - wpa_ssid, 85
- pid_file
 - wpa_params, 67
- pin
 - wpa_ssid, 85
- pkcs11_engine_path
 - eap_config, 11
 - eapol_ctx, 28
 - wpa_config, 39
- pkcs11_module_path
 - eap_config, 12
 - eapol_ctx, 28
 - wpa_config, 39
- pmksa_cache_add
 - preauth.c, 394
 - preauth.h, 402
- pmksa_cache_clear_current
 - preauth.c, 394
 - preauth.h, 403
- pmksa_cache_free
 - preauth.c, 394
 - preauth.h, 403
- pmksa_cache_get
 - preauth.c, 394
 - preauth.h, 403
- pmksa_cache_get_current
 - preauth.c, 395
 - preauth.h, 403
- pmksa_cache_list
 - preauth.c, 395
 - preauth.h, 404
- pmksa_cache_notify_reconfig
 - preauth.c, 395
 - preauth.h, 404
- pmksa_cache_set_current
 - preauth.c, 395
 - preauth.h, 404
- pmksa_candidate_add
 - preauth.c, 396
 - preauth.h, 405
- pmksa_candidate_free
 - preauth.c, 396
 - preauth.h, 405
- pnext
 - wpa_ssid, 85
- poll
 - wpa_driver_ops, 52
- preauth
 - eapol_ctx, 28
 - wpa_event_data::pmkid_candidate, 63
- preauth.c, 391
 - pmksa_cache_add, 394
 - pmksa_cache_clear_current, 394
 - pmksa_cache_free, 394
 - pmksa_cache_get, 394
 - pmksa_cache_get_current, 395
 - pmksa_cache_list, 395
 - pmksa_cache_notify_reconfig, 395
 - pmksa_cache_set_current, 395
 - pmksa_candidate_add, 396
 - pmksa_candidate_free, 396
 - rsn_preauth_candidate_process, 397
 - rsn_preauth_deinit, 397
 - rsn_preauth_get_status, 398
 - rsn_preauth_in_progress, 398
 - rsn_preauth_init, 398
 - rsn_preauth_scan_results, 399
- preauth.h, 401
 - pmksa_cache_add, 402
 - pmksa_cache_clear_current, 403
 - pmksa_cache_free, 403
 - pmksa_cache_get, 403
 - pmksa_cache_get_current, 403
 - pmksa_cache_list, 404
 - pmksa_cache_notify_reconfig, 404
 - pmksa_cache_set_current, 404
 - pmksa_candidate_add, 405
 - pmksa_candidate_free, 405
 - rsn_preauth_candidate_process, 405
 - rsn_preauth_deinit, 406
 - rsn_preauth_get_status, 406
 - rsn_preauth_in_progress, 407
 - rsn_preauth_init, 407
 - rsn_preauth_scan_results, 408
- preauth_test.c, 409
- priority
 - wpa_ssid, 85
- priv_netlink.h, 412
 - RTA_NEXT, 412
 - RTA_OK, 413
- private_key
 - wpa_ssid, 86
- private_key2
 - wpa_ssid, 86
- private_key2_passwd
 - wpa_ssid, 86
- private_key_passwd
 - wpa_ssid, 86
- proactive_key_caching
 - wpa_ssid, 86
- process
 - eap_method, 16
- PUTU32
 - aes.c, 93

- R0
 - sha1.c, 426
- R1
 - sha1.c, 426
- R3
 - sha1.c, 426
- R4
 - sha1.c, 426
- radius.c, 414
- radius_client.c, 417
- rc4
 - rc4.c, 420
 - rc4.h, 422
- rc4.c, 420
 - rc4, 420
 - rc4_skip, 421
- rc4.h, 422
 - rc4, 422
 - rc4_skip, 423
- rc4_skip
 - rc4.c, 421
 - rc4.h, 423
- remove_pmkid
 - wpa_driver_ops, 52
- req_ies
 - wpa_event_data::assoc_info, 59
- required_keys
 - eapol_config, 24
- resp_ies
 - wpa_event_data::assoc_info, 60
- rijndaelKeySetupDec
 - aes.c, 95
- rijndaelKeySetupEnc
 - aes.c, 95
- ROUND
 - aes.c, 93
- rsn_pmksa_cache, 33
- rsn_preauth_candidate_process
 - preauth.c, 397
 - preauth.h, 405
- rsn_preauth_deinit
 - preauth.c, 397
 - preauth.h, 406
- rsn_preauth_get_status
 - preauth.c, 398
 - preauth.h, 406
- rsn_preauth_in_progress
 - preauth.c, 398
 - preauth.h, 407
- rsn_preauth_init
 - preauth.c, 398
 - preauth.h, 407
- rsn_preauth_scan_results
 - preauth.c, 399
 - preauth.h, 408
- RTA_NEXT
 - priv_netlink.h, 412
- RTA_OK
 - priv_netlink.h, 413
- scan
 - wpa_driver_ops, 53
- scan_ssid
 - wpa_ssid, 87
- scard_ctx
 - eapol_ctx, 29
- scard_deinit
 - pesc_funcs.c, 386
- scard_get_imsi
 - pesc_funcs.c, 386
- scard_gsm_auth
 - pesc_funcs.c, 386
- scard_init
 - pesc_funcs.c, 387
- scard_set_pin
 - pesc_funcs.c, 387
- scard_umts_auth
 - pesc_funcs.c, 388
- send_eapol
 - wpa_driver_ops, 53
- set_auth_alg
 - wpa_driver_ops, 54
- set_bool
 - eapol_callbacks, 23
- set_config_blob
 - eapol_callbacks, 23
 - eapol_ctx, 29
- set_countermeasures
 - wpa_driver_ops, 54
- set_drop_unencrypted
 - wpa_driver_ops, 54
- set_int
 - eapol_callbacks, 23
- set_key
 - wpa_driver_ops, 54
- set_param
 - wpa_driver_ops, 55
- set_wep_key
 - eapol_ctx, 29
- set_wpa
 - wpa_driver_ops, 55
- sha1.c, 424
 - blk, 425
 - blk0, 425
 - hmac_sha1, 426
 - hmac_sha1_vector, 427
 - pbkdf2_sha1, 427
 - R0, 426

- R1, [426](#)
- R3, [426](#)
- R4, [426](#)
- sha1_prf, [428](#)
- sha1_t_prf, [428](#)
- sha1_transform, [429](#)
- sha1_vector, [429](#)
- tls_prf, [429](#)
- sha1.h, [431](#)
 - hmac_sha1, [432](#)
 - hmac_sha1_vector, [432](#)
 - pbkdf2_sha1, [433](#)
 - sha1_prf, [433](#)
 - sha1_t_prf, [434](#)
 - tls_prf, [434](#)
- sha1_prf
 - sha1.c, [428](#)
 - sha1.h, [433](#)
- sha1_t_prf
 - sha1.c, [428](#)
 - sha1.h, [434](#)
- sha1_transform
 - crypto.h, [153](#)
 - sha1.c, [429](#)
- sha1_vector
 - crypto.h, [154](#)
 - sha1.c, [429](#)
- SM_ENTRY
 - eap.c, [222](#)
 - eapol_sm.c, [307](#)
- SM_STATE
 - eap.c, [222](#)
 - eapol_sm.c, [307](#)
- ssid
 - wpa_config, [39](#)
 - wpa_ssid, [87](#)
- STR
 - config.c, [128](#)
- STR_RANGE
 - config.c, [128](#)
- subject_match
 - wpa_ssid, [87](#)
- subject_match2
 - wpa_ssid, [87](#)
- tls.h, [436](#)
 - tls_connection_client_hello_ext, [438](#)
 - tls_connection_decrypt, [439](#)
 - tls_connection_deinit, [439](#)
 - tls_connection_enable_workaround, [439](#)
 - tls_connection_encrypt, [440](#)
 - tls_connection_established, [440](#)
 - tls_connection_get_failed, [441](#)
 - tls_connection_get_keyblock_size, [441](#)
 - tls_connection_get_keys, [441](#)
 - tls_connection_get_read_alerts, [441](#)
 - tls_connection_get_write_alerts, [442](#)
 - tls_connection_handshake, [442](#)
 - tls_connection_init, [443](#)
 - tls_connection_resumed, [443](#)
 - tls_connection_server_handshake, [443](#)
 - tls_connection_set_anon_dh, [443](#)
 - tls_connection_set_master_key, [444](#)
 - tls_connection_set_params, [444](#)
 - tls_connection_set_verify, [444](#)
 - tls_connection_shutdown, [445](#)
 - tls_deinit, [445](#)
 - tls_get_cipher, [445](#)
 - tls_get_errors, [446](#)
 - tls_global_ca_cert, [446](#)
 - tls_global_client_cert, [446](#)
 - tls_global_private_key, [446](#)
 - tls_global_set_verify, [447](#)
 - tls_init, [447](#)
- tls_connection_client_hello_ext
 - tls.h, [438](#)
- tls_connection_decrypt
 - tls.h, [439](#)
 - tls_gnutls.c, [450](#)
 - tls_openssl.c, [464](#)
 - tls_schannel.c, [477](#)
- tls_connection_deinit
 - tls.h, [439](#)
 - tls_gnutls.c, [451](#)
 - tls_openssl.c, [465](#)
 - tls_schannel.c, [477](#)
- tls_connection_enable_workaround
 - tls.h, [439](#)
 - tls_gnutls.c, [451](#)
 - tls_openssl.c, [465](#)
 - tls_schannel.c, [477](#)
- tls_connection_encrypt
 - tls.h, [440](#)
 - tls_gnutls.c, [451](#)
 - tls_openssl.c, [465](#)
 - tls_schannel.c, [478](#)
- tls_connection_established
 - tls.h, [440](#)
 - tls_gnutls.c, [452](#)
 - tls_openssl.c, [466](#)
 - tls_schannel.c, [478](#)
- tls_connection_get_failed
 - tls.h, [441](#)
 - tls_gnutls.c, [452](#)
 - tls_openssl.c, [466](#)
 - tls_schannel.c, [478](#)
- tls_connection_get_keyblock_size
 - tls.h, [441](#)

- tls_openssl.c, 466
- tls_connection_get_keys
 - tls.h, 441
 - tls_gnutls.c, 452
 - tls_openssl.c, 466
 - tls_schannel.c, 479
- tls_connection_get_read_alerts
 - tls.h, 441
 - tls_gnutls.c, 453
 - tls_openssl.c, 467
 - tls_schannel.c, 479
- tls_connection_get_write_alerts
 - tls.h, 442
 - tls_gnutls.c, 453
 - tls_openssl.c, 467
 - tls_schannel.c, 479
- tls_connection_handshake
 - tls.h, 442
 - tls_gnutls.c, 453
 - tls_openssl.c, 467
 - tls_schannel.c, 479
- tls_connection_init
 - tls.h, 443
 - tls_gnutls.c, 454
 - tls_openssl.c, 468
 - tls_schannel.c, 480
- tls_connection_params, 34
- tls_connection_resumed
 - tls.h, 443
 - tls_gnutls.c, 454
 - tls_openssl.c, 468
 - tls_schannel.c, 480
- tls_connection_server_handshake
 - tls.h, 443
 - tls_gnutls.c, 455
 - tls_openssl.c, 468
 - tls_schannel.c, 481
- tls_connection_set_anon_dh
 - tls.h, 443
 - tls_gnutls.c, 455
 - tls_openssl.c, 469
 - tls_schannel.c, 481
- tls_connection_set_master_key
 - tls.h, 444
- tls_connection_set_params
 - tls.h, 444
 - tls_gnutls.c, 455
 - tls_openssl.c, 469
 - tls_schannel.c, 481
- tls_connection_set_verify
 - tls.h, 444
 - tls_gnutls.c, 456
 - tls_openssl.c, 470
 - tls_schannel.c, 482
- tls_connection_shutdown
 - tls.h, 445
 - tls_gnutls.c, 456
 - tls_openssl.c, 470
 - tls_schannel.c, 482
- tls_deinit
 - tls.h, 445
 - tls_gnutls.c, 456
 - tls_none.c, 460
 - tls_openssl.c, 470
 - tls_schannel.c, 482
- tls_get_cipher
 - tls.h, 445
 - tls_gnutls.c, 457
 - tls_openssl.c, 471
 - tls_schannel.c, 483
- tls_get_errors
 - tls.h, 446
 - tls_gnutls.c, 457
 - tls_openssl.c, 471
 - tls_schannel.c, 483
- tls_global_ca_cert
 - tls.h, 446
 - tls_gnutls.c, 457
 - tls_openssl.c, 471
 - tls_schannel.c, 483
- tls_global_client_cert
 - tls.h, 446
 - tls_gnutls.c, 457
 - tls_openssl.c, 472
 - tls_schannel.c, 484
- tls_global_private_key
 - tls.h, 446
 - tls_gnutls.c, 458
 - tls_openssl.c, 472
 - tls_schannel.c, 484
- tls_global_set_verify
 - tls.h, 447
 - tls_gnutls.c, 458
 - tls_openssl.c, 472
 - tls_schannel.c, 484
- tls_gnutls.c, 448
 - tls_connection_decrypt, 450
 - tls_connection_deinit, 451
 - tls_connection_enable_workaround, 451
 - tls_connection_encrypt, 451
 - tls_connection_established, 452
 - tls_connection_get_failed, 452
 - tls_connection_get_keys, 452
 - tls_connection_get_read_alerts, 453
 - tls_connection_get_write_alerts, 453
 - tls_connection_handshake, 453
 - tls_connection_init, 454
 - tls_connection_resumed, 454

- tls_connection_server_handshake, 455
- tls_connection_set_anon_dh, 455
- tls_connection_set_params, 455
- tls_connection_set_verify, 456
- tls_connection_shutdown, 456
- tls_deinit, 456
- tls_get_cipher, 457
- tls_get_errors, 457
- tls_global_ca_cert, 457
- tls_global_client_cert, 457
- tls_global_private_key, 458
- tls_global_set_verify, 458
- tls_init, 458
- tls_init
 - tls.h, 447
 - tls_gnutls.c, 458
 - tls_none.c, 461
 - tls_openssl.c, 473
 - tls_schannel.c, 484
- tls_none.c, 460
 - tls_deinit, 460
 - tls_init, 461
- tls_openssl.c, 462
 - tls_connection_decrypt, 464
 - tls_connection_deinit, 465
 - tls_connection_enable_workaround, 465
 - tls_connection_encrypt, 465
 - tls_connection_established, 466
 - tls_connection_get_failed, 466
 - tls_connection_get_keyblock_size, 466
 - tls_connection_get_keys, 466
 - tls_connection_get_read_alerts, 467
 - tls_connection_get_write_alerts, 467
 - tls_connection_handshake, 467
 - tls_connection_init, 468
 - tls_connection_resumed, 468
 - tls_connection_server_handshake, 468
 - tls_connection_set_anon_dh, 469
 - tls_connection_set_params, 469
 - tls_connection_set_verify, 470
 - tls_connection_shutdown, 470
 - tls_deinit, 470
 - tls_get_cipher, 471
 - tls_get_errors, 471
 - tls_global_ca_cert, 471
 - tls_global_client_cert, 472
 - tls_global_private_key, 472
 - tls_global_set_verify, 472
 - tls_init, 473
- tls_prf
 - sha1.c, 429
 - sha1.h, 434
- tls_schannel.c, 474
 - tls_connection_decrypt, 477
 - tls_connection_deinit, 477
 - tls_connection_enable_workaround, 477
 - tls_connection_encrypt, 478
 - tls_connection_established, 478
 - tls_connection_get_failed, 478
 - tls_connection_get_keys, 479
 - tls_connection_get_read_alerts, 479
 - tls_connection_get_write_alerts, 479
 - tls_connection_handshake, 479
 - tls_connection_init, 480
 - tls_connection_resumed, 480
 - tls_connection_server_handshake, 481
 - tls_connection_set_anon_dh, 481
 - tls_connection_set_params, 481
 - tls_connection_set_verify, 482
 - tls_connection_shutdown, 482
 - tls_deinit, 482
 - tls_get_cipher, 483
 - tls_get_errors, 483
 - tls_global_ca_cert, 483
 - tls_global_client_cert, 484
 - tls_global_private_key, 484
 - tls_global_set_verify, 484
 - tls_init, 484
- update_config
 - wpa_config, 39
- wait_for_interface
 - wpa_params, 67
- win_if_list.c, 486
- wpa.c, 487
 - wpa_parse_wpa_ie, 491
 - wpa_sm_aborted_cached, 491
 - wpa_sm_deinit, 491
 - wpa_sm_get_mib, 491
 - wpa_sm_get_param, 492
 - wpa_sm_get_status, 492
 - wpa_sm_init, 492
 - wpa_sm_key_request, 493
 - wpa_sm_notify_assoc, 493
 - wpa_sm_notify_disassoc, 494
 - wpa_sm_parse_own_wpa_ie, 494
 - wpa_sm_rx_eapol, 494
 - wpa_sm_set_ap_rsn_ie, 495
 - wpa_sm_set_ap_wpa_ie, 495
 - wpa_sm_set_assoc_wpa_ie, 496
 - wpa_sm_set_assoc_wpa_ie_default, 496
 - wpa_sm_set_config, 497
 - wpa_sm_set_eapol, 497
 - wpa_sm_set_fast_reauth, 497
 - wpa_sm_set_ifname, 497
 - wpa_sm_set_own_addr, 498
 - wpa_sm_set_param, 498

- wpa_sm_set_pmk, 498
- wpa_sm_set_pmk_from_pmksa, 498
- wpa_sm_set_scard_ctx, 499
- wpa.h, 500
 - wpa_parse_wpa_ie, 503
 - wpa_sm_aborted_cached, 503
 - wpa_sm_deinit, 504
 - wpa_sm_get_mib, 504
 - wpa_sm_get_param, 504
 - wpa_sm_get_status, 504
 - wpa_sm_init, 505
 - wpa_sm_key_request, 505
 - wpa_sm_notify_assoc, 506
 - wpa_sm_notify_disassoc, 506
 - wpa_sm_parse_own_wpa_ie, 506
 - wpa_sm_rx_eapol, 507
 - wpa_sm_set_ap_rsn_ie, 507
 - wpa_sm_set_ap_wpa_ie, 508
 - wpa_sm_set_assoc_wpa_ie, 508
 - wpa_sm_set_assoc_wpa_ie_default, 509
 - wpa_sm_set_config, 509
 - wpa_sm_set_eapol, 509
 - wpa_sm_set_fast_reauth, 510
 - wpa_sm_set_ifname, 510
 - wpa_sm_set_own_addr, 510
 - wpa_sm_set_param, 510
 - wpa_sm_set_pmk, 511
 - wpa_sm_set_pmk_from_pmksa, 511
 - wpa_sm_set_scard_ctx, 511
- WPA_4WAY_HANDSHAKE
 - defs.h, 164
- WPA_ASSOCIATED
 - defs.h, 164
- WPA_ASSOCIATING
 - defs.h, 164
- wpa_blacklist_add
 - wpa_supplicant.c, 531
 - wpa_supplicant_i.h, 554
- wpa_blacklist_clear
 - wpa_supplicant.c, 531
 - wpa_supplicant_i.h, 555
- wpa_blacklist_get
 - wpa_supplicant.c, 532
 - wpa_supplicant_i.h, 555
- wpa_clear_keys
 - wpa_supplicant.c, 532
 - wpa_supplicant_i.h, 555
- wpa_cli.c, 512
- WPA_COMPLETED
 - defs.h, 164
- wpa_config, 36
 - ap_scan, 37
 - ctrl_interface, 37
 - dot11RSNACfgPMKLifetime, 38
 - dot11RSNACfgPMKReauthThreshold, 38
 - dot11RSNACfgSATimeout, 38
 - driver_param, 38
 - eapol_version, 38
 - fast_reauth, 38
 - num_prio, 39
 - opensc_engine_path, 39
 - pkcs11_engine_path, 39
 - pkcs11_module_path, 39
 - ssid, 39
 - update_config, 39
- wpa_config_add_network
 - config.c, 129
 - config.h, 136
- wpa_config_add_prio_network
 - config.c, 129
 - config.h, 136
- wpa_config_alloc_empty
 - config.c, 129
 - config.h, 136
- wpa_config_allowed_eap_method
 - config.c, 129
 - config_ssid.h, 147
- wpa_config_blob, 41
- wpa_config_free
 - config.c, 130
 - config.h, 136
- wpa_config_free_blob
 - config.c, 130
 - config.h, 137
- wpa_config_free_ssid
 - config.c, 130
 - config.h, 137
- wpa_config_get
 - config.c, 130
 - config.h, 137
- wpa_config_get_blob
 - config.c, 131
 - config.h, 137
- wpa_config_get_network
 - config.c, 131
 - config.h, 138
- wpa_config_read
 - config.h, 138
 - config_file.c, 143
- wpa_config_remove_blob
 - config.c, 131
 - config.h, 139
- wpa_config_remove_network
 - config.c, 132
 - config.h, 139
- wpa_config_set
 - config.c, 132

- config.h, 139
- wpa_config_set_blob
 - config.c, 132
 - config.h, 140
- wpa_config_set_network_defaults
 - config.c, 133
 - config.h, 140
- wpa_config_update_psk
 - config.c, 133
 - config.h, 140
- wpa_config_write
 - config.h, 141
 - config_file.c, 143
- wpa_ctrl, 42
- wpa_ctrl.c, 514
 - wpa_ctrl_attach, 515
 - wpa_ctrl_close, 515
 - wpa_ctrl_detach, 516
 - wpa_ctrl_get_fd, 516
 - wpa_ctrl_open, 516
 - wpa_ctrl_pending, 517
 - wpa_ctrl_recv, 517
 - wpa_ctrl_request, 517
- wpa_ctrl.h, 519
 - wpa_ctrl_attach, 521
 - wpa_ctrl_close, 522
 - wpa_ctrl_detach, 522
 - wpa_ctrl_get_fd, 522
 - wpa_ctrl_open, 522
 - wpa_ctrl_pending, 523
 - wpa_ctrl_recv, 523
 - WPA_CTRL_REQ, 520
 - wpa_ctrl_request, 523
 - WPA_CTRL_RSP, 520
 - WPA_EVENT_CONNECTED, 520
 - WPA_EVENT_DISCONNECTED, 520
 - WPA_EVENT_EAP_FAILURE, 520
 - WPA_EVENT_EAP_METHOD, 521
 - WPA_EVENT_EAP_NOTIFICATION, 521
 - WPA_EVENT_EAP_STARTED, 521
 - WPA_EVENT_EAP_SUCCESS, 521
 - WPA_EVENT_PASSWORD_CHANGED, 521
 - WPA_EVENT_TERMINATING, 521
- wpa_ctrl_attach
 - wpa_ctrl.c, 515
 - wpa_ctrl.h, 521
- wpa_ctrl_close
 - wpa_ctrl.c, 515
 - wpa_ctrl.h, 522
- wpa_ctrl_detach
 - wpa_ctrl.c, 516
 - wpa_ctrl.h, 522
- wpa_ctrl_dst, 43
- wpa_ctrl_get_fd
 - wpa_ctrl.c, 516
 - wpa_ctrl.h, 522
- wpa_ctrl_open
 - wpa_ctrl.c, 516
 - wpa_ctrl.h, 522
- wpa_ctrl_pending
 - wpa_ctrl.c, 517
 - wpa_ctrl.h, 523
- wpa_ctrl_recv
 - wpa_ctrl.c, 517
 - wpa_ctrl.h, 523
- WPA_CTRL_REQ
 - wpa_ctrl.h, 520
- wpa_ctrl_request
 - wpa_ctrl.c, 517
 - wpa_ctrl.h, 523
- WPA_CTRL_RSP
 - wpa_ctrl.h, 520
- wpa_debug_print_timestamp
 - common.c, 116
 - common.h, 124
- wpa_debug_show_keys
 - wpa_params, 68
- WPA_DISCONNECTED
 - defs.h, 164
- wpa_driver_associate_params, 44
 - auth_alg, 44
 - bssid, 44
 - freq, 45
 - wpa_ie, 45
- wpa_driver_atmel_ops
 - driver_atmel.c, 168
- wpa_driver_broadcom_ops
 - driver_broadcom.c, 171
- wpa_driver_bsd_ops
 - driver_bsd.c, 174
- wpa_driver_capa, 46
- wpa_driver_hostap_ops
 - driver_hostap.c, 176
- wpa_driver_ipw_ops
 - driver_ipw.c, 180
- wpa_driver_madwifi_ops
 - driver_madwifi.c, 184
- wpa_driver_ndis_ops
 - driver_ndis.c, 188
- wpa_driver_ndiswrapper_ops
 - driver_ndiswrapper.c, 193
- wpa_driver_ops, 47
 - add_pmkid, 48
 - associate, 49
 - deauthenticate, 49
 - deinit, 49

- desc, 49
- disassociate, 49
- flush_pmkid, 50
- get_bssid, 50
- get_capa, 50
- get_ifname, 50
- get_mac_addr, 51
- get_scan_results, 51
- get_ssid, 51
- init, 52
- name, 52
- poll, 52
- remove_pmkid, 52
- scan, 53
- send_eapol, 53
- set_auth_alg, 54
- set_countermeasures, 54
- set_drop_unencrypted, 54
- set_key, 54
- set_param, 55
- set_wpa, 55
- wpa_driver_prism54_ops
 - driver_prism54.c, 196
- wpa_driver_test_ops
 - driver_test.c, 198
- wpa_driver_wext_deinit
 - driver_wext.c, 202
 - driver_wext.h, 209
- wpa_driver_wext_get_bssid
 - driver_wext.c, 202
 - driver_wext.h, 210
- wpa_driver_wext_get_ifflags
 - driver_wext.c, 202
 - driver_wext.h, 210
- wpa_driver_wext_get_scan_results
 - driver_wext.c, 202
 - driver_wext.h, 210
- wpa_driver_wext_get_ssid
 - driver_wext.c, 203
 - driver_wext.h, 211
- wpa_driver_wext_init
 - driver_wext.c, 203
 - driver_wext.h, 211
- wpa_driver_wext_ops
 - driver_wext.c, 207
- wpa_driver_wext_scan
 - driver_wext.c, 204
 - driver_wext.h, 211
- wpa_driver_wext_scan_timeout
 - driver_wext.c, 204
 - driver_wext.h, 212
- wpa_driver_wext_set_bssid
 - driver_wext.c, 205
 - driver_wext.h, 212
- wpa_driver_wext_set_freq
 - driver_wext.c, 205
 - driver_wext.h, 213
- wpa_driver_wext_set_ifflags
 - driver_wext.c, 205
 - driver_wext.h, 213
- wpa_driver_wext_set_key
 - driver_wext.c, 205
 - driver_wext.h, 213
- wpa_driver_wext_set_mode
 - driver_wext.c, 206
 - driver_wext.h, 214
- wpa_driver_wext_set_ssid
 - driver_wext.c, 206
 - driver_wext.h, 214
- wpa_driver_wired_ops
 - driver_wired.c, 216
- WPA_EVENT_CONNECTED
 - wpa_ctrl.h, 520
- wpa_event_data, 57
 - assoc_info, 58
- wpa_event_data::assoc_info, 59
 - beacon_ies, 59
 - req_ies, 59
 - resp_ies, 60
- wpa_event_data::interface_status, 61
- wpa_event_data::michael_mic_failure, 62
- wpa_event_data::pmkid_candidate, 63
 - bssid, 63
 - index, 63
 - preauth, 63
- WPA_EVENT_DISCONNECTED
 - wpa_ctrl.h, 520
- WPA_EVENT_EAP_FAILURE
 - wpa_ctrl.h, 520
- WPA_EVENT_EAP_METHOD
 - wpa_ctrl.h, 521
- WPA_EVENT_EAP_NOTIFICATION
 - wpa_ctrl.h, 521
- WPA_EVENT_EAP_STARTED
 - wpa_ctrl.h, 521
- WPA_EVENT_EAP_SUCCESS
 - wpa_ctrl.h, 521
- WPA_EVENT_PASSWORD_CHANGED
 - wpa_ctrl.h, 521
- WPA_EVENT_TERMINATING
 - wpa_ctrl.h, 521
- wpa_event_type
 - wpa_supplicant.h, 548, 549
- WPA_GET_BE32
 - common.h, 122
- wpa_global, 64
- WPA_GROUP_HANDSHAKE
 - defs.h, 164

- wpa_hexdump
 - common.c, 116
 - common.h, 124
- wpa_hexdump_ascii
 - common.c, 117
 - common.h, 124
- wpa_hexdump_ascii_key
 - common.c, 117
 - common.h, 124
- wpa_hexdump_key
 - common.c, 117
 - common.h, 125
- wpa_i.h, 525
 - packed, 525
- wpa_ie
 - wpa_driver_associate_params, 45
- WPA_INACTIVE
 - defs.h, 164
- wpa_interface, 65
 - confname, 65
 - ctrl_interface, 65
 - driver_param, 65
- wpa_msg
 - wpa_supplicant.h, 550
- wpa_params, 67
 - pid_file, 67
 - wait_for_interface, 67
 - wpa_debug_show_keys, 68
- wpa_parse_wpa_ie
 - wpa.c, 491
 - wpa.h, 503
- wpa_passphrase.c, 526
- wpa_printf
 - common.c, 118
 - common.h, 125
- wpa_ptk, 69
- WPA_PUT_BE16
 - common.h, 122
- WPA_PUT_LE16
 - common.h, 123
- wpa_scan_result, 70
- WPA_SCANNING
 - defs.h, 164
- wpa_sm, 71
- wpa_sm_aborted_cached
 - wpa.c, 491
 - wpa.h, 503
- wpa_sm_deinit
 - wpa.c, 491
 - wpa.h, 504
- wpa_sm_get_mib
 - wpa.c, 491
 - wpa.h, 504
- wpa_sm_get_param
 - wpa.c, 492
 - wpa.h, 504
- wpa_sm_get_status
 - wpa.c, 492
 - wpa.h, 504
- wpa_sm_init
 - wpa.c, 492
 - wpa.h, 505
- wpa_sm_key_request
 - wpa.c, 493
 - wpa.h, 505
- wpa_sm_notify_assoc
 - wpa.c, 493
 - wpa.h, 506
- wpa_sm_notify_disassoc
 - wpa.c, 494
 - wpa.h, 506
- wpa_sm_parse_own_wpa_ie
 - wpa.c, 494
 - wpa.h, 506
- wpa_sm_rx_eapol
 - wpa.c, 494
 - wpa.h, 507
- wpa_sm_set_ap_rsn_ie
 - wpa.c, 495
 - wpa.h, 507
- wpa_sm_set_ap_wpa_ie
 - wpa.c, 495
 - wpa.h, 508
- wpa_sm_set_assoc_wpa_ie
 - wpa.c, 496
 - wpa.h, 508
- wpa_sm_set_assoc_wpa_ie_default
 - wpa.c, 496
 - wpa.h, 509
- wpa_sm_set_config
 - wpa.c, 497
 - wpa.h, 509
- wpa_sm_set_eapol
 - wpa.c, 497
 - wpa.h, 509
- wpa_sm_set_fast_reauth
 - wpa.c, 497
 - wpa.h, 510
- wpa_sm_set_ifname
 - wpa.c, 497
 - wpa.h, 510
- wpa_sm_set_own_addr
 - wpa.c, 498
 - wpa.h, 510
- wpa_sm_set_param
 - wpa.c, 498
 - wpa.h, 510
- wpa_sm_set_pmk
 - wpa.c, 498
 - wpa.h, 510

- wpa.c, 498
- wpa.h, 511
- wpa_sm_set_pmk_from_pmksa
 - wpa.c, 498
 - wpa.h, 511
- wpa_sm_set_scard_ctx
 - wpa.c, 499
 - wpa.h, 511
- wpa_ssid, 73
 - altsubject_match, 77
 - altsubject_match2, 78
 - anonymous_identity, 78
 - auth_alg, 78
 - bssid, 78
 - ca_cert, 78
 - ca_cert2, 79
 - ca_path, 79
 - ca_path2, 79
 - client_cert, 79
 - client_cert2, 79
 - dh_file, 80
 - dh_file2, 80
 - disabled, 80
 - eap_methods, 80
 - eap_workaround, 80
 - eappsk_len, 81
 - engine, 81
 - engine_id, 81
 - id, 81
 - key_id, 81
 - key_mgmt, 81
 - leap, 82
 - mode, 82
 - mschapv2_retry, 82
 - new_password, 82
 - next, 82
 - non_leap, 82
 - otp, 83
 - pac_file, 83
 - passphrase, 83
 - pcsc, 83
 - pending_req_identity, 83
 - pending_req_new_password, 83
 - pending_req_otp, 84
 - pending_req_passphrase, 84
 - pending_req_password, 84
 - pending_req_pin, 84
 - phase1, 84
 - phase2, 85
 - pin, 85
 - pnext, 85
 - priority, 85
 - private_key, 86
 - private_key2, 86
 - private_key2_passwd, 86
 - private_key_passwd, 86
 - proactive_key_caching, 86
 - scan_ssid, 87
 - ssid, 87
 - subject_match, 87
 - subject_match2, 87
- wpa_ssid_txt
 - wpa_supplicant.c, 532
 - wpa_supplicant.h, 550
- wpa_states
 - defs.h, 164
- wpa_supplicant, 88
- wpa_supplicant.c, 527
 - wpa_blacklist_add, 531
 - wpa_blacklist_clear, 531
 - wpa_blacklist_get, 532
 - wpa_clear_keys, 532
 - wpa_ssid_txt, 532
 - wpa_supplicant_add_iface, 533
 - wpa_supplicant_associate, 533
 - wpa_supplicant_cancel_auth_timeout, 534
 - wpa_supplicant_cancel_scan, 534
 - wpa_supplicant_deauthenticate, 535
 - wpa_supplicant_deinit, 535
 - wpa_supplicant_disassociate, 535
 - wpa_supplicant_driver_init, 536
 - wpa_supplicant_get_iface, 537
 - wpa_supplicant_get_scan_results, 537
 - wpa_supplicant_get_ssid, 537
 - wpa_supplicant_get_state, 538
 - wpa_supplicant_init, 538
 - wpa_supplicant_initiate_eapol, 538
 - wpa_supplicant_license, 544
 - wpa_supplicant_reload_configuration, 539
 - wpa_supplicant_remove_iface, 540
 - wpa_supplicant_req_auth_timeout, 540
 - wpa_supplicant_req_scan, 541
 - wpa_supplicant_run, 541
 - wpa_supplicant_rx_eapol, 542
 - wpa_supplicant_set_non_wpa_policy, 542
 - wpa_supplicant_set_state, 543
 - wpa_supplicant_set_suites, 543
 - wpa_supplicant_state_txt, 544
 - wpa_supplicant_version, 544
- wpa_supplicant.h
 - EVENT_ASSOC, 549
 - EVENT_ASSOCINFO, 549
 - EVENT_DISASSOC, 549
 - EVENT_INTERFACE_STATUS, 549
 - EVENT_MICHAEL_MIC_FAILURE, 549
 - EVENT_PMKID_CANDIDATE, 549
 - EVENT_SCAN_RESULTS, 549
- wpa_supplicant.h, 546

- wpa_event_type, 548, 549
- wpa_msg, 550
- wpa_ssid_txt, 550
- wpa_supplicant_event, 550
- wpa_supplicant_rx_eapol, 551
- wpa_supplicant_add_iface
 - wpa_supplicant.c, 533
 - wpa_supplicant_i.h, 555
- wpa_supplicant_associate
 - wpa_supplicant.c, 533
 - wpa_supplicant_i.h, 556
- wpa_supplicant_cancel_auth_timeout
 - wpa_supplicant.c, 534
 - wpa_supplicant_i.h, 557
- wpa_supplicant_cancel_scan
 - wpa_supplicant.c, 534
 - wpa_supplicant_i.h, 557
- wpa_supplicant_ctrl_iface_deinit
 - ctrl_iface.c, 159
- wpa_supplicant_ctrl_iface_init
 - ctrl_iface.c, 159
- wpa_supplicant_ctrl_iface_send
 - ctrl_iface.c, 160
- wpa_supplicant_ctrl_iface_wait
 - ctrl_iface.c, 160
- wpa_supplicant_deauthenticate
 - wpa_supplicant.c, 535
 - wpa_supplicant_i.h, 558
- wpa_supplicant_deinit
 - wpa_supplicant.c, 535
 - wpa_supplicant_i.h, 558
- wpa_supplicant_disassociate
 - wpa_supplicant.c, 535
 - wpa_supplicant_i.h, 558
- wpa_supplicant_driver_init
 - wpa_supplicant.c, 536
 - wpa_supplicant_i.h, 559
- wpa_supplicant_event
 - events.c, 345
 - wpa_supplicant.h, 550
- wpa_supplicant_get_iface
 - wpa_supplicant.c, 537
 - wpa_supplicant_i.h, 560
- wpa_supplicant_get_scan_results
 - wpa_supplicant.c, 537
 - wpa_supplicant_i.h, 560
- wpa_supplicant_get_ssid
 - wpa_supplicant.c, 537
 - wpa_supplicant_i.h, 560
- wpa_supplicant_get_state
 - wpa_supplicant.c, 538
- wpa_supplicant_global_ctrl_iface_deinit
 - ctrl_iface.c, 161
- wpa_supplicant_global_ctrl_iface_init
 - ctrl_iface.c, 161
- wpa_supplicant_i.h, 552
 - wpa_blacklist_add, 554
 - wpa_blacklist_clear, 555
 - wpa_blacklist_get, 555
 - wpa_clear_keys, 555
 - wpa_supplicant_add_iface, 555
 - wpa_supplicant_associate, 556
 - wpa_supplicant_cancel_auth_timeout, 557
 - wpa_supplicant_cancel_scan, 557
 - wpa_supplicant_deauthenticate, 558
 - wpa_supplicant_deinit, 558
 - wpa_supplicant_disassociate, 558
 - wpa_supplicant_driver_init, 559
 - wpa_supplicant_get_iface, 560
 - wpa_supplicant_get_scan_results, 560
 - wpa_supplicant_get_ssid, 560
 - wpa_supplicant_init, 561
 - wpa_supplicant_initiate_eapol, 561
 - wpa_supplicant_reload_configuration, 562
 - wpa_supplicant_remove_iface, 563
 - wpa_supplicant_req_auth_timeout, 563
 - wpa_supplicant_req_scan, 564
 - wpa_supplicant_run, 564
 - wpa_supplicant_scard_init, 565
 - wpa_supplicant_set_non_wpa_policy, 565
 - wpa_supplicant_set_state, 566
 - wpa_supplicant_set_suites, 566
 - wpa_supplicant_state_txt, 567
- wpa_supplicant_init
 - wpa_supplicant.c, 538
 - wpa_supplicant_i.h, 561
- wpa_supplicant_initiate_eapol
 - wpa_supplicant.c, 538
 - wpa_supplicant_i.h, 561
- wpa_supplicant_license
 - wpa_supplicant.c, 544
- wpa_supplicant_reload_configuration
 - wpa_supplicant.c, 539
 - wpa_supplicant_i.h, 562
- wpa_supplicant_remove_iface
 - wpa_supplicant.c, 540
 - wpa_supplicant_i.h, 563
- wpa_supplicant_req_auth_timeout
 - wpa_supplicant.c, 540
 - wpa_supplicant_i.h, 563
- wpa_supplicant_req_scan
 - wpa_supplicant.c, 541
 - wpa_supplicant_i.h, 564
- wpa_supplicant_run
 - wpa_supplicant.c, 541
 - wpa_supplicant_i.h, 564
- wpa_supplicant_rx_eapol
 - wpa_supplicant.c, 542

- wpa_supplicant.h, [551](#)
- wpa_supplicant_scard_init
 - events.c, [345](#)
 - wpa_supplicant_i.h, [565](#)
- wpa_supplicant_set_non_wpa_policy
 - wpa_supplicant.c, [542](#)
 - wpa_supplicant_i.h, [565](#)
- wpa_supplicant_set_state
 - wpa_supplicant.c, [543](#)
 - wpa_supplicant_i.h, [566](#)
- wpa_supplicant_set_suites
 - wpa_supplicant.c, [543](#)
 - wpa_supplicant_i.h, [566](#)
- wpa_supplicant_state_txt
 - wpa_supplicant.c, [544](#)
 - wpa_supplicant_i.h, [567](#)
- wpa_supplicant_version
 - wpa_supplicant.c, [544](#)